

浙江大学实验报告

基本信息

- 课程名称 : Linux程序设计
- 实验项目名称 : Shell程序设计
- 学生姓名 : 徐震
- 学号 : 3180105504
- 专业 : 计算机科学与技术
- 电子邮件地址 : 3180105504@zju.edu.cn
- 实验日期 : 2020.07.28

实验环境

硬件配置

- CPU: 2.6 GHz 6-Core Intel Core i7-9750H
- GPU: NVIDIA® GeForce® GTX 1650 and Intel(R) UHD Graphics 630
- Memory: 16 GB 2666 MHz DDR4
- Disk: 500 GB Solid State PCI-Express Drive * 2

软件环境

- System: Microsoft Windows 10, macOS Catalina 10.15.5 dual booting
- Linux: WSL2 on Windows 10, VMWare Virtual Machine Ubuntu 18.04, Manjaro USB Boot Disk, Ali Cloud ECS Server CentOS 7
- 注意 : 我们会 在VMWare Virtual Machine Ubuntu 18.04上进行绝大多数实验操作 (Host: Windows 10) · 如实验过程中使用了其他系统我们会注明。
- 主要实验环境详细配置 :
 - 系统内核 : Linux ubuntu 5.3.0-43-generic #36~18.04.2-Ubuntu SMP Thu Mar 19 16:03:35 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
 - CPU : Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz
 - Memory : MemTotal: 6060516 kB
- Python 3: 我们使用Python 3来实现MyShell · 模拟Shell的简单功能
 - 经过测试的有 :
 - Python 3.8.2
 - Python 3.7.6
 - Python 3.6.9
 - 经过测试的系统有 :
 - Linux ubuntu 5.4.0-42-generic #46~18.04.1-Ubuntu SMP Fri Jul 10 07:21:24 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
 - Linux aliecs 3.10.0-1127.13.1.el7.x86_64 #1 SMP Tue Jun 23 15:46:38 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux

- macOS Catalina 10.15.5
- 系统命令有很大不同，但Shell可以基本正常运行的系统有：
◦ DESKTOP-XUZH Microsoft Windows 10 Pro 10.0.18363 N/A Build 18363
- 遗憾的是，我们没有经历测试全部的Python版本和所有可能的系统环境，但我们合理推断，在一般的 *nix 环境和 Python 3 下，MyShell都可以正常运行。在Windows环境下，MyShell的基本功能也可正常工作。

实验目的

1. 学习 Bourne shell 的shell脚本的基本概念
2. 学会shell程序如何执行
3. 通过写脚本，学会编写 Bourne shell 脚本程序的方法
4. 理解并掌握shell

实验要求

本实验在提交实验报告时，需要有下面内容：

- **源程序及详细注释**，源程序开始两行为程序名和作者及学号；
- 每题的源代码以文本内容附在对应题目后面；
- 程序运行结果的截图；
- **程序注释必须用中文，海外学生除外**
- 第4、5题需要设计文档
- 讨论与心得
- 第1、2题程序中不能使用sed、awk等工具
- 本实验完成后所有源代码（文本格式）同时上传到**拼题A系统**

实验内容/结果及分析

|

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

源程序

```
1  #!/bin/bash
2  # WordCount
3  # Author: Xu Zhen 徐震 3180105504
4
5  dir=${1:-"."}
6
7  # 统计目录文件个数
8  dir_num=$(ls -l $dir|grep ^d|wc -l|xargs)
9  regular_num=$(ls -l $dir|grep ^-|wc -l|xargs)
10 exec_num=$(find $dir -type f -perm /111 -not \(-path "*/\.*" -o -path "\. "\) | wc -l | xargs)
11 regular_sum=$(find $dir -type f -not \(-path "*/\.*" -o -path "\. "\) -exec du -cb {} + | grep total | tr -d -c 0-9)"
```

```

12 echo "Listing file count info of dir: $dir"
13 echo "Number of directories: $dir_num"
14 echo "Number of regular files: $regular_num"
15 echo "Number of executable files: $exec_num"
16 echo "Total size of regular files: ${regular_sum} Byte(s)"

