实验 2　shell 程序设计

浙江大学实验报告

课程名称：Linux 程序设计　实验类型：设计型

学生姓名：　专业：计算机科学与技术　学号：

电子邮件地址：

实验日期：2020 年 8 月 03 日

一、　实验环境

配置信息

- CPU：Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz
- 内存：16.0 GB

系统信息

- 主系统：Microsoft Windows 10 Pro (64 位)
- 子系统：
- Windows Subsystem for Linux: Ubuntu
- 发行版本：Ubuntu 18.04.1 LTS
- 内核：Linux version 5.4.0-42-generic (buildd@lgw01-amd64-023) (gcc version 7.5.0)

二、　实验内容和结果及分析

- 统计文件数目

需求描述：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，目录的路径名由参数传入。

- 设计思想：利用 ls 以及 grep 功能，列出结果并进行正则匹配
- 代码文件：

```
1. #!/bin/bash
2. #UTF-8
3. #**********************************************************
   *****************
4. #名称：  countFile.sh
```

```
#作者：  刘星烨
#学号：  3180105954
#功能：  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，目录的路径名由参数传入
#参数：  目录的路径
#************************************************************
#判断是否正确输入了路径参数
if test $# -ne '1'
then
echo -e "Input error!"
exit 1
fi
#统计传入的目录中不同类型文件的数量
echo -en "普通文件：\t"
ls -Al $1 | grep -c "^-"
echo -en "目录文件：\t"
ls -Al $1 | grep -c "^d"
echo -en "可执行文件：\t"
ls -Al $1 | grep -c "^-[rw-]*[sx]"
#统计所有文件的字节数
echo -en "普通文件字节数：\t"
#使用变量 count 计数
declare -i count=0
#对于目录下的 ls 结果进行正则匹配，取其中的文件大小字段求和
for i in $(ls -Al $1 | grep -o "^-\([^[:blank:]]*[[:blank:]]*\)\{4\}[^[:blank:]]*" |grep -o "[^[:blank:]]*$")
do
let count+=i
done
echo -e "$count"
#退出
exit 0
```

- 实验结果
- 根目录

```
bucket@ubuntu:~/lab$ ./countFile.sh /
普通文件：      1
目录文件：      22
可执行文件：    0
普通文件字节数：        1494845440
```

- 普通目录

```
bucket@ubuntu:~/lab$ ./countFile.sh /home/bucket
普通文件：      15
目录文件：      24
可执行文件：    0
普通文件字节数：        26770
bucket@ubuntu:~/lab$ ./countFile.sh /home/bucket/lab
普通文件：      4
目录文件：      1
可执行文件：    3
普通文件字节数：        1749
```

- 相对路径

```
bucket@ubuntu:~/lab$ ./countFile.sh .
普通文件：      4
目录文件：      1
可执行文件：    3
普通文件字节数：        1749
```

- 没有路径输入

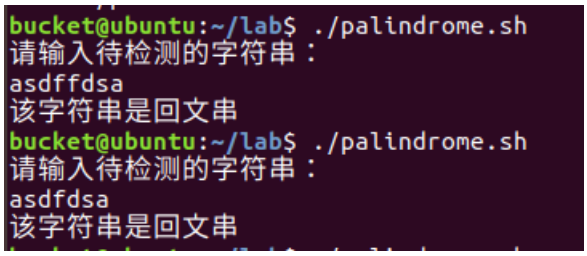
```
bucket@ubuntu:~/lab$ ./countFile.sh
Input error!
```

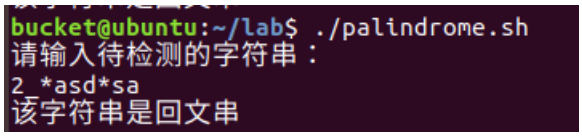
- 回文串
- 需求描述：输入一个字符串，忽略非字母后，检测该字符串是否为回文串。
- 设计思路：首先取出字符串中要忽略的字符，然后截取字符串，测试对应位置的字符是否都相同。如果不是则输出不是回文字串，是则输出是回文字串。
- 代码：

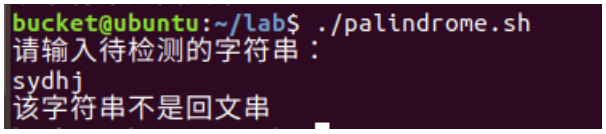
```
1. ec#!/bin/bash
2. #UTF-8
3. #**************************************************************
   **************
4. #名称：  palindrome.sh
5. #作者：  刘星烨
6. #学号：  3180105954
7. #功能：  输入一个字符串，忽略（删除）非字母后，检测该字符串是否为回文
   (palindrome)
8. #参数：  字符串
9. #**************************************************************
   **************
10.#从键盘读入字符串
11.echo "请输入待检测的字符串："
12.read input
13.#删除字符串中除字母外的部分
14.word=$(echo $input | tr -c -d [:alpha:])
```

```
#变量 len 为字符串的
declare -i len
len=${#word}
# echo $len
# echo $(($len-1))
#判断经过处理的字符串是否为回文串
for((i=0; i<`expr len/2`; i++))
do
# echo ${word:i:1}
# echo ${word:len-i-1:1}
if [ ${word:i:1} != ${word:len-i-1:1} ]
then
echo "该字符串不是回文串"
exit 0
fi
done
echo "该字符串是回文串"
exit 0
```

- ho "该字符串是回文串"
- exit 0

- 实验结果：
- 偶数/奇数回文串
- 

- 有非字母的回文串
- 

- 非回文串
- 

- 有非法字符，非字符串

```
bucket@ubuntu:~/lab$ ./palindrome.sh
请输入待检测的字符串：
a*_djw3
该字符串不是回文串
```

- 同步文件
- 需求描述：编写一个实现文件备份和同步的 shell 脚本程序 dirsync。程序的参数是两个需要备份同步的目录。
- 设计思路：共有两个功能，一个备份目录一个同步目录。对目录下的文件进行遍历，目录文件继续执行脚本，否则进行复制。在同步模式下要计较更改时间，更新文件。
- 代码：

```
1. #!/bin/bash
2. #UTF-8
3. #**********************************************************
   *****************
4. #名称：  dirsync.sh
5. #作者：  刘星烨
6. #学号：  3180105954
7. #功能：  实现文件备份和同步
8. #参数：  源目录及目标目录
9. #**********************************************************
   *****************
10.#检查输入的参数是否为三个，以及是否为目录
11.if test $# -ne 3; then
12.   echo -e "输入参数错误！"
13.   exit 1
14.fi
15.mode=$1
16.src=$2
17.dist=$3
18.#case 语句判断运行模式
19.case $mode in
20.   -b)
21.   # 进入目录
22.   cd $src
23.   # 判断目录是否非空
24.   if [ "`ls -A $src`" != "" ]
25.   then
26.# 遍历每个文件
27.for file in *
28.do
29.   # 如果是目录文件
30.   if [ -d "$file" ]; then
31.#目标目录是否存在
```

```
32.if [ ! -d "$dist/$file" ]; then
33.  cp -r $file $dist/
34.#递归调用
35.else
36.  ./dirsync.sh $mod $src/$file $dist/$file
37.fi
38.  #直接复制
39.  else
40.cp -a $file $dist/
41.  fi
42.done
43.  fi
44.  cd ..
45.  -s)
46.  find $src -print >/.src   #把/src 的所有文件的全名保存到
   /.src
47.  find $dist -print >/.dist
48.  # 进入目录
49.  cd $dist
50.  #比较两个文件里的不同
51.  for i in $(comm -23 /.dist /.src) ; do
52.  rm "$dist/$i" #删除多余的文件和计数。
53.  done  #循环到此结束。
54.  # 目录非空
55.  if [ "`ls -A $dist`" != "" ]
56.  then
57.for file in *
58.do
59.  # 如果是目录文件
60.  if [ -e "$src/$file" ]
61.  then
62.#已存在，递归调用
63.if [ -d "$file" ]
64.then
65.
66.  ./dirsync.sh $mod $src/$file $dist/$file
67.
68.else
69.  #比较更新时间
70.  if [ "$src/$file" -nt "$dist/$file" ]
71.  then
72.cp -a $src/$file $dist/
73.  else
74.cp -a $dist/$file $src/
```

```
75.   fi
76.fi
77.   fi
78.done
79.   fi
80.   cd ..
81.   ./dirsync.sh -b $src $dir2
82.
83.#其他参数则报错，没有其他模式了
84.   *) echo "Wrong mode!"
85.exit 0
```

- 实验结果：
- 准备一个目录，进行增量备份

```
bucket@ubuntu:~/lab$ ls -l
total 20
-rwxr-xr-x 1 bucket bucket 1152 Jul 31 06:09 countFile.sh
-rwxr-xr-x 1 bucket bucket 2672 Aug 21 04:55 dirsync.sh
drwxr-xr-x 2 bucket bucket 4096 Aug 14 01:36 hwmanager
-rwxr-xr-x 1 bucket bucket  486 Aug 21 04:40 palindrome.sh
-rwxr-xr-x 1 bucket bucket   27 Aug  9 20:20 test.sh
bucket@ubuntu:~/lab$ cd
bucket@ubuntu:~$ cd cpy
bucket@ubuntu:~/cpy$ ls -l
total 0
```

```
bucket@ubuntu:~/lab$ ./dirsync.sh -b /home/bucket/lab /home/bucket/cpy
bucket@ubuntu:~/lab$ cd ../cpy
bucket@ubuntu:~/cpy$ ls -l
total 20
-rwxr-xr-x 1 bucket bucket 1152 Jul 31 06:09 countFile.sh
-rwxr-xr-x 1 bucket bucket 2672 Aug 21 04:55 dirsync.sh
drwxr-xr-x 2 bucket bucket 4096 Aug 21 23:24 hwmanager
-rwxr-xr-x 1 bucket bucket  486 Aug 21 04:40 palindrome.sh
-rwxr-xr-x 1 bucket bucket   27 Aug  9 20:20 test.sh
```

- 修改一个目录下的文件，进行同步更新（此处修改 cpy 下的 test.sh 和 lab 下的 dirsync.sh）

```
bucket@ubuntu:~/lab$ ./dirsync.sh -s /home/bucket/lab /home/bucket/cpy
```

```
bucket@ubuntu:~/cpy$ ls -l
total 20
-rwxr-xr-x 1 bucket bucket 1152 Jul 31 06:09 countFile.sh
-rwxr-xr-x 1 bucket bucket 2672 Aug 21 04:55 dirsync.sh
drwxr-xr-x 2 bucket bucket 4096 Aug 21 23:24 hwmanager
-rwxr-xr-x 1 bucket bucket  486 Aug 21 04:40 palindrome.sh
-rwxr-xr-x 1 bucket bucket   40 Aug 21 23:25 test.sh
```

```
bucket@ubuntu:~/lab$ ./dirsync.sh -s /home/bucket/lab /home/bucket/cpy
bucket@ubuntu:~/lab$ ls -l
total 20
-rwxr-xr-x 1 bucket bucket 1152 Jul 31 06:09 countFile.sh
-rwxr-xr-x 1 bucket bucket 2678 Aug 21 23:30 dirsync.sh
drwxr-xr-x 2 bucket bucket 4096 Aug 14 01:36 hwmanager
-rwxr-xr-x 1 bucket bucket  486 Aug 21 04:40 palindrome.sh
-rwxr-xr-x 1 bucket bucket   40 Aug 21 23:25 test.sh
bucket@ubuntu:~/lab$ cd ../cpy
bucket@ubuntu:~/cpy$ ls -l
total 20
-rwxr-xr-x 1 bucket bucket 1152 Jul 31 06:09 countFile.sh
-rwxr-xr-x 1 bucket bucket 2678 Aug 21 23:30 dirsync.sh
drwxr-xr-x 2 bucket bucket 4096 Aug 21 23:24 hwmanager
-rwxr-xr-x 1 bucket bucket  486 Aug 21 04:40 palindrome.sh
-rwxr-xr-x 1 bucket bucket   40 Aug 21 23:25 test.sh
```

- 修改目录下的文件，同步

```
bucket@ubuntu:~/cpy/hwmanager$ gedit sqltest.sh
bucket@ubuntu:~/cpy/hwmanager$ cd ..
bucket@ubuntu:~/cpy$ ls -l
total 20
-rwxr-xr-x 1 bucket bucket 1152 Jul 31 06:09 countFile.sh
-rwxr-xr-x 1 bucket bucket 2678 Aug 21 23:30 dirsync.sh
drwxr-xr-x 2 bucket bucket 4096 Aug 21 23:47 hwmanager
-rwxr-xr-x 1 bucket bucket  486 Aug 21 04:40 palindrome.sh
-rwxr-xr-x 1 bucket bucket   40 Aug 21 23:25 test.sh
```

-

```
bucket@ubuntu:~/cpy$  ./dirsync.sh -s /home/bucket/lab /home/bucket/cpy
```

-

```
bucket@ubuntu:~/lab/hwmanager$ ls -l
total 4
-rw-r--r-- 1 bucket bucket 15 Aug 21 23:47 sqltest.sh
```

- 作业管理系统
- 需求描述：

• 数据：本作业管理系统的数据分为两大模块，分别是用户和课程。用户数据包括用户类型（学生、教师、 管理员），用户账号（工号或学号），用户密码。课程数据包括课程号，课程信息，课程任。两大模块之间的联系是教师的授课信息和学生的选课信息。

• 功能：本系统的所有操作均有菜单提示，并通过输入选项来选择相应的操作。启动系统时，用户选择登录模式（学生、教师、管理员）进行登录验证。 管理员模式下，用户可对教师信息和课程信息进行管理。管理教师信息的功能包括教师信息（工号和姓名）的增删改查，以及将教师与课程绑定或解绑。管理课程信息的功能包括课程信息的增删改查。此外，管理员可对用户的密码进行修改。

教师模式下，用户可选择自己所教的课程进行管理，包括对选课学生、课程信息、课程任务的管理。管理选课学生的功能包括选课学生的增删改查，如果学生尚未存在于系统，则添加进系统。教师可对系统中的所有学生信息进行修改，无论其是否选课。管理课程信息的功能包括课程信息的增删改查。管理课程任务的功能包括发布任务、删除任务、修改任务、查询任务和查看所有学生的完成情况。此外，教师可修改本人的密码。

学生模式下，用户可选择自己所修读的课程进行管理，包括查看课程信息和任务管理。作业管理的功能包括查看任务，上传作业（因使用 mysql 数据库的限制，只能完成了提交标记，无法真实上交作业）。此外，学生可修改本人的密码。