```

程序运行结果截图

```

xuzh@ubuntu ~/Projects/ShellDesign master ./WordCount/WordCount.sh MyShell
Listing file count info of dir: MyShell
Number of directories: 2
Number of regular files: 10
Number of executable files: 2
Total size of regular files: 5063041 Byte(s)
xuzh@ubuntu ~/Projects/ShellDesign master ll MyShell
total 256K
-rw-rw-r-- 1 xuzh xuzh 3.0K Jul 22 19:23 COLOR.py
-rw-rw-r-- 1 xuzh xuzh 3.1K Jul 30 22:39 dummy.msh
-rw-rw-r-- 1 xuzh xuzh 1.8K Jul 25 12:11 dummy.py
-rw-rw-r-- 1 xuzh xuzh 40 Jul 30 12:29 log.log
drwxrwxr-x 2 xuzh xuzh 4.0K Jul 30 16:38 MyShell.assets
-rw-rw-r-- 1 xuzh xuzh 2.5K Jul 30 16:20 MyShellException.py
-rw-rw-r-- 1 xuzh xuzh 116K Jul 30 22:39 MyShell.md
-rwxrwxr-x 1 xuzh xuzh 91K Jul 30 22:39 MyShell.py
-rwxrwxr-x 1 xuzh xuzh 417 Jul 25 14:09 process.py
drwxrwxr-x 2 xuzh xuzh 4.0K Jul 30 16:24 __pycache__
-rw-rw-r-- 1 xuzh xuzh 11K Jul 29 12:57 should.out
-rw-rw-r-- 1 xuzh xuzh 74 Jul 26 17:43 sleep10s.py
xuzh@ubuntu ~/Projects/ShellDesign master

```

2

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

源程序

手动版

```

1  #!/bin/bash
2  # PalindromeTester
3  # Author: Xu Zhen 徐震 3180105504
4  # 我们注释掉了一些测试用的信息打印语句，读者可取消注释以观察程序运行过程
5  # 我们通过反转并对比字符串来检测其是否为回文
6  # 若我们并非在写脚本，我们或许会通过头尾两个指针做对比来实现这一功能
7  # 但在脚本语言中实现相关的操作会显得很麻烦
8  read -p "The string you want to test is: " word
9  trimmed=""
10 reversed=""
11 # 我们通过下标遍历字符串，剔除非字母部分
12 for i in $(seq 0 ${#word} - 1))
13 do
14     thechar=${word:i:1}
15     # echo "The letter at position $i is: $thechar"
16     if [[ "$thechar" =~ [a-zA-Z] ]]
17     then
18         # echo "And I'm adding it to the trimmed string"
19         trimmed+=$thechar
20         # 我们可以在此处通过本语句构建反转了的字符串
21         # 这样就不需要在下面调用reversed=$(echo $trimmed | rev)
22         reversed="$thechar$reversed"
23     fi

```

```

24 done
25 echo "The trimmed string would be: $trimmed"
26 echo -n "The reversed version of the trimmed: "
27 # 除了在循环中直接反转字符串，我们也可以通过管道调用rev命令获取其反转以检测回文
28 # reversed=$(echo $trimmed | rev)
29 echo $reversed
30 if [ $reversed = $trimmed ]
31 then
32     echo "Gotcha! You're a palindrome!"
33 else
34     echo "Well, you're not a palindrome!"
35 fi

```

简单版

```

1 #!/bin/bash
2 # PalindromeSmall
3 # Author: Xu Zhen 徐震 3180105504
4 # 在本实现中，我们直接调用了tr命令来剔除不合要求的字符串
5
6 # 本语句读入一行用户输入内容并删除非字母内容，复制给word变量
7 word=$(read word; echo $word | tr -d -c a-zA-Z)"
8 # 本语句反转word变量，赋值给reve变量
9 reve=$(echo $word | rev)"
10
11 # DEBUG INFO
12 # echo $word
13 # echo $reve
14
15 # 本语句进行回文测试并打印测试结果
16 [ $word = $reve ] && echo "True" || echo "False"