- 设计文档：

  设计思想 本作业管理系统全部采用 shell 程序设计。除此以外，还使用了 mysql 数据库作为辅助。使用数据库使数据组织更加直观方便，在主系统中每层都显示菜单，并且执行完成后返回上层函数。

  数据库结构：

  student(sid, sname, spassword);

  teacher(tid, tname, tpassword);

  course(cid, cname, tid);

  bindcourse(tid, cid);

  submit(sid, cid, lid, ac);

  lab(cid, lid, content);

  choose_course(sid, cid);

- 功能模块

  程序根据用户登录模式的不同，分为三大功能模块（学生、教师、管理员），并有一顶层模块用于 调用这三大功能模块中的一个。 系统运行时，首先执行 main.sh 脚本，选择登录模式后，程序通过 source 命令 (.) 将 module.sh 文 件包含进来。模块的实现文件中只有该模块所用的若干函数。全部包含后，调用该模块的 main 函 数。 管理员模块中，程序实现了管理员登录，教师用户的增删改查，课程的增删改查，教师与课程的 绑定和解绑，以及修改任意用户的密码。 教师模块中，程序实现了教师登录，选课学生的增删改查，课程信息的增删改查，课程任务的增删改查，以及修改本人密码。 学生模块中，程序实现了学生登录，查看课程信息，上传和下载作业，以及修改本人密码。

- 实现方法

  在主函数中选择模式并进入相应的界面。进入管理员模式只有一个账号，直接使用密码文件进行对比。

  此后主要的数据交互使用 mysql，在每种用户界面内先列出菜单，选择功能后进入相应函数执行，然后再返回菜单。

- 不足：因使用 mysql，实际上忽略了 linux 系统对于文件处理的强大功能，导致作业管理系统未能真正实现文件的提交。如果使用文本方式处理学生信息和作业情况就可以解决关于文件上传下载的问题。

- 代码：

```
1. #!/bin/bash
2. #UTF-8
3. #**************************************************
   ****************
```

```
#名称：  main.sh
#作者：  刘星烨
#学号：  3180105954
#功能：  显示主菜单
#参数：  无
#*************************************************************
10.
#显示菜单
echo "*******************************"
echo "欢迎来到作业管理系统！"
echo "输入身份：1 \t 管理员 \t 2 \t 教师 \t 3 \t 学生 \t 0 \t 退出系统"
echo "*******************************"
read usermode
#判断用户模式
case $usermode in
1)
    echo "进入管理员模式"
    . ./admin.sh
    main_admin
    ;;
2)
    echo "进入教师模式"
    . ./teacher.sh
    main_teacher
    ;;
3)
    echo "进入学生模式"
    . ./student.sh
    main_student
    ;;
0) echo "退出系统！"
    sleep 2
    while true; do
        exit
    done
    ;;
*) echo "输入错误！"
    sleep 2
    exit 2
    ;;
esac
45.
```

```
46.exit 1
```

```
#!/bin/bash
#UTF-8
#*************************************************************
****************
#名称:  admin.sh
#作者:  刘星烨
#学号:  3180105954
#功能:  执行管理员功能
#参数:  无
#*************************************************************
****************

#创建教师账号

create_teacher()
{
    #显示提示信息
    echo -e "输入 0 退出此功能，其他输入创建教师"
    #判断是否回到上级菜单
    read state
    if test $state -eq 0; then
        echo -e "回到菜单"
        sleep 2
        admin_function
        exit 1
    fi
    #判断教师工号是否存在
    echo -e "输入教师工号："
    read teacher_id
    #执行 sql 语句，查找工号是否已经存在
    MYSQL="mysql -ubucket -p123456 -Dmanage --default-chara
cter-set=utf8 -A -N"
    sql="select * from teacher where tid='$teacher_id'"
    result="$($MYSQL -e "$sql")"
    #如果工号不存在，则插入
    #否则不进行插入
    if [ ! -n "$result" ]; then
        echo -e "输入教师姓名："
        read teacher_name
        #设置初始密码为 111111
        password=111111
```

```
39.         sql="insert into teacher values('$teacher_id, '$tea
    cher_name,'$password')"
40.         $MYSQL -e "$sql"
41.         #检查sql 语句是否正常执行
42.         if [[ $? -eq 0 ]]
43.         then
44.             echo -e "\033[32m 执行插入成功! \033[0m"
45.         else
46.             echo -e "\033[31m 执行插入失败! \033[0m"
47.             #执行sql 语句异常提示
48.             exit -1
49.         fi
50.     else
51.         echo -e "工号已存在! "
52.     fi
53.     #回到admin 主菜单
54.     admin_function
55.     #未知错误
56.     exit 1
57.
58.}
59.
60.#删除教师账号
61.
62.delete_teacher()
63.{
64.     #显示提示信息
65.     echo -e "输入0退出此功能，其他输入删除教师"
66.     #判断是否回到上级菜单
67.     read state
68.     if test $state -eq 0; then
69.         echo -e "回到菜单"
70.         sleep 2
71.         admin_function
72.         #未知错误信息
73.         exit 1
74.     fi
75.     #判断教师工号是否存在
76.     echo -e "输入教师工号: "
77.     read teacher_id
78.     #执行sql 语句，查找工号是否已经存在
79.     MYSQL="mysql -ubucket -p123456 -Dmanage --default-chara
    cter-set=utf8 -A -N"
80.     sql="select * from teacher where tid='$teacher_id'"
```

```
    result="$($MYSQL -e "$sql")"
    #如果工号存在，则删除
    #否则不进行删除
    if [ ! -n "$result" ]; then
        echo -e "工号不存在！"
    else
        sql="delete from teacher where tid='$teacher_id'"

        $MYSQL -e "$sql"
        #检查 sql 语句是否正常执行
        if [[ $? -eq 0 ]]
        then
            echo -e "\033[32m 执行删除成功！\033[0m"
        else
            echo -e "\033[31m 执行删除失败！\033[0m"
            #执行 sql 语句异常提示
            exit -1
        fi
    fi
    #回到 admin 主菜单
      admin_function
      exit 1
}

#修改教师信息

edit_teacher()
{
    #显示提示信息
    echo -e "输入 0 退出此功能，其他输入编辑教师信息"
    #判断是否回到上级菜单
    read state
    if test $state -eq 0; then
        echo -e "回到菜单"
        sleep 2
        admin_function
        #未知错误信息
        exit 1
    fi
    #判断教师工号是否存在
    echo -e "输入教师工号："
    read teacher_id
    #执行 sql 语句，查找工号是否已经存在
```

```
    MYSQL="mysql -ubucket -p123456 -Dmanage --default-character-set=utf8 -A -N"
    sql="select * from teacher where tid='$teacher_id'"
    result="$($MYSQL -e "$sql")"
     #如果工号存在，则进行修改
    #否则不进行修改
    if [ ! -n "$result" ]; then
        echo -e "工号不存在！"
    else
        echo -e "1\t 修改教师姓名\t2\t 修改教师密码"
        read edit_mode
        if test $edit_mode -eq 1; then
            echo -e "输入修改后姓名："
            read edit_name
            sql="update teacher set tname='$edit_name' where tid='$teacher_id'"
            $MYSQL -e "$sql"
            #检查 sql 语句是否正常执行
            if [[ $? -eq 0 ]]
            then
                echo -e "\033[32m 执行修改成功！\033[0m"
            else
                echo -e "\033[31m 执行修改失败！\033[0m"
                #执行 sql 语句异常提示
                exit -1
            fi
        else
            echo -e "输入修改后密码："
            read edit_password
            sql="update teacher set tpassword='$edit_password' where tid='$teacher_id'"
            $MYSQL -e "$sql"
            #检查 sql 语句是否正常执行
            if [[ $? -eq 0 ]]
            then
                echo -e "\033[32m 执行修改成功！\033[0m"
            else
                echo -e "\033[31m 执行修改失败！\033[0m"
                #执行 sql 语句异常提示
                exit -1
            fi
        fi
    fi
    #回到 admin 主菜单
```

```
164.     admin_function
165.     exit 1
166. }
167.
168. #按id查询教师账号
169. display_by_id()
170. {
171.     echo -e "输入要查询的教师工号："
172.     read teacher_id
173.     MYSQL="mysql -ubucket -p123456 -Dmanage --default-cha
     racter-set=utf8 -A -N"
174.     sql="select * from manager.teacher where tid='$teache
     r_id'"
175.     result="$($MYSQL -e "$sql")"
176.     if [[ $? -eq 0 ]]
177.     then
178.         echo -e "\033[32m 查询成功！\033[0m"
179.         echo ${result[*]}
180.     else
181.         echo -e "\033[31m 查询失败！\033[0m"
182.         #执行sql语句异常提示
183.         exit -1
184.     fi
185.     display_teacher
186.     exit 1
187. }
188.
189. #按姓名查询教师账号
190.
191. display_by_name()
192. {
193.     echo -e "输入要查询的教师姓名："
194.     read teacher_name
195.     MYSQL="mysql -ubucket -p123456 -Dmanage --default-cha
     racter-set=utf8 -A -N"
196.     sql="select * from manager.teacher where tname='$teac
     her_name'"
197.     result="$($MYSQL -e "$sql")"
198.     if [[ $? -eq 0 ]]
199.     then
200.         echo -e "\033[32m 查询成功！\033[0m"
201.         echo ${result[*]}
202.     else
203.         echo -e "\033[31m 查询失败！\033[0m"
```

```
204.         #执行sql 语句异常提示
205.         exit -1
206.     fi
207.     display_teacher
208.     exit 1
209. }
210.
211. #显示所有教师账号
212.
213. display_all()
214. {
215.     echo -e "显示所有教师信息："
216.     MYSQL="mysql -ubucket -p123456 -Dmanage --default-cha
   racter-set=utf8 -A -N"
217.     sql="select * from manager.teacher"
218.     result="$($MYSQL -e "$sql")"
219.     if [[ $? -eq 0 ]]
220.     then
221.         echo -e "\033[32m 查询成功！\033[0m"
222.         echo ${result[*]}
223.     else
224.         echo -e "\033[31m 查询失败！\033[0m"
225.         #执行sql 语句异常提示
226.         exit -1
227.     fi
228.     display_teacher
229.     exit 1
230.
231. }
232.
233. #显示教师账号
234.
235. display_teacher()
236. {
237.     #显示提示信息
238.     echo -e "输入 0 退出此功能，其他输入显示教师信息"
239.     #判断是否回到上级菜单
240.     read state
241.     if test $state -eq 0; then
242.         echo -e "回到菜单"
243.         sleep 2
244.         admin_function
245.         #未知错误信息
246.         exit 1
```

```
247.     fi
248.     echo -e "选择查询方式："
249.     echo -e "1\t 输入工号查询"
250.     echo -e "2\t 输入姓名查询"
251.     echo -e "3\t 显示所有教师"
252.     echo -e "4\t 返回上层"
253.     echo -e "0\t 退出"
254.     read select_mode
255.     case $select_mode in
256.         1) display_by_id;;
257.         2) display_by_name;;
258.         3) display_all;;
259.         4) admin_function
260.             exit 1;;
261.         0) echo -e "退出系统！"
262.             sleep 2
263.             exit 0
264.             ;;
265.         *) echo -e "非法输入！"
266.             display_teacher;;
267.     esac
268.     exit 1
269. 
270. }
271. 
272. #创建新的课程
273. 
274. create_course()
275. {
276.     #显示提示信息
277.     echo -e "输入 0 退出此功能，其他输入创建课程"
278.     #判断是否回到上级菜单
279.     read state
280.     if test $state -eq 0; then
281.         echo -e "回到菜单"
282.         sleep 2
283.         admin_function
284.         #未知错误
285.         exit 1
286.     fi
287.     #判断课程代号是否存在
288.     echo -e "输入课程代号："
289.     read course_id
290.     #执行 sql 语句，查找工号是否已经存在
```

```
291.     MYSQL="mysql -ubucket -p123456 -Dmanage --default-cha
   racter-set=utf8 -A -N"
292.     sql="select * from teacher where cid='$course_id'"
293.     result="$($MYSQL -e "$sql")"
294.     #如果工号不存在，则插入
295.     #否则不进行插入
296.     if [ ! -n "$result" ]; then
297.         echo -e "输入课程名称："
298.         read course_name
299.         sql="insert into course values('$course_id', '$co
   urse_name')"
300.         $MYSQL -e "$sql"
301.         #检查sql 语句是否正常执行
302.         if [[ $? -eq 0 ]]
303.         then
304.             echo -e "\033[32m 执行插入成功！\033[0m"
305.         else
306.             echo -e "\033[31m 执行插入失败！\033[0m"
307.             #执行sql 语句异常提示
308.             exit -1
309.         fi
310.     else
311.         echo -e "课程代号已存在！"
312.     fi
313.     #回到admin 主菜单
314.     admin_function
315.     #未知错误
316.     exit 1
317. }
318.
319. #删除课程
320. delete_course()
321. {
322.     #显示提示信息
323.     echo -e "输入 0 退出此功能，其他输入删除教师"
324.     #判断是否回到上级菜单
325.     read state
326.     if test $state -eq 0; then
327.         echo -e "回到菜单"
328.         sleep 2
329.         admin_function
330.         #未知错误信息
331.         exit 1
332.     fi
```

```
333.     #判断课程代号是否存在
334.     echo -e "输入课程代号：  "
335.     read course_id
336.     #执行 sql 语句，查找工号是否已经存在
337.     MYSQL="mysql -ubucket -p123456 -Dmanage --default-cha
   racter-set=utf8 -A -N"
338.     sql="select * from course where cid='$course_id'"
339.     result="$($MYSQL -e "$sql")"
340.     #如果代号存在，则删除
341.     #否则不进行删除
342.     if [ ! -n "$result" ]; then
343.         echo -e "代号不存在！ "
344.     else
345.         sql="delete from course where cid=$course_id"

346.         $MYSQL -e "$sql"
347.         #检查 sql 语句是否正常执行
348.         if [[ $? -eq 0 ]]
349.         then
350.             echo -e "\033[32m 执行删除成功！ \033[0m"
351.         else
352.             echo -e "\033[31m 执行删除失败！ \033[0m"
353.             #执行 sql 语句异常提示
354.             exit -1
355.         fi
356.     fi
357.     #回到 admin 主菜单
358.     admin_function
359.     exit 1
360. }
361.
362. #修改课程
363. edit_course()
364. {
365.     #显示提示信息
366.     echo -e "输入 0 退出此功能，其他输入编辑课程信息"
367.     #判断是否回到上级菜单
368.     read state
369.     if test $state -eq 0; then
370.         echo -e "回到菜单"
371.         sleep 2
372.         admin_function
373.         #未知错误信息
374.         exit 1
```

```
375.     fi
376.     #判断课程代号是否存在
377.     echo -e "输入课程代号："
378.     read course_id
379.     #执行 sql 语句，查找工号是否已经存在
380.     MYSQL="mysql -ubucket -p123456 -Dmanage --default-cha
   racter-set=utf8 -A -N"
381.     sql="select * from course where cid='$course_id'"
382.     result="$($MYSQL -e "$sql")"
383.      #如果工号存在，则进行修改
384.     #否则不进行修改
385.     if [ ! -n "$result" ]; then
386.         echo -e "代号不存在！"
387.     else
388.         echo -e "输入修改后名称："
389.         read edit_name
390.         sql="update course set cname=$edit_name where cid
   =$course_id"
391.         $MYSQL -e "$sql"
392.         #检查 sql 语句是否正常执行
393.         if [[ $? -eq 0 ]]
394.         then
395.             echo -e "\033[32m 执行修改成功！\033[0m"
396.         else
397.             echo -e "\033[31m 执行修改失败！\033[0m"
398.             #执行 sql 语句异常提示
399.             exit -1
400.         fi
401.     fi
402.     #回到 admin 主菜单
403.     admin_function
404.     exit 1
405. }
406.
407. #绑定课程和教师信息
408.
409. bind_course()
410. {
411.      #显示提示信息
412.     echo -e "输入 0 退出此功能，其他输入绑定课程"
413.     #判断是否回到上级菜单
414.     read state
415.     if test $state -eq 0; then
416.         echo -e "回到菜单"
```

```
417.          sleep 2
418.          admin_function
419.          #未知错误信息
420.          exit 1
421.      fi
422.      echo -e "输入课程代号："
423.      read course_id
424.      #执行 sql 语句，查找工号是否已经存在
425.      MYSQL="mysql -ubucket -p123456 -Dmanage --default-cha
   racter-set=utf8 -A -N"
426.      sql="select * from course where cid='$course_id'"
427.      result="$($MYSQL -e "$sql")"
428.      if [ ! -n "$result" ]; then
429.          echo -e "代号不存在！"
430.          admin_function
431.          exit 1
432.      fi
433.      echo -e "输入教师工号："
434.      read teacher_id
435.      #执行 sql 语句，查找工号是否已经存在
436.      sql="select * from teacher where tid='$teacher_id'"
437.      result="$($MYSQL -e "$sql")"
438.      if [ ! -n "$result" ]; then
439.          echo -e "工号不存在！"
440.          admin_function
441.          exit 1
442.      fi
443.      #查询绑定关系是否已经存在
444.      sql="select * from bindcourse where tid='$teacher_id'
    and cid='$course_id'"
445.      result="$($MYSQL -e "$sql")"
446.      if [ ! -n "$result" ]; then
447.          sql="insert into bindcourse values('$teacher_id',
   '$course_id')"
448.          $MYSQL -e "$sql"
449.          if [[ $? -eq 0 ]]
450.          then
451.              echo -e "\033[32m 绑定成功！\033[0m"
452.          else
453.              echo -e "\033[31m 绑定失败！\033[0m"
454.              #执行 sql 语句异常提示
455.              exit -1
456.          fi
457.      else
```

```
458.          echo -e "绑定关系已存在!"
459.          admin_function
460.          exit 1
461.      fi
462.      admin_function
463.      exit 1
464.
465. }
466.
467. bind_cancel()
468. {
469.      #显示提示信息
470.      echo -e "输入 0 退出此功能，其他输入绑定课程"
471.      #判断是否回到上级菜单
472.      read state
473.      if test $state -eq 0; then
474.          echo -e "回到菜单"
475.          sleep 2
476.          admin_function
477.          #未知错误信息
478.          exit 1
479.      fi
480.      echo -e "输入课程代号: "
481.      read course_id
482.      #执行 sql 语句，查找工号是否已经存在
483.      MYSQL="mysql -ubucket -p123456 -Dmanage --default-cha
   racter-set=utf8 -A -N"
484.      sql="select * from course where cid='$course_id'"
485.      result="$($MYSQL -e "$sql")"
486.      if [ ! -n "$result" ]; then
487.          echo -e "代号不存在! "
488.          admin_function
489.          exit 1
490.      fi
491.      echo -e "输入教师工号: "
492.      read teacher_id
493.      #执行 sql 语句，查找工号是否已经存在
494.      sql="select * from teacher where tid='$teacher_id'"
495.      result="$($MYSQL -e "$sql")"
496.      if [ ! -n "$result" ]; then
497.          echo -e "工号不存在! "
498.          admin_function
499.          exit 1
500.      fi
```

```
501.     #查询绑定关系是否已经存在
502.     sql="select * from bindcourse where tid='$teacher_id'
    and cid='$course_id'"
503.     result="$($MYSQL -e "$sql")"
504.     if [ ! -n "$result" ]; then
505.         #绑定关系不存在，返回上级菜单
506.         echo -e "绑定关系已存在!"
507.         admin_function
508.         exit 1
509.
510.     else
511.         sql="delete from bindcourse where tid='$teacher_i
   d' and cid=$course_id"
512.         $MYSQL -e "$sql"
513.         if [[ $? -eq 0 ]]
514.         then
515.             echo -e "\033[32m 取消绑定成功! \033[0m"
516.         else
517.             echo -e "\033[31m 取消绑定失败! \033[0m"
518.             #执行 sql 语句异常提示
519.             exit -1
520.         fi
521.     fi
522.     admin_function
523.     exit 1
524.
525. }
526.
527. #选择 admin 功能函数
528.
529.
530. admin_function()
531. {
532.     #显示功能菜单
533.     echo -e ""
534.     echo -e "选择以下操作: "
535.     echo -e "1\t 创建教师账号"
536.     echo -e "2\t 删除教师账号"
537.     echo -e "3\t 修改教师账号"
538.     echo -e "4\t 显示教师账号"
539.     echo -e "5\t 创建课程"
540.     echo -e "6\t 删除课程"
541.     echo -e "7\t 修改课程"
542.     echo -e "8\t 绑定课程"
```

```
543.     echo -e "9\t 取消课程绑定"
544.     echo -e "0\t 退出系统"
545.     read input
546.     #模式选择
547.     case $input in
548.         1) create_teacher;;
549.         2) delete_teacher;;
550.         3) edit_teacher;;
551.         4) display_teacher;;
552.         5) create_course;;
553.         6) delete_course;;
554.         7) edit_course;;
555.         8) bind_course;;
556.         9) bind_cancel;;
557.         0) echo -e "退出系统！"
558.             sleep 2
559.             exit 0;;
560.         *) echo -e "输入错误！"
561.             admin_function
562.             exit 1;;
563.     esac
564.     #未知错误
565.     exit 1
566. }
567.
568. #用户验证
569.
570.
571. main_admin()
572. {
573.     stty -echo
574.     #从文件里读取管理员密码
575.     password=$( cat ./file/admin_passwd )
576.     #exit_flag 作为退出输入密码flag
577.     declare -i exit_flag=0
578.     declare -i count_input=3
579.     #判断密码输入是否正确，错误三次退出
580.     while [ $exit_flag -eq 0 ]
581.     do
582.     #输入密码
583.        echo -e "请输入管理员密码："
584.        read input
585.        #为了避免直接看到文件中的密码，实行加密
586.        try = $(echo -e "$input" | md5)
```

```
587.         #对比字符串，对比成功即为通过
588.         if test $password = $try; then
589.              stty echo
590.              let exit_flag++
591.              break
592.          else
593.              let count_input = count_input-1
594.              echo -e "密码错误！请重新输入！你还有
   $count_input 次机会"
595.
596.          fi
597.          #输入次数用完
598.          if test count_input -eq 0; then
599.              stty echo
600.              exit 3
601.          fi
602.      done
603.      echo -e ""
604.      echo  "**********欢迎进入管理员系统**********"
605.      #进入功能界面
606.      admin_function
607.      exit 1
608. }
```

```
1. #!/bin/bash
2. #UTF-8
3. #****************************************************************
   **************
4. #名称：  admin.sh
5. #作者：  刘星烨
6. #学号：  3180105954
7. #功能：  执行管理员功能
8. #参数：  无
9. #****************************************************************
   **************
10.
11.display_student()
12.{
13.     #显示提示信息
14.     echo -e "输入 0 退出此功能，其他输入进行查询"
15.     #判断是否回到上级菜单
16.     read state
```

```
    if test $state -eq 0; then
        echo -e "回到菜单"
        sleep 2
        teacher_function
        exit 1
    fi
    #显示查询菜单
    echo -e "选择查询方式："
    echo -e "1\t 查询学生选课情况"
    echo -e "2\t 查询课程所有学生"
    echo -e "0\t 退出系统"
    read display_mode
    #退出系统
    if test $display_mode -eq 0; then
        echo -e "退出系统！"
        sleep 2
        exit 0
    fi
    #判断模式，执行查询
    if test $display_mode -eq 1; then
        echo -e "输入学生学号："
        #检查学生是否存在
        read student_id
        MYSQL="mysql -ubucket -p123456 -Dmanage --default-cha
racter-set=utf8 -A -N"
        sql="select * from student where sid=$student_id"
        result="$($MYSQL -e "$sql")"
        #学号不存在则退出
        #学号存在则进行查询
        if [ ! -n "$result" ]; then
            echo -e "学号不存在！"
            teacher_function
            exit 1
        fi
        #按学号进行查询
        sql="select * from choose_course where sid=$student_i
d"
        result="$($MYSQL -e "$sql")"
        if [[ $? -eq 0 ]]
        then
            echo -e "\033[32m 查询成功！\033[0m"
            echo ${result[*]}
        else
            echo =e "\033[31m 查询失败！\033[0m"
```

```
            manage_student
            #执行sql语句异常提示
            exit -1
        fi
    else
        echo -e "输入课程代号："
        #检查课程是否存在
        read course_id
        MYSQL="mysql -ubucket -p123456 -Dmanage --default-character-set=utf8 -A -N"
        sql="select * from student where sid=$student_id"
        result="$($MYSQL -e "$sql")"
        #代号不存在则退出
        #代号存在则进行查询
        if [ ! -n "$result" ]; then
            echo -e "代号不存在！"
            teacher_function
            exit 1
        fi
        #按代号进行查询
        sql="select * from choose_course where cid=$course_id"
        result="$($MYSQL -e "$sql")"
        if [[ $? -eq 0 ]]
        then
            echo -e "\033[32m 查询成功！\033[0m"
            echo ${result[*]}
        else
            echo =e "\033[31m 查询失败！\033[0m"
            manage_student
            #执行sql语句异常提示
            exit -1
        fi
    fi
    teacher_function
    exit 1
}

#添加选课学生

create_student()
{
    #显示提示信息
        echo -e "输入0退出此功能，其他输入创建教师"
```

```
        #判断是否回到上级菜单
        read state
        if test $state -eq 0; then
            echo -e "回到菜单"
            sleep 2
            teacher_function
            exit 1
        fi
        echo -e "输入学生学号："
        #检查学生是否存在
        read student_id
        MYSQL="mysql -ubucket -p123456 -Dmanage --default-character-set=utf8 -A -N"
        sql="select * from student where sid=$student_id"
        result="$($MYSQL -e "$sql")"
        #学号不存在则创建
        #学号存在则进行查询
        if [ ! -n "$result" ]; then
            echo -e "学号$student_id 不存在！是否新建？【Y/N】"
            read input
            if test $input="Y"; then
                echo -e "输入学生姓名："
                read student_name
                password=111111
                sql="insert into student values('$student_id', '$student_name', '$password')"
                $MYSQL -e "$sql"
                if [[ $? -eq 0 ]]; then
                echo -e "\033[32m 新建成功！\033[0m"
                else
                echo =e "\033[31m 新建失败！\033[0m"
                create_student
                #执行 sql 语句异常提示
                exit -1
                fi
            else
                manage_student
                exit 1
            fi
        fi
        #判断课程代号是否存在
        echo -e "输入课程代号："
        read course_id
        #执行 sql 语句，查找代号是否已经存在
```

```
143.
144.        sql="select * from course where cid=$course_id"
145.        result="$($MYSQL -e "$sql")"
146.        #代号不存在则退出
147.        #代号存在则进行查询
148.        if [ ! -n "$result" ]; then
149.            echo -e "代号不存在! "
150.            teacher_function
151.            exit 1
152.        fi
153.
154.        #查看学生是否选课
155.        sql="select * from choose_course where sid=$student_i
    d and cid=$course_id"
156.        result="$($MYSQL -e "$sql")"
157.        if [ ! -n "$result" ]; then
158.            #建立学生与课程联系
159.            sql="insert into choose_course values('$student_i
    d', '$course_id')"
160.            $MYSQL -e "$sql"
161.            #判断 sql 语句是否成功执行
162.            if [[ $? -eq 0 ]]
163.            then
164.                echo -e "\033[32m 插入成功! \033[0m"
165.            else
166.                echo =e "\033[31m 插入失败! \033[0m"
167.                manage_student
168.                #执行 sql 语句异常提示
169.                exit -1
170.            fi
171.        else
172.            echo -e "学生已选课! "
173.        fi
174.
175.        #回到 admin 主菜单
176.        teacher_function
177.        #未知错误
178.        exit 1
179.    }
180.
181.    #删除选课学生
182.
183.    delete_student()
184.    {
```

```
185.         #显示提示信息
186.         echo -e "输入 0 退出此功能，其他输入删除教师"
187.         #判断是否回到上级菜单
188.         read state
189.         if test $state -eq 0; then
190.             echo -e "回到菜单"
191.             sleep 2
192.             teacher_function
193.             #未知错误信息
194.             exit 1
195.         fi
196.         #判断课程代号是否存在
197.         echo -e "输入课程代号："
198.         read course_id
199.         #执行 sql 语句，查找工号是否已经存在
200.         MYSQL="mysql -ubucket -p123456 -Dmanage --default-cha
    racter-set=utf8 -A -N"
201.         sql="select * from course where cid=$course_id"
202.         result="$($MYSQL -e "$sql")"
203.         #代号不存在则退出
204.         if [ ! -n "$result" ]; then
205.             echo -e "代号不存在！"
206.             teacher_function
207.             exit 1
208.         fi
209.         echo -e "输入学生学号："
210.         #检查学生是否存在
211.         read student_id
212.         sql="select * from student where sid=$student_id"
213.         result="$($MYSQL -e "$sql")"
214.         #学号不存在则退出
215.         if [ ! -n "$result" ]; then
216.             echo -e "学号不存在！"
217.
218.             teacher_function
219.             exit 1
220.         fi
221.         #查看学生是否选课
222.         sql="select * from choose_course where sid=$student_i
    d and cid=$course_id"
223.         result="$($MYSQL -e "$sql")"
224.         if [ ! -n "$result" ]; then
225.             echo -e "学生未选课！"
226.         else
```

```
227.            sql="delete from choose_course where cid=$course_
    id and tid='$teacher_id'"
228.            $MYSQL -e "$sql"
229.            #检查sql 语句是否正常执行
230.            if [[ $? -eq 0 ]]
231.            then
232.                echo -e "\033[32m 执行删除成功! \033[0m"
233.            else
234.                echo -e "\033[31m 执行删除失败! \033[0m"
235.                manage_student
236.                #执行sql 语句异常提示
237.                exit -1
238.            fi
239.        fi
240.        #回到teacher 主菜单
241.        teacher_function
242.        exit 1
243.    }
244.
245.    edit_student()
246.    {
247.        #显示提示信息
248.        echo -e "输入 0 退出此功能，其他输入删除教师"
249.        #判断是否回到上级菜单
250.        read state
251.        if test $state -eq 0; then
252.            echo -e "回到菜单"
253.            sleep 2
254.            teacher_function
255.            #未知错误信息
256.            exit 1
257.        fi
258.        echo -e "输入学生学号: "
259.        #检查学生是否存在
260.        read student_id
261.        MYSQL="mysql -ubucket -p123456 -Dmanage --default-cha
    racter-set=utf8 -A -N"
262.        sql="select * from student where sid=$student_id"
263.        result="$($MYSQL -e "$sql")"
264.        #学号不存在则退出
265.        if [ ! -n "$result" ]; then
266.            echo -e "学号不存在! "
267.            teacher_function
268.            exit 1
```

```
        fi
        echo -e "请输入修改后的学生姓名："
        #更改学生信息
        read edit_name
        sql="update student set sname='$edit_name'"
        $MYSQL -e "$sql"
        #判断 sql 语句是否成功执行
        if [[ $? -eq 0 ]]
        then
            echo -e "\033[32m 修改成功！\033[0m"
        else
            echo -e "\033[31m 修改失败！\033[0m"
            manage_student
            #执行 sql 语句异常提示
            exit -1
        fi

    }

    #管理选课名单
    manage_student()
    {
        #显示管理选课菜单
        echo -e ""
        echo -e "请选择操作："
        echo -e "1\t 查看选课学生"
        echo -e "2\t 添加选课学生"
        echo -e "3\t 删除选课学生"
        echo -e "4\t 修改学生信息"
        echo -e "5\t 返回上级菜单"
        echo -e "0\t 退出系统"
        #读取输入模式
        read manage_mode
        #进入功能函数
        case $manage_mode in
            1) display_student;;
            2) create_student;;
            3) delete_student;;
            4) edit_student;;
            5) teacher_function
                exit -1
                ;;

            *) echo -e "输入错误！"
```

```
313.                #返回菜单
314.                manage_student
315.                exit 1;;
316.        esac
317.        exit 0
318.    }
319.
320.    #创建课程信息
321.
322.    create_course()
323.    {
324.         #显示提示信息
325.        echo -e "输入 0 退出此功能，其他输入创建教师"
326.        #判断是否回到上级菜单
327.        read state
328.        if test $state -eq 0; then
329.            echo -e "回到菜单"
330.            sleep 2
331.            teacher_function
332.            exit 1
333.        fi
334.        #判断课程代号是否存在
335.        echo -e "输入课程代号："
336.        read course_id
337.        #执行 sql 语句，查找代号是否已经存在
338.        MYSQL="mysql -ubucket -p123456 -Dmanage --default-cha
    racter-set=utf8 -A -N"
339.        sql="select * from course where cid=$course_id"
340.        result="$($MYSQL -e "$sql")"
341.        #代号不存在则新建
342.        #代号存在则退出
343.        if [ ! -n "$result" ]; then
344.            echo -e "请输入课程名称："
345.            read course_name
346.            sql="insert into course values('$course_id', '$co
    urse_name', '$teacher_id')"
347.            $MYSQL -e "$sql"
348.            if [[ $? -eq 0 ]]
349.            then
350.                echo -e "\033[32m 新建成功！\033[0m"
351.            else
352.                echo =e "\033[31m 新建失败！\033[0m"
353.                manage_student
354.                #执行 sql 语句异常提示
```

```
355.                exit -1
356.            fi
357.        else
358.            echo -e "代号已存在！"
359.            manage_student
360.            exit 1
361.        fi
362.
363.        #回到teacher主菜单
364.        teacher_function
365.        #未知错误
366.        exit 1
367.    }
368.
369.
370.    #显示所有课程信息
371.
372.    display_course()
373.    {
374.        echo -e "所有课程信息："
375.        MYSQL="mysql -ubucket -p123456 -Dmanage --default-cha
    racter-set=utf8 -A -N"
376.        sql="select * from course where tid='$teacher_id'"
377.        result="$($MYSQL -e "$sql")"
378.        if [ ! -n "$result" ]; then
379.            echo -e "课程不存在！"
380.            manage_course
381.            exit 1
382.        else
383.            echo ${result[*]}
384.        fi
385.        manage_course
386.
387.        exit 1
388.
389.    }
390.
391.    #编辑课程信息
392.    edit_course()
393.    {
394.        display_course
395.        echo -e "请输入要编辑的课程代号："
396.        #判断课程代号是否存在
397.        read course_id
```

```
398.        #执行 sql 语句，查找代号是否已经存在
399.        MYSQL="mysql -ubucket -p123456 -Dmanage --default-cha
    racter-set=utf8 -A -N"
400.        sql="select * from course where cid=$course_id"
401.        result="$($MYSQL -e "$sql")"
402.        if [ ! -n "$result" ]; then
403.            echo -e "课程不存在！"
404.            manage_course
405.            exit 1
406.        else
407.            echo -e "输入修改后的课程名称："
408.            read edit_name
409.            sql="update course set cname='$edit_name'"
410.            $MYSQL -e "$sql"
411.            if [[ $? -eq 0 ]]
412.            then
413.                echo -e "\033[32m 修改成功！\033[0m"
414.            else
415.                echo =e "\033[31m 修改失败！\033[0m"
416.                manage_student
417.                #执行 sql 语句异常提示
418.                exit -1
419.            fi
420.
421.        fi
422.        manage_course
423.        exit 1
424.    }
425.
426.    delete_course()
427.    {
428.        #显示提示信息
429.        echo -e "输入 0 退出此功能，其他输入删除课程"
430.        #判断是否回到上级菜单
431.        read state
432.        if test $state -eq 0; then
433.            echo -e "回到菜单"
434.            sleep 2
435.            teacher_function
436.            #未知错误信息
437.            exit 1
438.        fi
439.        #判断课程代号是否存在
440.        echo -e "输入课程代号："
```

```
441.         read course_id
442.         #执行 sql 语句，查找工号是否已经存在
443.         MYSQL="mysql -ubucket -p159512 --default-character-se
     t=utf8 -A -N"
444.         sql="select * from course where cid=$course_id"
445.         result="$($MYSQL -e "$sql")"
446.         #如果代号存在，则删除
447.         #否则不进行删除
448.         if [ ! -n "$result" ]; then
449.             echo -e "代号不存在！"
450.         else
451.             sql="delete from course where cid=$course_id"
452.             $MYSQL -e "$sql"
453.             #检查 sql 语句是否正常执行
454.             if [[ $? -eq 0 ]]
455.             then
456.                 echo -e "\033[32m 执行删除成功！\033[0m"
457.             else
458.                 echo -e "\033[31m 执行删除失败！\033[0m"
459.                 #执行 sql 语句异常提示
460.                 exit -1
461.             fi
462.         fi
463.         #回到上级菜单
464.         manage_course
465.         exit 1
466.     }
467.
468.     #课程信息菜单显示
469.     manage_course()
470.     {
471.         echo -e ""
472.         echo -e "请选择操作："
473.         echo -e "1\t 新建课程信息"
474.         echo -e "2\t 编辑课程信息"
475.         echo -e "3\t 删除课程信息"
476.         echo -e "4\t 显示课程信息"
477.         echo -e "5\t 返回上层"
478.         echo -e "0\t 退出系统"
479.         read course_mode
480.         case $course_mode in
481.             1) create_course;;
482.             2) edit_course;;
```

```
            3) delete_course;;
            4) display_course;;
            5) teacher_function
                exit 1;;
            0) exit 0;;
            *) echo -e "非法输入！"
                manage_course
                exit 1;;
        esac
        exit 1
    }

    create_lab()
    {
        #显示提示信息
        echo -e "输入 0 退出此功能，其他输入创建作业"
        #判断是否回到上级菜单
        read state
        if test $state -eq 0; then
            echo -e "回到菜单"
            sleep 2
            teacher_function
            exit 1
        fi
        #判断课程代号是否存在
        echo -e "输入作业所属课程代号："
        read course_id
        #执行 sql 语句，查找代号是否已经存在
        MYSQL="mysql -ubucket -p123456 -Dmanage --default-cha
racter-set=utf8 -A -N"
        sql="select * from course where cid='$course_id'"
        result="$($MYSQL -e "$sql")"
        #代号存在则新建作业
        #代号存在则退出
        if [ ! -n "$result" ]; then
            echo -e "代号不存在！"
            manage_lab
            exit 1

        else
            echo -e "请输入作业编号："
            read lab_id
            sql="select * from lab where cid='$course_id' and
 lid='$lab_id'"
```

```
525.
526.            result="$($MYSQL -e "$sql")"
527.            if [ ! -n "$result" ]; then
528.                echo -e "请输入作业名称："
529.                read lab_name
530.                echo -e "请输入作业内容："
531.                read lab_content
532.                sql="insert into lab values('$course_id', '$l
    ab_id', '$lab_name', '$lab_content')"
533.                $MYSQL -e "$sql"
534.                if [[ $? -eq 0 ]]
535.                then
536.                    echo -e "\033[32m 新建成功！\033[0m"
537.                else
538.                    echo -e "\033[31m 新建失败！\033[0m"
539.                    manage_student
540.                    #执行 sql 语句异常提示
541.                    exit -1
542.                fi
543.            else
544.                echo -e "作业编号已存在！"
545.                manage_lab
546.                exit 1
547.            fi
548.        fi
549.
550.        #回到 teacher 主菜单
551.        manage_lab
552.        #未知错误
553.        exit 1
554.    }
555.
556.    #编辑作业信息
557.    edit_lab()
558.    {
559.        #显示提示信息
560.        echo -e "输入 0 退出此功能，其他输入创建作业"
561.        #判断是否回到上级菜单
562.        read state
563.        if test $state -eq 0; then
564.            echo -e "回到菜单"
565.            sleep 2
566.            teacher_function
567.            exit 1
```

```
568.        fi
569.        #判断课程代号是否存在
570.        echo -e "输入作业所属课程代号："
571.        read course_id
572.        #执行 sql 语句，查找代号是否已经存在
573.        MYSQL="mysql -ubucket -p123456 -Dmanage --default-cha
    racter-set=utf8 -A -N"
574.        sql="select * from course where cid='$course_id'"
575.        result="$($MYSQL -e "$sql")"
576.        if [ ! -n "$result" ]; then
577.            echo -e "代号不存在！"
578.            manage_lab
579.            exit 1
580.        fi
581.        #查找作业编号是否存在
582.
583.        echo -e "请输入作业编号："
584.        read lab_id
585.        sql="select * from lab where cid='$course_id' and lid
    ='$lab_id'"
586.        result="$($MYSQL -e "$sql")"
587.        if [ ! -n "$result" ]; then
588.            echo -e "作业不存在！"
589.            manage_lab
590.            exit 1
591.        fi
592.        echo -e "1\t 修改作业名称"
593.        echo -e "2\t 修改作业内容"
594.        read edit_mode
595.        if test edit_mode -eq 1; then
596.            echo -e "输入修改后的作业名称："
597.            read edit_name
598.            sql="update lab set lname='$edit_name'"
599.            $MYSQL -e "$sql"
600.            if [[ $? -eq 0 ]]
601.            then
602.                echo -e "\033[32m 修改成功！\033[0m"
603.            else
604.                echo =e "\033[31m 修改失败！\033[0m"
605.                manage_lab
606.                #执行 sql 语句异常提示
607.                exit -1
608.            fi
609.        else
```

```
610.            echo -e "输入修改后的作业内容："
611.            read edit_content
612.            sql="update lab set lcontent='$edit_content'"
613.            $MYSQL -e "$sql"
614.            if [[ $? -eq 0 ]]
615.            then
616.                echo -e "\033[32m 修改成功！\033[0m"
617.            else
618.                echo =e "\033[31m 修改失败！\033[0m"
619.                manage_lab
620.                #执行 sql 语句异常提示
621.                exit -1
622.            fi
623.        fi
624.        manage_lab
625.        exit 1
626.    }
627.
628.    #删除作业
629.
630.    delete_lab()
631.    {
632.        #显示提示信息
633.        echo -e "输入 0 退出此功能，其他输入创建作业"
634.        #判断是否回到上级菜单
635.        read state
636.        if test $state -eq 0; then
637.            echo -e "回到菜单"
638.            sleep 2
639.            teacher_function
640.            exit 1
641.        fi
642.        #判断课程代号是否存在
643.        echo -e "输入作业所属课程代号："
644.        read course_id
645.        #执行 sql 语句，查找代号是否已经存在
646.        MYSQL="mysql -ubucket -p123456 -Dmanage --default-cha
    racter-set=utf8 -A -N"
647.        sql="select * from course where cid='$course_id'"
648.        result="$($MYSQL -e "$sql")"
649.        if [ ! -n "$result" ]; then
650.            echo -e "代号不存在！"
651.            manage_lab
652.            exit 1
```

```
653.        fi
654.        #查找作业编号是否存在
655.
656.        echo -e "请输入作业编号：  "
657.        read lab_id
658.        sql="select * from lab where cid='$course_id' and lid
    ='$lab_id'"
659.        result="$($MYSQL -e "$sql")"
660.        if [ ! -n "$result" ]; then
661.            echo -e "作业不存在！ "
662.            manage_lab
663.            exit 1
664.        fi
665.        #进行删除
666.        sql="delete from lab where cid='$course_id' and lid='
    $lab_id'"
667.        $MYSQL -e "$sql"
668.        if [[ $? -eq 0 ]]
669.        then
670.            echo -e "\033[32m 修改成功！ \033[0m"
671.        else
672.            echo =e "\033[31m 修改失败！ \033[0m"
673.             manage_lab
674.            #执行 sql 语句异常提示
675.            exit -1
676.        fi
677.        manage_lab
678.        exit 1
679.
680.    }
681.
682.    #显示某课程下所有作业
683.    display_lab()
684.    {
685.        #显示提示信息
686.        echo -e "输入 0 退出此功能，其他输入创建作业"
687.        #判断是否回到上级菜单
688.        read state
689.        if test $state -eq 0; then
690.            echo -e "回到菜单"
691.            sleep 2
692.            teacher_function
693.            exit 1
694.        fi
```

```
695.        #提示输出课程号
696.        echo -e "输入要查询作业的课程号："
697.        read course_id
698.        #执行 sql 语句，查找代号是否已经存在
699.        MYSQL="mysql -ubucket -p123456 -Dmanage --default-cha
    racter-set=utf8 -A -N"
700.        sql="select * from course where cid='$course_id'"
701.        result="$($MYSQL -e "$sql")"
702.        if [ ! -n "$result" ]; then
703.            echo -e "代号不存在！"
704.            manage_lab
705.            exit 1
706.        fi
707.        sql="select * from lab where cid='$course_id'"
708.        result="$($MYSQL -e "$sql")"
709.        if [[ $? -eq 0 ]]
710.        then
711.            echo -e "\033[32m 查询成功！\033[0m"
712.            echo ${result[*]}
713.        else
714.            echo -e "\033[31m 查询失败！\033[0m"
715.             manage_lab
716.            #执行 sql 语句异常提示
717.            exit -1
718.        fi
719.        manage_lab
720.        exit 1
721.    }
722.
723.    #显示作业完成情况
724.    check_lab()
725.    {
726.        #显示提示信息
727.        echo -e "输入 0 退出此功能，其他输入创建作业"
728.        #判断是否回到上级菜单
729.        read state
730.        if test $state -eq 0; then
731.            echo -e "回到菜单"
732.            sleep 2
733.            teacher_function
734.            exit 1
735.        fi
736.        #提示输出课程号
737.        echo -e "输入要查询作业的课程号："
```

```
738.        read course_id
739.        #执行 sql 语句，查找代号是否已经存在
740.        MYSQL="mysql -ubucket -p123456 -Dmanage --default-cha
   racter-set=utf8 -A -N"
741.        sql="select * from course where cid=$course_id"
742.        result="$($MYSQL -e "$sql")"
743.        if [ ! -n "$result" ]; then
744.            echo -e "代号不存在！"
745.            manage_lab
746.            exit 1
747.        fi
748.        echo -e "已完成作业名单："
749.        sql="select * from submit where cid='$course_id' and
   ac='1'"
750.        result="$($MYSQL -e "$sql")"
751.        if [[ $? -eq 0 ]]
752.        then
753.            echo -e "\033[32m 查询成功！\033[0m"
754.            echo ${result[*]}
755.        else
756.            echo =e "\033[31m 查询失败！\033[0m"
757.             manage_lab
758.            #执行 sql 语句异常提示
759.            exit -1
760.        fi
761.        manage_lab
762.        exit 1
763.    }
764.
765.    #管理作业信息
766.    manage_lab()
767.    {
768.        echo -e "请选择操作："
769.        echo -e "1\t 新建作业"
770.        echo -e "2\t 编辑作业"
771.        echo -e "3\t 删除作业"
772.        echo -e "4\t 显示作业"
773.        echo -e "5\t 显示完成情况"
774.        echo -e "6\t 返回上级目录"
775.        echo -e "0\t 退出系统"
776.        read lab_mode
777.        case $lab_mode in
778.            1) create_lab;;
779.            2) edit_lab;;
```

```
780.            3) delete_lab;;
781.            4) display_lab;;
782.            5) check_lab;;
783.            6) teacher_function
784.                exit 1;;
785.            0) echo -e "非法输入！"
786.                manage_lab
787.                exit 1;;
788.        esac
789.
790.        exit 1
791.    }
792.
793.    #修改密码
794.
795.    change_pwd()
796.    {
797.         #显示提示信息
798.        echo -e "输入 0 退出此功能，其他输入创建作业"
799.        #判断是否回到上级菜单
800.        read state
801.        if test $state -eq 0; then
802.            echo -e "回到菜单"
803.            sleep 2
804.            teacher_function
805.            exit 1
806.        fi
807.        echo -e "1\t 修改学生密码"
808.        echo -e "2\t 修改教师密码"
809.        read change_mode
810.        if test $change_mode -eq 1; then
811.            echo -e "输入学生学号："
812.            #检查学生是否存在
813.            read student_id
814.            MYSQL="mysql -ubucket -p123456 -Dmanage --default
    -character-set=utf8 -A -N"
815.            sql="select * from student where sid=$student_id"
816.            result="$($MYSQL -e "$sql")"
817.            #学号不存在则退出
818.            if [ ! -n "$result" ]; then
819.                echo -e "学号不存在！"
820.                teacher_function
821.                exit 1
822.            fi
```

```
823.            echo -e "输入更改后的密码："
824.            read change_password
825.            sql="update student set spassword='$change_passwo
    rd'"
826.            if [[ $? -eq 0 ]]
827.            then
828.                echo -e "\033[32m 修改成功！\033[0m"
829.            else
830.                echo =e "\033[31m 修改失败！\033[0m"
831.                change_pwd
832.                #执行 sql 语句异常提示
833.                exit -1
834.            fi
835.        else
836.            echo -e "输入更改后的密码："
837.            read change_password
838.            sql="update teacher set tpassword='$change_passwo
    rd'"
839.            if [[ $? -eq 0 ]]
840.            then
841.                echo -e "\033[32m 修改成功！\033[0m"
842.            else
843.                echo =e "\033[31m 修改失败！\033[0m"
844.                change_pwd
845.                #执行 sql 语句异常提示
846.                exit -1
847.            fi
848.        fi
849.        teacher_function
850.        exit 1
851.
852.    }
853.
854.    #教师功能函数
855.    teacher_function()
856.    {
857.        #显示功能菜单
858.        echo -e ""
859.        echo -e "选择以下操作："
860.        echo -e "1\t 管理选课名单"
861.        echo -e "2\t 管理课程信息"
862.        echo -e "3\t 管理作业信息"
863.        echo -e "4\t 修改用户密码"
864.        echo -e "0\t 退出系统"
```

```
865.        read input
866.        #模式选择
867.        case $input in
868.            1) manage_student;;
869.            2) manage_course;;
870.            3) manage_lab;;
871.            4) change_pwd;;
872.            0) echo -e "退出系统！"
873.                sleep 2
874.                exit 0;;
875.            *) echo -e "输入错误！"
876.                teacher_function
877.                exit 1;;
878.        esac
879.        #未知错误
880.        exit 1
881.    }
882.
883.    #教师模式登陆界面
884.    main_teacher()
885.    {
886.
887.        echo -e "请输入教师工号："
888.        read teacher_id
889.        #检查教师信息是否存在
890.        MYSQL="mysql -ubucket -p123456 -Dmanage --default-cha
    racter-set=utf8 -A -N"
891.        sql="select * from teacher where tid='$teacher_id'"
892.        result="$($MYSQL -e "$sql")"
893.        if [ ! -n "$result" ]; then
894.            echo -e "工号不存在！"
895.            teacher_function
896.            exit 1
897.        fi
898.        sql="select tpassword from teacher where tid='$teache
    r_id'"
899.        result="$($MYSQL -e "$sql")"
900.        password=$result
901.        #exit_flag 作为退出输入密码flag
902.     stty -echo
903.        declare -i exit_flag=0
904.        declare -i count_input=3
905.        #判断密码输入是否正确，错误三次退出
906.        while [ $exit_flag -eq 0 ]
```

```
        do
        #输入密码
           echo -e "请输入教师密码："
           read input
           #对比字符串，对比成功即为通过
           if test $password = $input; then
                stty echo
                let exit_flag++
                break
            else
                let count_input = count_input-1
                echo -e "密码错误！你还有$count_input 次机会"

            fi
            #输入次数用完
            if test count_input -eq 0; then
                stty echo
                sleep 2
                exit 3
            fi
        done
        echo -e ""
        echo  "**********欢迎进入教师系统**********"
        #进入功能界面
        teacher_function
        exit 1
    }
```

```
#!/bin/bash
#UTF-8
#**************************************************************
**************
#名称：  student.sh
#作者：  刘星烨
#学号：  3180105954
#功能：  执行管理员功能
#参数：  无
#**************************************************************
**************

#显示所有课程
display_course()
```

```
{
    #显示提示信息
    echo -e "输入 0 退出此功能，其他输入显示课程"
    #判断是否回到上级菜单
    read state
    if test $state -eq 0; then
        echo -e "回到菜单"
        sleep 2
        student_function
        exit 1
    fi
    MYSQL="mysql -ubucket -p123456 -Dmanage --default-character-set=utf8 -A -N"
    sql="select * from choose_course where sid='$student_id'"
    result="$($MYSQL -e "$sql")"
    if [[ $? -eq 0 ]]
    then
        echo -e "\033[32m 查询成功！\033[0m"
        echo ${result[*]}
    else
        echo -e "\033[31m 查询失败！\033[0m"
        student_function
        #执行 sql 语句异常提示
        exit -1
    fi
    student_function
    exit 1
}

#学生提交作业
submit_lab()
{
    #显示提示信息
    echo -e "输入 0 退出此功能，其他输入提交作业"
    #判断是否回到上级菜单
    read state
    if test $state -eq 0; then
        echo -e "回到菜单"
        sleep 2
        student_function
        exit 1
    fi
    #提示输出课程号
    echo -e "输入要查询作业的课程号："
```