```

程序运行结果截图

```

xuzh@ubuntu:~/Projects/ShellDesign/PalindromeTester$ ./PalindromeSmall.sh
moon
True
xuzh@ubuntu:~/Projects/ShellDesign/PalindromeTester$ ./PalindromeSmall.sh
monom
True
xuzh@ubuntu:~/Projects/ShellDesign/PalindromeTester$ ./PalindromeSmall.sh
mmmm1234 4321mm
True
xuzh@ubuntu:~/Projects/ShellDesign/PalindromeTester$ ./PalindromeSmall.sh
1222333...3332221
True
xuzh@ubuntu:~/Projects/ShellDesign/PalindromeTester$ ./PalindromeSmall.sh
1222333...3332 221
True
xuzh@ubuntu:~/Projects/ShellDesign/PalindromeTester$ ./PalindromeSmall.sh
1123
True
xuzh@ubuntu:~/Projects/ShellDesign/PalindromeTester$ ./PalindromeSmall.sh
1133
True
xuzh@ubuntu:~/Projects/ShellDesign/PalindromeTester$ ./PalindromeSmall.sh
nanh
False
xuzh@ubuntu:~/Projects/ShellDesign/PalindromeTester$ ./PalindromeSmall.sh
123123ui232131ooiu
True
xuzh@ubuntu:~/Projects/ShellDesign/PalindromeTester$ 

```

3

要求：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，如：

```
1  dirsync ~\dir1 ~\dir2 # ~\dir1为源目录 ~\dir2为目标目录
```

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

1. 备份功能：目标目录将使用来自源目录的最新文件，新文件和新子目录进行升级，源目录将保持不变。dirsync程序能够实现增量备份。
2. 同步功能：两个方向上的旧文件都将被最新文件替换，新文件都将被双向复制。源目录被删除的文件和子目录，目标目录也要对应删除。
3. 其它功能自行添加设计。

提示：不能使用现有的备份或同步程序，如：`/usr/bin/rsync`

源程序

```
1 #!/bin/bash
2 # DirSync
3 # Author: Xu Zhen 徐震 3180105504
4
5 # echo "\$1 is $1"
6 # echo "\$2 is $2"
7 # echo "\$3 is $3"
8
9 ######
10 # 程序说明
11 #####
12
13 # 我们实现了目录备份：
14 #     1. 增添型备份 (-a : append)    : 目标目录中的非重名文件会得到保留
15 #     2. 覆盖型备份 (-r : replace)   : 目标目录中的非重名文件不会得到保留
16 # 与目录同步：
17 #     1. 保守型同步 (-u : update)    : 所有的文件都会得到保留，旧文件会被替换为新文件
18 #     2. 激进型同步 (-s : sync)      : 我们根据文件夹和文件的修改时间戳判断文件是否应得到保
留：
19 #                                     修改时间 (dir1中文件(夹)) 晚于目标文件夹
# (dir2) 的文件
20
21 # 注意，若被同步目录下有同名文件，但两者类型不同，我们不敢贸然复制，请用户手动选择要保留文件夹
还是文件
22
23 # 对于两个文件夹的同步功能，我们必须认识到的一点是：
24 # 若要完全保证所有修改都按照时间顺序进行，我们就不得不进行log记录
25 # 作为一个普通shell脚本而非Microsoft OneDrive这种具有自我储存的文件同步程序
26 # 我们无法在不使用外部储存的方式进行完全按照时间戳的文件同步，但我们可以尽量模拟这一过程
27
28 # 我们采取这样一种模拟操作，在进行同步 (-s : sync) 过程中 (非-u : update，update会保留全部文
件)
29 # 我们只保留修改日期最新的文件
30 # 对于文件夹，我们以文件夹的修改日期和文件的修改日期为标准判断用户是否删除了某一文件，例如：
31 # 在dir1中，用户有file1 · file2 (修改时间为500)，在dir2中也有file1 · 和file2 (修改时间为
500)
32 # 用户在时间点1000删除了目录一的file1，因此dir1的最后修改时间为1000，接着，用户在dir2创建
了file3，时间点为2000
```

```

33 # 我们发现500<1000而1000<2000，因此在同步过程中会删除dir2中的file1，而保留并复制file3
34
35 # 当然这种操作有删除不必要文件的风险（某些新文件可能不会得到同步）
36 # 若用户希望保留所有文件，他/她应使用-u（update）而非-s（sync）
37
38 export PATH="$PATH:."
39
40 copyContent() {
41     # 本函数的功能是进行不检查时间的文件与目录同步
42     # 我们会默认$2所示的目录不存在
43     # 创建相关文件夹后我们会对文件进行拷贝，对目录进行递归调用操作
44     mkdir $2 > /dev/null 2>&1 # SILENCE
45     [ -z "$(ls $1)" ] && return
46     for file in $1/*
47     do
48         if [ -f $file ]; then
49             # echo "We're copying $file to $2"
50             cp $3a $file $2 # 这里的$3可能是-a或者-u，分别进行更新拷贝或全覆盖拷贝
51         else
52             stripped=${file##*/}
53             # echo "This is where the recursion starts"
54             # echo "The stripped version is: $stripped"
55             echo "[RECURSION] Source: $file, Destination: $2/$stripped"
56             # 我们假设新的文件夹是不存在的（当然若已经存在我们会转移报错信息）
57             mkdir $2/$stripped > /dev/null 2>&1
58             # 我们进行一次全脚本的递归调用，以对子目录进行相同选项下的同步操作
59             DirSync.sh $file $2/$stripped $3 "DUMMY" # 我们传入DUMMY参数以禁用提示符
60         fi
61     done
62 }
63
64 syncContent() {
65     # 我们调用此函数来进行不同文件夹下的同步
66     # 同步的逻辑如文件开头所说，根据目标目录和当前文件的文件的最近修改时间来确定何时删除
67     # 因此这一操作是危险的，若用户的改动较为复杂，我们不推荐使用这种方式
68     [ -z "$(ls $1)" ] && return
69     dir_time=$(stat -c %Y $2) # 我们在进入循环前就计算目标目录的修改时间，因为循环内部的文
件操作可能会导致修改时间发生改变
70     for file in $1/*
71     do
72         stripped=${file##*/}
73         if [ -f $file -o ! -d $2/$stripped ]; then
74             # 对于在目标目录不存在的文件，我们进行按照时间的更新
75             # 若$file是目录，而目标目录下不存在这一子目录，我们将其当作文件处理
76             file_time=$(stat -c %Y $file)
77             # echo "$file is last modified at $file_time"
78             # echo "$2 is last modified at $dir_time"
79             if [ ! -f $2/$stripped ]; then
80                 # echo "Careful now, there's a new file or a file to be deleted"
81                 if [ $file_time -lt $dir_time ]; then
82                     echo "[DELETE] I think the file is ought to be deleted since
it's not newer than the target dir"
83                     rm -rf $file
84                 else
85                     echo "[SYNC] I think we should sync the file since the file
is newer than the target dir"
86                     cp -ua $file $2
87                 fi