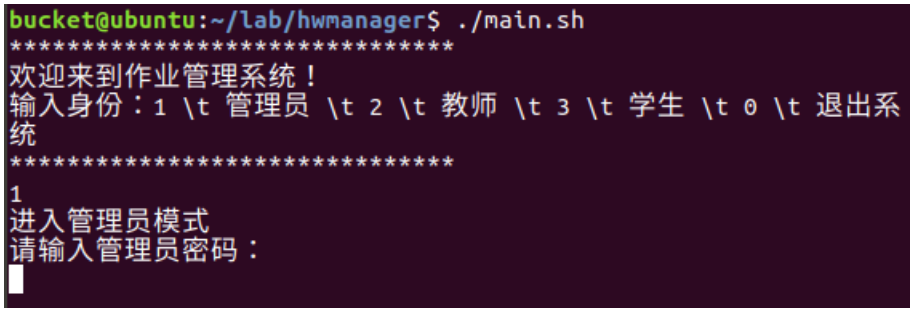
```
    read course_id
    #执行sql语句，查找代号是否已经存在
    MYSQL="mysql -ubucket -p123456 -Dmanage --default-charact
er-set=utf8 -A -N"
    sql="select * from course where cid='$course_id'"
    result="$($MYSQL -e "$sql")"
    if [ ! -n "$result" ]; then
        echo -e "代号不存在！"
        manage_lab
        exit 1
    fi
    #查找作业编号是否存在

    echo -e "请输入作业编号："
    read lab_id
    sql="select * from lab where cid='$course_id' and lid='$l
ab_id'"
    result="$($MYSQL -e "$sql")"
    if [ ! -n "$result" ]; then
        echo -e "作业不存在！"
        manage_lab
        exit 1
    fi
    sql="update submit set ac='1' where sid='$student_id' and
 cid='$course_id' and lid='$lab_id'"
    $MYSQL -e "$sql"
        if [[ $? -eq 0 ]]
        then
            echo -e "\033[32m 修改成功！\033[0m"
        else
            echo -e "\033[31m 修改失败！\033[0m"
            manage_lab
            #执行sql语句异常提示
            exit -1
        fi
    student_function
    exit 1

}

#修改密码
change_password()
{
    #显示提示信息
```

```
97.     echo -e "输入 0 退出此功能，其他输入创建作业"
98.     #判断是否回到上级菜单
99.     read state
100.        if test $state -eq 0; then
101.            echo -e "回到菜单"
102.            sleep 2
103.            student_function
104.            exit 1
105.        fi
106.        echo -e "输入更改后的密码："
107.            read change_password
108.            sql="update student set spassword='$change_passwo
    rd'"
109.            if [[ $? -eq 0 ]]
110.            then
111.                echo -e "\033[32m 修改成功！\033[0m"
112.            else
113.                echo =e "\033[31m 修改失败！\033[0m"
114.                change_pwd
115.                #执行 sql 语句异常提示
116.                exit -1
117.            fi
118.        student_function
119.        exit 1
120.
121.    }
122.
123.    #学生主函数
124.
125.    student_function()
126.    {
127.        echo -e "请选择操作："
128.        echo -e "1\t 查看所有课程"
129.        echo -e "2\t 管理作业"
130.        echo -e "3\t 修改密码"
131.        echo -e "0\t 退出系统"
132.        read student_mode
133.        case $student_mode in
134.            1) display_course;;
135.            2) submit_lab;;
136.            3) change_password;;
137.            0) echo -e "退出系统！"
138.                sleep 2
139.                exit 0;;
```

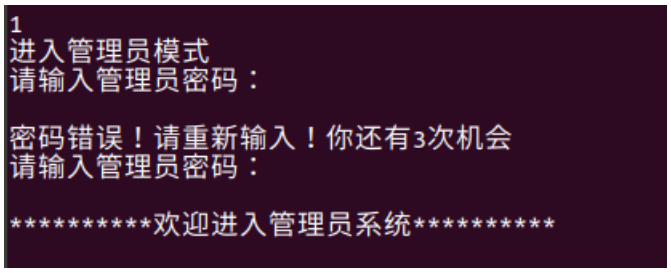
```
            *) echo -e "错误输入！"
                student_function
                exit 1;;
        esac

        exit 1
    }


    main_student()
    {
        echo -e "请输入学号："
        read student_id
        MYSQL="mysql -ubucket -p123456 -Dmanage --default-character-set=utf8 -A -N"
        sql="select * from student where sid='$student_id'"
        result="$($MYSQL -e "$sql")"
        if [ ! -n "$result" ]; then
            echo -e "学号不存在！"
            main_student
            exit 1
        fi
        sql="select spassword from student where sid='$student_id'"
        password="$($MYSQL -e "$sql")"
        stty -echo
        #exit_flag 作为退出输入密码flag
        declare -i exit_flag=0
        declare -i count_input=3
        #判断密码输入是否正确，错误三次退出
        while [ $exit_flag -eq 0 ]
        do
        #输入密码
           echo -e "请输入学生密码："
           read input
           #对比字符串，对比成功即为通过
           if test $password = $input; then
                stty echo
                let exit_flag++
                break
            else
                let count_input = count_input-1
                echo -e "密码错误！你还有$count_input 次机会"