```

```

88         else
89             # echo "Whatever, we'll still do a -au copy"
90             cp -ua $file $2
91         fi
92     else
93         # 我们在目标目录下存在$file对应的子目录的情况下进行递归调用
94         echo "[RECURSION] Source: $file, Destination: $2/$stripped"
95         # echo "This is where the syncing recursion starts"
96         DirSync.sh $file $2/$stripped $3 "DUMMY" # 我们传入DUMMY参数以禁用提示符
97     fi
98 done
99 }

100 testDir() {
101     # 检查是否为目录 · 否则退出整个脚本
102     if [ ! -d $1 ]; then
103         echo "[FATAL] $1 is not a directory"
104         exit 1
105     fi
106 }
107 }

108 promptYN() {
109     # 提示用户确认操作
110     # 我们通过调用local确定相关regex
111     message=$1
112     set -- $(locale LC_MESSAGES)
113     yesptrn=$1
114     noptrn=$2
115     # echo $yesptrn
116     # echo $noptrn
117     while true; do
118         read -p "$message(Y/n)? " yn
119         case $yn in
120             ${yesptrn##^} ) return 0;; # 这里返回零表示True
121             ${noptrn##^} ) return 1;; # 这里返回一表示False ( 我们也很奇怪 )
122             * ) echo "Please answer (Y/n).";; # 用户输入的内容有误
123         esac
124     done
125 }

126 }

127 displayHelp()
128 {
129     # 显示帮助信息
130     # 在程序遇到无法识别的flag或识别到-h选项时会打印出相关信息
131     echo -e "\nUsage: DirSync.sh Source Destination [ -a | -r | -u | -s ]"
132     echo -e "\n-a      Perform an appending backup: files not in dir1 will be
133     retained in dir2"
134     echo -e "\n-r      Perform a replacing backup: everything under dir2 will be
135     exact what they were in dir1"
136     echo -e "\n-u      Perform a mutual update: files with same names will be
137     updated to the newest version"
138     echo -e "\n-s      Perform a syncing update: "
139     echo "          We'll decide that a file is newly created if its timestamp is
bigger than the destination dir"
140     echo "          and there's no same-named file in the destination dir."
141     echo "          And we'll decided that the user deleted the file if otherwise (no
same-named and dir newer than file)"

```

```
140     echo -e "\n-h      Display this help page. This option should be used
141     separatedly: ./DirSync.sh -h"
142     echo -e "\nAnd you can pass in the fourth argument to silence the program,
143     not asking for your permission"
144     echo -e "\nExample:\n          DirSync.sh Source Destination -a -y # run the
145     programm right on, performing appending updates from \"Source\" to
146     \"Destination\""
147     echo "          DirSync.sh dir1 dir2 -s                  # run the programm with
148     user confirmation, doing a dangerous syncing between \"dir1\" and \"dir2\""
149 }
150 #####
151 # 程序的主逻辑
152 #####
153
154 if [ $1 = "-h" ]; then
155     # 对于-h选项 · 我们打印帮助信息
156     displayHelp
157     exit 0
158 fi
159
160 testDir $1 # 检查$1是否为有效目录
161
162 if [ "$3" = "-a" -o "${#3}" -eq 0 ]; then
163     # 没有给出选项的情况下默认使用"增加同步"的功能
164     if [ -z $4 ]; then
165         # 注意 · 第四个命令行参数是用于让程序安静运行的 ( 不给出用户确认机会 · 一般用于内部递归调用 )
166         echo "[INFO] Are you trying to back up $1 to $2?"
167         promptYN || exit 0 # 调用promptYN获取用户的答案
168     fi
169     copyContent $1 $2 "-a" # 调用内部赋值函数
170 elif [ $3 = "-r" ]; then # 覆盖型同步
171     if [ -z $4 ]; then # 同上
172         echo "[INFO] Are you trying to do a replace backup from $1 to $2?"
173         echo "All the files originally in $2 will be deleted"
174         promptYN || exit 0
175     fi
176     # 我们会清空原目标目录 · 接着将相关内容原封不动的复制到目标目录
177     rm -rf $2
178     cp -ua $1 $2
179 elif [ $3 = "-u" ]; then
180     if [ -z $4 ]; then
181         # 对于同步型复制 · 我们进行左右复制以保留两个文件夹中都没有的内容
182         testDir $2
183         copyContent $1 $2 $3
184         copyContent $2 $1 $3
185     fi
186     if [ -z $4 ]; then
187         echo "[INFO] This is the syncing part, aha! And it's DANGEROUS TO USE!"
188         echo "[INFO] You'd only want to use this when you've only made DIR CHANGE
189         IN ONE OF THE TWO DIRS"
190         promptYN || exit 0
191     fi
```

```
190      # 我们进行同步，调用方式同上，只不过我们会根据目标文件夹与当前目录的时间戳判断文件为新添加  
191      或将要被删除的  
192          testDir $2  
193          syncContent $1 $2 $3  
194      syncContent $2 $1 $3  
194  else  
195      # 对于未识别的参数，我们打印帮助信息后直接退出  
196      echo "[FATAL] Unrecognized flag, use -a to do appending update, -u to do  
196      mutual update, -r to do full replacement and -s to do sync"  
197      displayHelp  
198      exit 1  
199 fi
```