```

```
            fi
            #输入次数用完
            if test count_input -eq 0; then
                stty echo
                sleep 2
                exit 3
            fi
        done
        echo -e ""
        echo  "**********欢迎进入学生系统**********"
        #进入功能界面
        student_function
        exit 1
    }
```

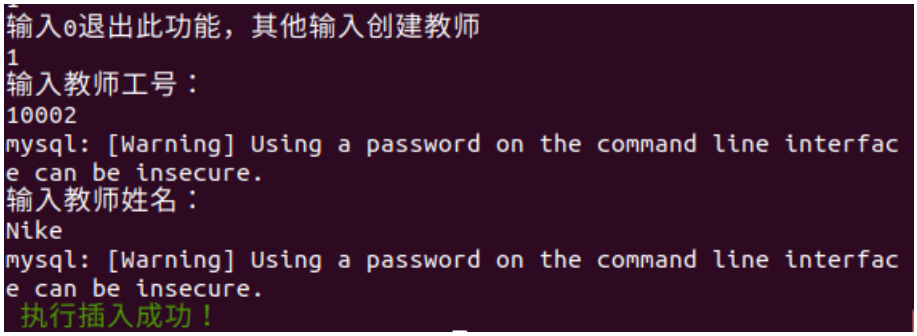
- 实验结果：
- 进入管理员模式，输入不回显



- 输入密码，错误时会有错误提示



- 执行创建教师功能



- 修改信息

```
3
输入0退出此功能，其他输入编辑教师信息
1
输入教师工号：
10002
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
1       修改教师姓名    2       修改教师密码
2
输入修改后密码：
123456
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
 执行修改成功！
```

- 删除教师

```
2
输入0退出此功能，其他输入删除教师
2
输入教师工号：
10002
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
 执行删除成功！
```

- 查询教师信息

```
输入0退出此功能，其他输入显示教师信息
1
选择查询方式：
1       输入工号查询
2       输入姓名查询
3       显示所有教师
4       返回上层
0       退出
3
显示所有教师信息：
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
 查询成功！
10001 James 123456 10003 Harry 111111
```

- 创建课程

```
5
输入0退出此功能，其他输入创建课程
1
输入课程代号：
10001
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
输入课程名称：
Linux
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
 执行插入成功！
```

- 绑定课程
-

```
输入课程代号：
10002
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
输入教师工号：
10002
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
 绑定成功！
```

- 解除绑定

```
输入课程代号：
10002
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
输入教师工号：
10002
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
 取消绑定成功！
```

- 登陆教师模式

```
2
进入教师模式
请输入教师工号：
10001
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
请输入教师密码：

**********欢迎进入教师系统**********

选择以下操作：
1       管理选课名单
2       管理课程信息
3       管理作业信息
4       修改用户密码
0       退出系统
```

- 添加选课学生

```
1
输入学生学号：
10003
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
学号10003不存在！是否新建？【Y/N】
Y
输入学生姓名：
Cindy
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
 新建成功！
```

-

```
输入课程代号：
10001
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
 插入成功！
```

-
- 查询学生选课情况

```
输入0退出此功能，其他输入进行查询
1
选择查询方式：
1       查询学生选课情况
2       查询课程所有学生
0       退出系统
1
输入学生学号：
10003
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
 查询成功！
10003 10001
```

-
- 创建新的作业

```
输入0退出此功能，其他输入创建作业
1
输入作业所属课程代号：
10001
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
请输入作业编号：
01
```

-

```
请输入作业名称：
case1
请输入作业内容：
For a test
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
 新建成功！
```

-
- 编辑作业

```
1
输入作业所属课程代号：
10001
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
请输入作业编号：
01
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
1       修改作业名称
2       修改作业内容
2
./teacher.sh: line 595: test: edit_mode: integer expression ex
pected
输入修改后的作业内容：
Do a simple test on shell
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
 修改成功！
```

- 删除作业

```
1
输入作业所属课程代号：
10001
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
请输入作业编号：
01
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
 修改成功！
```

- 查看已布置作业

```
输入要查询作业的课程号：
10001
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
 查询成功！
10001 01 Lab1 Do a simple lab 10001 02 lab2 null
```

- 修改自己的密码

```
4
输入0退出此功能，其他输入创建作业
1
1       修改学生密码
2       修改教师密码
2
输入更改后的密码：
654321
 修改成功！
```

- 学生查看课程
-

```
请选择操作：
1       查看所有课程
2       管理作业
3       修改密码
0       退出系统
1
输入0退出此功能，其他输入显示课程
1
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
 查询成功！
10001 10001 10001 10002
```

- 提交作业（此功能暂时只完成提交标记）

```
2
输入0退出此功能，其他输入提交作业
1
输入要查询作业的课程号：
10001
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
请输入作业编号：
01
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
mysql: [Warning] Using a password on the command line interfac
e can be insecure.
 修改成功！
```

- 学生修改密码

```
请选择操作：
1       查看所有课程
2       管理作业
3       修改密码
0       退出系统
3
输入0退出此功能，其他输入创建作业
1
输入更改后的密码：
123456
 修改成功！
```

Myshell

- 设计思想：

- myshell 用 C++编写，遵循结构化程序设计的原则，自顶向下，逐步细化，模块化设计。在程序中大量运用了 Unix/Linux 的系统调用。程序采取的运行命令方式即创建子进程，在子进程内执行命令。这样做虽然增加了执行内部命令时的开销，但可以保证接口的统一和一致，避免了实现后台运行、管道和重定向时的麻烦。前台命令的运行采用同步机制，主进程暂停，等待子进程执行结束，再继续执行下去。后台命令的执行采用异步机制，主进程正常运行，接收到子进程结束的信号后，执行信号处理的函数。在内部命令的实现上，mysh 通过对底层文件系统的调用，实现有关目录操作的功能。在管道和重 定向的实现中，mysh 也是通过操作文件描述符，

达到了改变输入输出的效果。

- 功能模块
- mysh 主要有 3 个部分：进程控制、命令解析、命令执行。
  控制器负责控制主进程的行为，包括从指定的文件流中读入输入并解析，控制命令的执行，接收信号并进行处理；判断启动的方式，选择命令行模式或文件模式；从指定的文件流中逐行读入，调用解析器将输入的字符串解析为命令；解析完成后调用执行器进行执行。 命令解析负责将传入的字符串解析为命令，包括特殊字符、命令、参数。命令执行器负责接收命令，创建子进程，待父进程向进程表内写入信息后开始执行子进程。执行前台命令时，将主进程阻塞，待子进程结束后，进行回收处理；执行后台命令时，注册信号处理 函数，结束执行器，返回控制器继续读入输入。
- 数据结构
  使用了 C++标准库中的许多结构，包括利用向量实现 jobs 以及 env 结构体的插入删除等操作，以及管道运行命令时使用队列存储等。C++标准库中的模板基本上可以满足对于数据结构的要求，在使用时也尽量使用特性最符合的结构。
- 源代码：

```
1. /*************************************************************
   ***************
2. 名称：  environment.h
3. 作者：  刘星烨
4. 学号：  3180105954
5. 功能：  环境变量及进程有关声明
6.
7. *************************************************************
   **************/
8.
9. #ifndef ENVIRONMENT_H_
10.#define ENVIRONMENT_H_
11.
12.#include "main.h"
13.#include<string>
14.using namespace std;
15.//环境变量的结构
16.typedef struct envNode{
17.    string var_name;
18.    string var_val;
19.} envNode;
20.
21.std::vector<envNode> envVar;
22.
23.//增加环境变量
24.void setEnvVal(string name, string value){
```

```
25.     envNode newnode;
26.     newnode.var_name=name;
27.     newnode.var_val=value;
28.     envVar.push_back(newnode);
29. }
30. 
31. //获得环境变量
32. envNode& getEnvAddr(string name){
33. 
34.     for(int i=0;i<envVar.size();++i){
35.         if(envVar[i].val_name==name){
36.             return envVar[i];
37.         }
38.     }
39.     return NULL;
40. }
41. //获得环境变量值
42. string getEnvVal(string name){
43.     envNode node=getEnvAddr(name);
44.     if(node!=NULL){
45.         return node.var_val;
46.     }
47.     else{
48.         cout<<"Wrong variable name!"<<endl;
49.     }
50. }
51. 
52. void setNewVal(string name, string newVal){
53.     envNode& env=getEnvAddr(name);
54.     env.var_val=newVal;
55.     return;
56. }
57. 
58. void delEnv(string name){
59.     vector<envNode>::iterator it = find(envVar.begin(), env
    Var.end(), getEnvAddr(name));
60. 
61.     if(it!=envVar.end()){
62.         envVar.erase(it);
63.     }
64.     else{
65.         cout<<"Can't find variable!"<<endl;
66.     }
67. }
```

```
68.
69.
70.//进程状态及结构定义
71.typedef enum jobType { FG, BG } jobType;
72.typedef struct job
73.{
74.    int pid;
75.    string job_name;
76.    jobType type;
77.    int status;
78.
79.}job;
80.
81.//进程表
82.vector<job> jobList;
83.
84.//添加新的进程
85.job& addJob(int pid, string job_name, jobType type, int sta
   tus){
86.    job newjob;
87.    newjob.pid=pid;
88.    newjob.job_name=job_name;
89.    newjob.type=type;
90.    newjob.status=status;
91.    jobList.insert(jobList.begin(),newjob);
92.    return newjob;
93.}
94.
95.//获得进程结构的引用
96.
97.job& getJobAddr(int pid){
98.    for(int i=0;i<jobList.size();++i){
99.        if(jobList[i].pid==pid){
100.            return jobList[i];
101.         }
102.     }
103.     return NULL;
104. }
105.
106. //删除进程信息
107.
108. void delJob(int pid){
109.
```

```
    vector<job>::iterator it = find(jobList.begin(), jobList.end(), getJobAddr(pid));

    if(it!=jobList.end()){
        jobList.erase(it);
    }
    else{
        cout<<"Can't find process!"<<endl;
    }
}
```

```
/*************************************************************
名称：  command.h
作者：  刘星烨
学号：  3180105954
功能：  实现命令功能

*************************************************************/
#include <stdio.h>
#include <unistd.h>
#include<iostream>
#include <fstream>
#include <algorithm>
#include <sys/stat.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <sstream>
#include <iterator>
#include <string.h>
#include <wordexp.h>
#include <functional>
#include <map>
#include <pwd.h>
#include <vector>
#include <boost/algorithm/string.hpp>
#include <fcntl.h>
#include <sys/types.h>


//pwd 函数
void pwd(string command, string dir)
{
```

```
32.     //直接输出当前目录
33.     cout<<dir<<endl;
34.     return;
35. }
36. //time 函数
37. void time(){
38.     //得到系统时间
39.     std::time_t result = std::time(NULL);
40.     std::cout << std::asctime(std::localtime(&result))<<end
    l;
41.     return;
42. 
43. }
44. 
45. //cd 函数
46. void cd(string dir, string command, string para)
47. {
48.     string newdir=para;
49.     struct stat sb;
50.     //没有参数，不执行
51.     if(newdir == NULL){
52.         return;
53.     }
54.     //命令说明
55.     if (!newdir.compare("-h"))
56.     {
57.         cout << "To change current path" << endl;
58. 
59.         return 0;
60.     }
61.     //更新目录为当前目录
62.     else if (!newdir.compare("."))
63.     {
64.         newdir=dir;
65.     }
66.     //更新目录为上层目录
67.     else if (!new_dir.compare(".."))
68.     {
69.         newdir = dir.substr(0, dir.find_last_of("/\\"));
70. 
71.     }
72.     //判断更新目录字符串是否合法
73.     const char *cstr = newdir.c_str();
74.     if (!(stat(cstr, &sb) == 0 && S_ISDIR(sb.st_mode)))
```

```
75.     {
76.         errorExit("Wrong parameters",PARA_FAILED);
77.
78.     }
79.     //修改目录
80.     if(chdir(newdir)){
81.         exit(CD_FAILED);
82.     }else{
83.         char* curdir;
84.
85.         getcwd(curdir);
86.         string env="pwd";
87.         setNewVal(env, curdir);
88.
89.     }
90.}
91.//判断输入字符串是否为数字
92.bool is_number(const std::string &s)
93.{
94.     return !s.empty() && find_if(s.begin(), s.end(), [](cha
   r c) { return !isdigit(c); }) == s.end();
95.}
96.
97.//echo 函数
98.void echo(string line)
99.{
100.      string output;
101.      //判断是否有变量
102.      if(line.find('$') != line.find('`'))
103.      {
104.          output = getEnvVal(line.substr(line.find('$') + 1
   , line.size()));
105.      }
106.      else
107.      {
108.          //将参数输出
109.          output = line;
110.      }
111.      cout << output << endl;
112.      return;
113. }
114. //clr 函数
115. void clr(){
116.      //输出屏幕控制符
```

```
117.     cout<<"\033[2J\033[0;0H";
118. }
119.
120.
121.
122. //退出
123. void exit(string command, string line,int &ERRNO)
124. {
125.     string value = line.substr(line.find(" ") + 1);
126.     //无参数直接退出
127.     if (command==line)
128.     {
129.         std::cout << "Process finished with exit code 0"
   << std::endl;
130.
131.         ERRNO = 0;
132.
133.         return;
134.     }//含参数，以特定信号推出
135.     else if (is_number(value))
136.     {
137.         std::cout << "Process finished with exit code " <
   < std::stoi(value) << std::endl;
138.
139.         ERRNO = std::stoi(value);
140.         return;
141.     }
142. }
143.
144. //cat 函数
145. void cat(string line){
146.     //判断是否有参数
147.     if(!line.empty()){
148.         //参数为存在的文件
149.         if(is_file_exist(line.c_str())){
150.             ifstream openFile(line.c_str());
151.             char ch;
152.             //输出所有内容
153.             while(!openFile.eof()){
154.                 openFile.get(ch);
155.                 cout<<ch;
156.             }
157.         }
158.     }
```

```
159.     return;
160. }
161.
162. //umask 函数
163. void umask(string command , string line){
164.     if(command!=line){
165.         //获得参数
166.         string value = line.substr(line.find(' ')+1);
167.         mode_t mask = mode_t(atoi(value.c_str()));
168.         //调用系统函数修改 umask
169.         mode_t old_mask = umask(mask);
170.     }
171.     else return;
172.     return;
173. }
174.
175. void environ(){
176.     //遍历向量输出
177.     for(int i=0;i<envVar.size();++i){
178.         cout<<envVar[i].val_name<<"="<<envCar[i].val_val<
    <endl;
179.
180.     }
181.     return;
182. }
183.
184. void set(string command, string line){
185.     //没有参数，显示所有环境变量
186.     if(command==line){
187.         environ();
188.         return;
189.     }
190.     //获得变量名和值
191.     string value = line.substr(line.find(" ") + 1);
192.     string name = value.substr(0, value.find(" ")-1);
193.     value = value.substr(value.find(" ") + 1);
194.     //判断环境变量是否存在，存在则更新
195.     if(getEnvAddr(name)!=NULL){
196.         setEnvVal(name, value);
197.     }else{
198.         setNewVal(name, value);
199.     }
200.     return;
201. }
```

```
202. void unset(string command, string line){
203.     if(command==line){
204.         return;
205.     }
206.     string name=line.substr(line.find(" ") + 1);
207.     //系统环境变量，不能执行删除
208.     if(!name.compare("pwd")||! name.compare("shell")){
209.         return;
210.     }
211.     //变量名存在则删除
212.     if(getEnvAddr(name)==NULL){
213.         return;
214.     }else{
215.         delEnv(name);
216.     }
217. }
218. //test 函数（简单的字符串处理）
219. void test(string command, string line){
220.     if(command==line){
221.         return;
222.     }
223.     //得到参数：字符串及模式
224.     string value = line.substr(line.find(" ")+1);
225.     string option = value.substr(0, value.find(" ")-1);
226.     string str = value.substr(value.find(" ")+1);
227.     //不合法的选项
228.     if(option.size()!=2) return;
229.     //判断是否非空
230.     if(option[1]=='n'){
231.         if(str.empty()){
232.             cout<<"false"<<endl;
233.         }else{
234.             cout<<"true"<<endl;
235.         }
236.     //判断是否空
237.     }else if(option[i]=='z'){
238.         if(str.empty()){
239.             cout<<"true"<<endl;
240.         }else{
241.             cout<<"false"<<endl;
242.         }
243.     }
244. }
245. //输入参数移位
```

```
246. void shift(string command, string line){
247.     string value = line.substr(line.find(" ")+1);
248.     //移位数字与字符串
249.     string shifts = value.substr(0, value.find(" ")-1);
250.     string str = value.substr(value.find(" ")+1);
251.     int shift = atoi(shifts.c_str());
252.     int index;
253.     //根据空格分隔变量,去除 shift 个变量
254.     for(int i=0;i<shift;++i){
255.         if((index = str.find(' ',index)) != string::npos)
    continue;
256.         else break;
257.     }
258.
259.     //取移位后子串
260.     if(index!=string::npos) str = str.substr(index+1);
261.     cout<<str<<endl;
262. }
263. //jobs 函数
264. void jobs(){
265.     //遍历 job 向量
266.     for(int i=0;i<jobList.size();++i){
267.         //找到符合条件的进程
268.         if(jobList[i].pid&&BG==jobList[i].type){
269.             if(jobList[i].status==RUN){
270.                 cout<<jobList[i].pid<<jobList[i].name<<"r
   unning"<<endl;
271.             }else{
272.                 cout<<jobList[i].pid<<jobList[i].name<<"s
   uspend"<<endl;
273.             }
274.         }
275.     }
276. }
277. //fg 函数
278. void fg(){
279.     job bg_job= jobList.at(0);
280.     if(bg_job!=NULL){
281.         cout<<bg_job.pid<<bg_job.name<<"running"<<endl;
282.         while(true);
283.     }else{
284.         cout<<"no current job\n";
285.     }
286. }
```

```
287.
288. void bg(){
289.     for(int i=0;i<jobList.size();++i){
290.         if(jobList[i].pid&&BG==jobList[i].type&&jobList[i
   ].status==SUSPEND){
291.             jobList[i].status = RUN;
292.             kill(jobList[i].pid, SIGCONT);
293.             cout<<jobList[i].pid<<jobList[i].name<<"runni
   ng"<<endl;
294.         }
295.     }
296. }
```

```
1. using std::cout;
2. using std::cerr;
3. using std::endl;
4. using std::string;
5.
6. using namespace std;
7.
8. map<string, string> global_variable;
9.
10.//得到当前目录
11.string GetCurrentWorkingDir(void)
12.{
13.    char buff[FILENAME_MAX];
14.//系统调用
15.    getcwd(buff, FILENAME_MAX);
16.    string current_working_dir(buff);
17.
18.    return current_working_dir;
19.}
20.
21.
22.//判断命令是否后台运行
23.int is_bg(string line){
24.    int line_size = line.size();
25.    //判断名令结尾字符
26.    if(line[line_size-1]=='&'){
27.        return 1;
28.    }
29.    return 0;
30.}
31.
```

```
32.//判断文件是否存在
33.
34.inline bool file_exists(const std::string &name)
35.{
36.
37.    if (FILE *file = fopen(name.c_str(), "r"))
38.    {
39.        //正常打开
40.        fclose(file);
41.
42.        return true;
43.    }
44.    else
45.        {
46.        return false;
47.    }
48.}
49.//文件存在上层目录
50.bool exist_in_parrent_dir(string name, string p_dir){
51.    //指定目录打开文件
52.    if (FILE *file = fopen((p_dir+"/"+name).c_str(), "r"))
53.    {
54.        fclose(file);
55.        //正常打开即存在
56.        return true;
57.    }
58.    else
59.    {
60.        return false;
61.    };
62.
63.}
64.//将字符串分解为单个参数
65.vector<string> split_command(string command){
66.    stringstream ss(command);
67.    istream_iterator<string> begin(ss);
68.    istream_iterator<string> end;
69.    //迭代器初始化为命令向量
70.    vector<string> splitted_command(begin, end);
71.    return splitted_command;
72.}
73.//判断是否重定向输出
74.int checkOut(string line) {
75.    vector <string> comm = split_command(line);
```

```
76.     //查找重定向符
77.     if(comm[comm.size()-2].find(">")!=string::npos && comm[
   comm.size()-2].size()==1) {
78.         //找到
79.         return 1;
80.     }else if(comm[comm.size()-2].find(">")!=string::npos &&
    comm[comm.size()-2].size()!=1){
81.         return 2;
82.     }
83.     return 0;
84. }
85. 
86. string change_line(string line){
87.     vector<string> new_line = split_command(line);
88.     line="";
89.     for(int i=0;i<new_line.size()-2;i++){
90.         if(i!=new_line.size()-3){
91.             line+=new_line[i]+" ";
92.         }else{
93.             line+=new_line[i];
94.         }
95.     }
96.     return line;
97. }
98. //建立进程，输出重定向
99. void fork_ex_out(string command, int isBG, string line, int
    &ERRNO)
100. {
101.     string old_line = line;
102.     line = change_line(line);
103.     //得到参数向量
104.     vector<string> splited_command = split_command(old_li
   ne);
105.     //字符串解析，获得结果字符串个数
106.     wordexp_t p;
107.     char **w;
108.     char *ch = new char[line.length() + 1];
109.     strcpy(ch, line.c_str());
110.     wordexp(ch, &p, 0);
111.     w = p.we_wordv;
112.     //创建进程
113.     pid_t parent = getpid();
114.     pid_t pid = fork();
115. 
```

```
116.     if (pid == -1)
117.     {
118.         std::cout << "Bad parameters. Try again" << std::
     endl;
119.         ERRNO = 1;
120.         exit(EXIT_FAILURE);
121.     }
122.     else if (pid > 0 && !isBG)
123.     {
124.         //父进程
125.         int status;
126.         waitpid(pid, &status, 0);
127.     }
128.     else {
129.         //判断命令合法
130.         if(file_exists(command)){
131.             if(checkOut(old_line)==1){
132.                 int fd = open(splited_command[splited_com
     mand.size()-1].c_str(), O_WRONLY | O_TRUNC | O_CREAT, S_IRU
     SR | S_IRGRP | S_IWGRP | S_IWUSR);
133.                 //建立文件通信
134.                 dup2(fd, 1);
135.                 dup2(fd, 2);
136.
137.                 close(fd);
138.             }else if(checkOut(old_line)==2){
139.                 ofstream outfile (splited_command[splited
     _command.size()-1]);
140.                 outfile << "Wrong parameters. Usage: comm
     and > filename " << std::endl;
141.
142.                 outfile.close();
143.             }
144.             //命令执行
145.             execve(command.c_str(), w, environ);
146.         } else {
147.             //建立传递给文件的数组指针
148.             vector<const char *> args{line.c_str()};
149.             //空字符串结尾
150.             args.push_back(nullptr);
151.             if(checkOut(old_line)==1){
152.                 int fd = open(splited_command[splited_com
     mand.size()-1].c_str(), O_WRONLY | O_TRUNC | O_CREAT, S_IRU
     SR | S_IRGRP | S_IWGRP | S_IWUSR);
```

```
153.
154.                 dup2(fd, 1);
155.                 dup2(fd, 2);
156.
157.                 close(fd);
158.             }else if(checkOut(old_line)==2){
159.                 //获取输出重定向文件
160.                 ofstream outfile (splited_command[splited
    _command.size()-1]);
161.                 outfile << "Wrong parameters. Usage: comm
    and > filename " << std::endl;
162.
163.                 outfile.close();
164.             }
165.             execvp(command.c_str(), const_cast<char *cons
    t *>(args.data()));
166.         }
167.         cout << "Enter the command as the first argument!
    " << endl;
168.         cout << "Use -h | --help to see available command
    s" << endl;
169.
170.         ERRNO = 1;
171.         exit(EXIT_FAILURE);   // exec never returns
172.     }
173. }
174. //判断重定向输入
175. int checkIn(string line) {
176.     vector <string> comm = split_command(line);
177.     if(comm[comm.size()-2].find("<")!=string::npos && com
    m[comm.size()-2].size()==1) {
178.         return 1;
179.     }else if(comm[comm.size()-2].find("<")!=string::npos
    && comm[comm.size()-2].size()!=1){
180.         return 2;
181.     }
182.     return 0;
183. }
184.
185.
186.
187. //建立进程，输入重定向
188. void fork_exIn(string command, int isBG, string line, int
    &ERRNO)
```

```
189. {
190.     string old_line = line;
191.     line = change_line(line);
192.     //得到参数向量
193.     vector<string> splited_command = split_command(old_li
   ne);
194.     string file = splited_command[splited_command.size()-
   1];
195.     if (!is_file_exist(file.c_str())){
196.         cerr << "File " << file << " doesn't exist" << en
   dl;
197.         ERRNO = 1;
198.         return;
199.     }
200.     //字符串解析，获得结果字符串个数
201.     wordexp_t p;
202.     char **w;
203.     char *ch = new char[line.length() + 1];
204.     strcpy(ch, line.c_str());
205.     wordexp(ch, &p, 0);
206.     w = p.we_wordv;
207.     //创建进程
208.     pid_t parent = getpid();
209.     pid_t pid = fork();
210.     //进程是否创建成功
211.     if (pid == -1)
212.     {
213.         cerr << "Bad parameters. Try again" << endl;
214.         ERRNO = 1;
215.         exit(EXIT_FAILURE);
216.     }
217.     else if (pid > 0 && !isBG)
218.     {
219.         int status;
220.         waitpid(pid, &status, 0);
221.     }
222.     else {
223.         //判断命令文件存在
224.         if(file_exists(command)) {
225.             if(checkIn(old_line)==1) {
226.                 int fd = open(splited_command[splited_com
   mand.size()-1].c_str(), O_RDONLY);
227.                 dup2(fd, 0);
228.                 close(fd);
```

```
229.             } else if(checkIn(old_line)==2) {
230.                 cerr << "Wrong parameters. Usage: command < filename " << endl;
231.                 ERRNO = 1;
232.             }
233.             execve(command.c_str(), w, environ);
234.         } else {
235.             vector<const char *> args;
236.             for (int i = 0; i < splited_command.size()-2; i++) {
237.                 string cur = splited_command[i];
238.                 args.push_back(cur.c_str());
239.             }
240.             args.push_back(nullptr);
241.             if(checkIn(old_line)==1) {
242.                 int fd = open(splited_command[splited_command.size()-1].c_str(), O_RDONLY);
243.                 dup2(fd, 0);
244.                 close(fd);
245.             }else if(checkIn(old_line)==2) {
246.                 cerr << "Wrong parameters. Usage: command < filename " << endl;
247.                 ERRNO = 1;
248.             }
249.             char *const * asd = const_cast<char *const *>(args.data());
250.             execvp(command.c_str(), asd);
251.         }
252.         cout << "Enter the command as the first argument! " << endl;
253.         cout << "Use -h | --help to see available commands" << endl;
254.
255.         ERRNO = 1;
256.         exit(EXIT_FAILURE);
257.     }
258. }
259. //创建进程，代码与以上两个函数类似，不需要重定向
260. void fork_ex(string command, int isBG, string line, int &ERRNO)
261. {
262.     wordexp_t p;
263.     char **w;
264.     char *ch = new char[line.length() + 1];
```

```
265.      strcpy(ch, line.c_str());
266.      wordexp(ch, &p, 0);
267.      w = p.we_wordv;
268.
269.      pid_t parent = getpid();
270.      pid_t pid = fork();
271.
272.      if (pid == -1)
273.      {
274.          std::cout << "Bad parameters. Try again" << std::
    endl;
275.          ERRNO = 1;
276.          exit(EXIT_FAILURE);
277.      }
278.      else if (pid > 0 && !isBG)
279.      {
280.          int status;
281.          waitpid(pid, &status, 0);
282.      }
283.                else {
284.          if(file_exists(command)){
285.              execve(command.c_str(), w, environ);
286.          } else {
287.              vector<const char *> args{line.c_str()};
288.              args.push_back(nullptr);
289.              execvp(command.c_str(), const_cast<char *cons
    t *>(args.data()));
290.          }
291.          cout << "Enter the command as the first argument!
    " << endl;
292.          cout << "Use -h | --help to see available command
    s" << endl;
293.
294.          ERRNO = 1;
295.          exit(EXIT_FAILURE);   // exec never returns
296.      }
297. }
298.
299. //执行文件
300. void executeSriptFile(stringstream &fileName, int &ERRNO)
301. {
302.      ifstream inFile(fileName.str());
303.
304.      if(!inFile)
```

```
305.     {
306.         cout << "there are no such file" << endl;
307.        ERRNO = - 1;
308.
309.         return;
310.     }
311.
312.     string line;
313.
314.     while (std::getline(inFile, line))
315.     {
316.         istringstream iss(line);
317.         string command;
318.
319.         if (!(iss >> command))
320.         {
321.             break;
322.         }
323.         if(command.c_str()[0] != '#')
324.         {
325.             // std::cout << command << std::endl;
326.             int isBG = is_bg(line);
327.             //调用函数执行命令
328.             fork_ex(command.c_str(),isBG, line, ERRNO);
329.         }
330.     }
331.
332.     inFile.close();
333.
334.     ERRNO = 1;
335.     return;
336. }
337. //定义变量
338. void var_support(string command)
339. {
340.     string key;
341.     string value;
342.     //获取参数
343.     key = command.substr(0, command.find('='));
344.     value = command.substr(command.find('=')+1,command.fi
    nd(' '));
345.     //如果变量已经存在
346.     if(global_variable.count(key) ==  1)
347.     {
```

```
348.          global_variable[key] = value;
349.          return;
350.      }
351.      //插入一对新的环境变量
352.      global_variable.insert(pair<string,string>(key, value
   ));
353. }
354. //检查是否含管道
355. int checkPipe(const string &line){
356.      //查找'|'
357.      return line.find('|') != string::npos;
358. }
359.
360.
361. //管道执行
362. int executePipeNext(string line, int from_fd, int to_fd,
   int isBG, int &ERRNO) {
363.
364.      pid_t pid = fork();
365.      //获取进程失败
366.      if (pid == -1) {
367.          std::cout << "Bad parameters. Try again" << std::
   endl;
368.          ERRNO = 1;
369.          exit(EXIT_FAILURE);
370.      }
371.      else if (pid > 0 && !isBG) {
372.          // 父进程
373.
374.          int status;
375.          waitpid(pid, &status, 0);
376.          ERRNO = status;
377.
378.          return 0;
379.      }
380.      else {
381.
382.
383.          if (from_fd != -1) {
384.              dup2(STDIN_FILENO ,from_fd);
385.          }
386.          if (to_fd != -1) {
387.              dup2(STDOUT_FILENO , to_fd);
388.          }
```

```
389.         string command = line.substr(0, line.find(' '));
390.
391.         //建立传递给文件的数组指针
392.         vector<const char *> args{line.c_str()};
393.         //最后一位为空指针
394.         args.push_back(nullptr);
395.         //外部命令执行
396.         execvp(command.c_str(), const_cast<char *const *>
    (args.data()));
397.         exit(EXIT_FAILURE);
398.     }
399. }
400.
401. //----------------------------------------
402. void pipeExecute(string line, int &ERRNO){
403.     std::vector<std::string> results;
404.     //以'|'为标识符区分字符串
405.     boost::split(results, line, [](char c){return c == '|
    ';});
406.     deque<string> commands;
407.
408.     //对第一个及最后一个结果分别处理，得到管道指令队列
409.     for (int k = 0; k < results.size(); ++k) {
410.         if(k == 0){
411.             commands.push_back(results[k].substr(0, resul
    ts[k].size()-1));
412.         }
413.
414.         else if(k == results.size() - 1){
415.             commands.push_back(results[k].substr(1, resul
    ts[k].size()));
416.         } else{
417.             commands.push_back(results[k].substr(1, resul
    ts[k].size() - 1));
418.         }
419.     }
420.
421.
422.     size_t  size = commands.size();
423.     int pipefd[commands.size() - 1][2];
424.
425.     for (int i = 0; i < size - 1; ++i) {
426.         //使用pipe 函数实现父子进程通信
427.         if (pipe(pipefd[i]) == -1) {
```

```
428.             for (int j = 0; j < i; ++j) {
429.                 close(pipefd[j][0]);
430.                 close(pipefd[j][1]);
431.             }
432.         }
433.     }
434.
435.
436.     for (int i = 0; i < size; ++i) {
437.         int isBG = is_bg(commands[i]);
438.         if (i == 0) {
439.             //第一条指令，开放写端
440.             executePipeNext(commands[i], -1, pipefd[i][1]
    ,isBG, ERRNO);
441.             close(pipefd[i][1]);
442.         }
443.         //中间的指令，输入输出均建立通信
444.         else if (i > 0 && i < (size - 1)){
445.
446.             executePipeNext(commands[i], pipefd[i - 1][0]
    , pipefd[i][1], isBG, ERRNO);
447.             close(pipefd[i - 1][0]);
448.             close(pipefd[i][1]);
449.         }
450.         //最后一条指令，开放读端
451.         else {
452.             executePipeNext(commands[i], pipefd[i - 1][0]
    , -1, isBG, ERRNO);
453.             close(pipefd[i - 1][0]);
454.         }
455.     }
456.     return;
457. }
458.
459. //-------------------------------------------------------
    ---------------------------
460. void execCommand(string line, int &ERRNO){
461.     if (line.empty())
462.         {
463.             continue;
464.         }
465.         //help 命令
466.         if (!line.compare("help"))
467.         {
```

```
468.                ifstream openFile("myshell_help");
469.                char ch;
470.                //输出 help 文件内容
471.                while(!openFile.eof()){
472.                    openFile.get(ch);
473.                    cout<<ch;
474.                }
475.
476.                continue;
477.            }
478.            //检查是否有管道
479.            if(checkPipe(line)) {
480.                pipeExecute(line, ERRNO);
481.                continue;
482.            }
483.            isBG = is_bg(line);
484.            //执行文件
485.            if(line[0] == '.')
486.            {
487.                command = line.substr(1, line.find(' '));
488.                stringstream stringstream1{command};
489.                executeSriptFile(stringstream1, ERRNO);
490.
491.                continue;
492.            }
493.            if (line[0] != '\"')
494.            {
495.                command = line.substr(0, line.find(' '));
496.            }
497.            else
498.            {
499.                command = line.substr(1, line.find_last_of('\
    "') - 1);
500.            }
501.            //输出重定向
502.            if(line.find(">")!=string::npos){
503.                fork_ex_out(command,isBG,line,ERRNO);
504.            }
505.            //输入重定向
506.            if(line.find("<")!=string::npos){
507.                fork_exIn(command,isBG,line,ERRNO);
508.            }
509.            //echo 指令
510.            if (!command.compare("echo"))
```

```
511.         {
512.             echo(line.substr(line.find(' ') + 1, line.siz
    e()));
513.         }//pwd 指令
514.         else if (!command.compare("pwd"))
515.         {
516.             pwd(command, line, currentdir);
517.         }//time 命令
518.         else if (!command.compare("time"))
519.         {
520.             time();
521.
522.         }//cd 命令
523.         else if (!command.compare("cd"))
524.         {
525.             string mcdreturn = mcd(currentdir, command, l
    ine);
526.
527.             if (mcdreturn.size() > 1)
528.             {
529.                 chdir(mcdreturn.c_str());
530.                 currentdir = mcdreturn;
531.             }
532.             else
533.             {
534.                 ERRNO = atoi(mcdreturn.c_str());
535.             }
536.         }//exit 命令
537.         else if (!command.compare("exit"))
538.         {
539.             int check = 1;
540.             exit(command, line, check, ERRNO);
541.
542.             if (check)
543.             {
544.                 break;
545.             }
546.         }
547.         else if(!command.compare("clr")){
548.             clr();
549.         }
550.         else if(!command.compare("cat")){
551.             cat(line.substr(line.find(' ')+1));
552.         }
```

```
553.          else if(!command.compare("umask")){
554.              umask(line.substr(0, line.find(' ')-1),line);
555.          }
556.          else if(!command.compare("environ")){
557.              environ();
558.          }else if(!command.compare("set")){
559.              set(line.substr(0, line.find(' ')-1),line);
560.          }
561.          else if(!command.compare("unset")){
562.              unset(line.substr(0, line.find(' ')-1),line);
563.          }
564.          else if(!command.compare("test")){
565.              test(line.substr(0, line.find(' ')-1),line);
566.          }
567.          else if(!command.compare("shift")){
568.              shift(line.substr(0, line.find(' ')-1),line);
569.          }
570.          else if(!command.compare("jobs")){
571.              jobs();
572.          }
573.          else if(!command.compare("fg")){
574.              fg();
575.          }
576.          else if(!command.compare("bg")){
577.              bg();
578.          }
579.          else if(!command.compare("exec")){
580.              line = line.substr(line.find(' ')+1);
581.              execCommand(line, ERRNO);
582.          }
583.          else if(line.find(">")==string::npos && line.find
    ("<")==string::npos)
584.          {
585.              if(line.find('=') != line.find('`'))
586.              {
587.                  var_support(line);
588.
589.                  continue;
590.              }
591.              if (file_exists(command))
592.              {
593.                  if(line.find("$") != line.find('`'))
594.                  {
```

```
595.                     if(file_exists(ampersant(line.substr(
    line.find('$') + 1, line.size()))))
596.                     {
597.                         line.replace(line.find('$'), ampe
    rsant(line.substr(line.find('$') + 1, line.size())).size()
    , ampersant(line.substr(line.find('$')+ 1, line.size())));
598.                         fork_ex(command, isBG, line, ERRN
    O);
599.                     } else {
600.                         std::cout << "bad param for comma
    nd" << std::endl;
601.
602.                         ERRNO = -2;
603.                     }
604.                 } else {fork_ex(command, isBG, line, ERRN
    O);}
605.             }
606.             else if(exist_in_parrent_dir(command,p_dir))
607.             {
608.                 fork_ex(p_dir+"/"+command, isBG, line, ER
    RNO);
609.             }
610.
611.             else
612.             {
613.                 fork_ex(command, isBG, line, ERRNO);
614.             }
615.         }
616.     }
617. }
618.
619.
620. //主函数，处理读入内容
621. int main(int argc, char** argv)
622. {
623.     char bufer[FILENAME_MAX];
624.     //定义错误码
625.     int ERRNO = 0;
626.
627.     string path_to_subprograms = bufer;
628.     //使用环境变量PATH
629.     if (const char *path = getenv("PATH")) {
630.         string to_path;
631.         to_path = path_to_subprograms + ":" + to_path;
```

```
632.          setenv("PATH", to_path.c_str(), 1);
633.      }
634.      //运行时带有参数
635.      if(argc == 2)
636.      {
637.          string nameFile = argv[1];
638.          stringstream stream{nameFile};
639.          //执行文件
640.          executeSriptFile(stream, ERRNO);
641.      }
642.      //过多参数
643.      else if(argc > 2)
644.      {
645.          cout << "Incorect parameters" << endl;
646.
647.          return -1;
648.      }
649.      string line;
650.      string command;
651.      string currentdir = GetCurrentWorkingDir();
652.      string p_dir = currentdir;
653.      int isBG;
654.      while (true)
655.      {
656.          //显示提示符
657.          cout << currentdir << " $ ";
658.          getline(cin, line);
659.
660.          execCommand(string line, int &ERRNO);
661.      return ERRNO;
662.  }
```