程序运行结果截图

```
xuzh@ubuntu ~/Projects/ShellDesign/DirSync [master] tree d1 d2
```

```
d1
├── d11
├── d12
│   ├── d12
│   ├── f121
│   ├── f122
│   └── f123
└── d13
    ├── f1
    ├── f2
    ├── f3
    └── f4
d2
├── f5
├── f6
└── f7
```

```
4 directories, 10 files
```

```
xuzh@ubuntu ~/Projects/ShellDesign/DirSync [master] ./DirSync.sh d1 d2 -a
```

```
[INFO] Are you trying to back up d1 to d2?
```

```
(Y/n)? Y
```

```
[RECURSION] Source: d1/d11, Destination: d2/d11
[RECURSION] Source: d1/d12, Destination: d2/d12
[RECURSION] Source: d1/d12/d12, Destination: d2/d12/d12
[RECURSION] Source: d1/d13, Destination: d2/d13
```

```
xuzh@ubuntu ~/Projects/ShellDesign/DirSync [master] tree d1 d2
```

```
d1
├── d11
├── d12
│   ├── d12
│   ├── f121
│   ├── f122
│   └── f123
└── d13
    ├── f1
    ├── f2
    ├── f3
    └── f4
d2
├── d11
├── d12
│   ├── d12
│   ├── f121
│   ├── f122
│   └── f123
└── d13
    ├── f1
    ├── f2
    ├── f3
    ├── f4
    └── f5
        ├── f6
        └── f7
```

```
8 directories, 17 files
```

```
xuzh@ubuntu ~/Projects/ShellDesign/DirSync [master]
```

```
xuzh@ubuntu ~ ~/Projects/ShellDesign/DirSync master ./DirSync.sh d1 d2 -r
```

[INFO] Are you trying to do a replace backup from d1 to d2?

All the files originally in d2 will be deleted

(Y/n)? Y

```
xuzh@ubuntu ~ ~/Projects/ShellDesign/DirSync master tree d1 d2
```

```
d1
└── d11
    └── d12
        ├── d12
        │   ├── f121
        │   ├── f122
        │   └── f123
        └── d13
            ├── f1
            ├── f2
            ├── f3
            └── f4
```

```
d2
└── d11
    └── d12
        ├── d12
        │   ├── f121
        │   ├── f122
        │   └── f123
        └── d13
            ├── f1
            ├── f2
            ├── f3
            └── f4
```

8 directories, 14 files

```
xuzh@ubuntu ~ ~/Projects/ShellDesign/DirSync master
```