-
- 实验结果
- 输入 help，显示帮助文件

```
/home/bucket/lab/myshell $ help
Introduction
------------
Mysh is a simple shell for Unix/Linux, which is implemented for homework
of Linux Course.
Features
--------
1. Supports inner commands: pwd, time, clear, exit, environ, jobs, fg, bg,
 echo, cd, exec, set,
unset, umask, test, shift.
2. Supports external commands.
3. Supports shell scripts.
4. Supports I/O redirection.
5. Supports pipe.
6. Supports job management, including foreground, background, hangup.
Instructions
------------
1. Basic operation
Some basic inner commands in mysh are as followings.
-time print the current time
-clear clear the terminal screen
```

- cd/pwd 命令

```
/home/bucket/lab/myshell $ cd ..
/home/bucket/lab $ cd /home/bucket
/home/bucket $ pwd
/home/bucket
/home/bucket $
```

- 运行 clr 命令

```
/home/bucket/lab/myshell $
```

- 后台运行命令，jobs 命令显示，fg 转到前台

```
/home/bucket/lab/myshell $ sleep 30 &
/home/bucket/lab/myshell $ jobs
11015 sleep running
/home/bucket/lab/myshell $ fg
11015 sleep running
```

- 重定向功能

```
/home/bucket/lab/myshell $ cat file.in
one two three four five
/home/bucket/lab/myshell $ cat file.out
/home/bucket/lab/myshell $ shift 2 < file.in > file.out
/home/bucket/lab/myshell $
/home/bucket/lab/myshell $ cat file.out
three four five
```

- 管道

```
/home/bucket/lab/myshell $ echo Have a nice day | shift 2
nice day
```

- 显示系统时间

```
/home/bucket/lab/myshell $ time
Wed Aug 19 20:55:34 2020
```

- 环境变量操作

```
/home/bucket/lab/myshell $ set DIST/home/bucket
/home/bucket/lab/myshell $ set COUNT 5
/home/bucket/lab/myshell $ environ
shell=/usr/local/myshell
DIST=/hone/bucket
COUNT=5
/home/bucket/lab/myshell $ unset COUNT 5
/home/bucket/lab/myshell $ environ
shell=/usr/local/myshell
DIST=/hone/bucket
```

- 在 shell 中运行文件

```
/home/bucket/lab/myshell $ cat test.sh
pwd
echo yes
cd ..
/home/bucket/lab/myshell $ .test.sh
/home/bucket/lab/myshell
yes
/home/bucket/lab
```

- exec 命令

```
/home/bucket/lab/myshell $ exec test -n hello
true
```

- 退出程序

```
/home/bucket/lab/myshell $ exit
Process finished with exit code 0
bucket@ubuntu:~/lab/myshell$
```

三、 讨论、心得（10 分）

通过本次实验，我初步掌握了 Unix/Linux 的系统编程，掌握了文件 IO、进程、信号、共享内存等 系统调用的特点和用法。在编写 myshell 程序之前，

我先了解了 shell 的架构，并研究了系统调用函数的使用。虽然是囫囵吞枣，但仍然获益匪浅，了解了 Linux 系统编程 的知识。在编写 myshell 的过程中，我综合运用所掌握的知识，并在实践中不断改进，从软件的框架到 内存中的数据结构，都经过了多次的结构性调整。特别是在加入信号后，每添加一个新功能，都会出现一些莫名的 bug，这个问题直到整个程序快要编写完成时才找到原因并得以解决。事实上正是在编写的的过程中，我去思考每个命令如何产生运行，研究背后的问题，才让我不断认识到之前的一些认知错误。由于时间有限，个人水平有限，代码仍有很多不足之处。比如，内部命令调用的函数应统一接口，传入参数列表，返回整数；进程间的通信可以使用信号量；内部命令应该支持更多的选项；应该支持 shell 语言。我原本想阅读一些现有 shell 程序的代码， 如经典的 bash 和强大的 zsh，但时间已经不允许了。未来若有机会，我希望能更加深入地学习。在 Linux 程序设计课程上，我感到收获颇多。通过这门课程的学习，我对 Unix/Linux 系统有了更多的了解，编程能力也得到了锻炼，期待在操作系统课程上学到更多的有关知识。