↳ xuzh@ubuntu 21 days ago In 28 Col 59 Spacing: 4

```
xuzh@ubuntu ~/Projects/ShellDesign/DirSync master touch d1/f5
xuzh@ubuntu ~/Projects/ShellDesign/DirSync master touch d2/f6
xuzh@ubuntu ~/Projects/ShellDesign/DirSync master tree d1 d2
d1
└── d11
    └── d12
        ├── d12
        ├── f121
        ├── f122
        └── f123
    └── d13
        ├── f1
        ├── f2
        ├── f3
        ├── f4
        └── f5
d2
└── d11
    └── d12
        ├── d12
        ├── f121
        ├── f122
        └── f123
    └── d13
        ├── f1
        ├── f2
        ├── f3
        ├── f4
        └── f6

8 directories, 16 files
xuzh@ubuntu ~/Projects/ShellDesign/DirSync master ./DirSync.sh d1 d2 -u
[INFO] Are you trying to update between d1 and d2?
(Y/n)? Y
[RECURSION] Source: d1/d11, Destination: d2/d11
[RECURSION] Source: d1/d12, Destination: d2/d12
[RECURSION] Source: d1/d12/d12, Destination: d2/d12/d12
[RECURSION] Source: d2/d12/d12, Destination: d1/d12/d12
[RECURSION] Source: d1/d13, Destination: d2/d13
[RECURSION] Source: d2/d11, Destination: d1/d11
[RECURSION] Source: d2/d12, Destination: d1/d12
[RECURSION] Source: d2/d12/d12, Destination: d1/d12/d12
[RECURSION] Source: d1/d12/d12, Destination: d2/d12/d12
[RECURSION] Source: d2/d13, Destination: d1/d13
```

```
xuzh@ubuntu ~ ~/Projects/ShellDesign/DirSync master tree d1 d2
```

```
d1
```

```
  └── d11  
  └── d12  
      ├── d12  
      ├── f121  
      ├── f122  
      └── f123  
  └── d13  
      ├── f1  
      ├── f2  
      ├── f3  
      ├── f4  
      └── f5  
          └── f6
```

```
d2
```

```
  └── d11  
  └── d12  
      ├── d12  
      ├── f121  
      ├── f122  
      └── f123  
  └── d13  
      ├── f1  
      ├── f2  
      ├── f3  
      ├── f4  
      └── f5  
          └── f6
```

```

xuzh@ubuntu > ~/Projects/ShellDesign/DirSync > } master > touch d1/d12/f124
xuzh@ubuntu > ~/Projects/ShellDesign/DirSync > } master > stat d1/d12/f124
  File: d1/d12/f124
  Size: 0          Blocks: 0          IO Block: 4096   regular empty file
Device: 801h/2049d  Inode: 688523      Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/    xuzh)  Gid: ( 1000/    xuzh)
Access: 2020-07-30 23:05:06.753127285 +0800
Modify: 2020-07-30 23:05:06.753127285 +0800
Change: 2020-07-30 23:05:06.753127285 +0800
 Birth: -
xuzh@ubuntu > ~/Projects/ShellDesign/DirSync > } master > stat d1/d12/f121
  File: d1/d12/f121
  Size: 0          Blocks: 0          IO Block: 4096   regular empty file
Device: 801h/2049d  Inode: 688518      Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/    xuzh)  Gid: ( 1000/    xuzh)
Access: 2020-07-30 23:03:13.819488232 +0800
Modify: 2020-07-30 23:03:03.627999687 +0800
Change: 2020-07-30 23:03:03.627999687 +0800
 Birth: -
xuzh@ubuntu > ~/Projects/ShellDesign/DirSync > } master > stat d2/d12
  File: d2/d12
  Size: 4096        Blocks: 8          IO Block: 4096   directory
Device: 801h/2049d  Inode: 810291      Links: 3
Access: (0775/drwxrwxr-x)  Uid: ( 1000/    xuzh)  Gid: ( 1000/    xuzh)
Access: 2020-07-30 23:04:04.956753106 +0800
Modify: 2020-07-30 23:04:46.534347700 +0800
Change: 2020-07-30 23:04:46.534347700 +0800
 Birth: -
xuzh@ubuntu > ~/Projects/ShellDesign/DirSync > } master > ./DirSync.sh d1 d2 -s
[INFO] This is the syncing part, aha! And it's DANGEROUS TO USE!
[INFO] You'd only want to use this when you've only made DIR CHANGE IN ONE OF THE TWO DIRS
(Y/n)? y
[RECURSION] Source: d1/d11, Destination: d2/d11
[RECURSION] Source: d1/d12, Destination: d2/d12
[RECURSION] Source: d1/d12/d12, Destination: d2/d12/d12
[DELETE] I think the file is ought to be deleted since it's not newer than the target dir
[SYNC] I think we should sync the file since the file is newer than the target dir
[RECURSION] Source: d2/d12/d12, Destination: d1/d12/d12
[RECURSION] Source: d1/d13, Destination: d2/d13
[RECURSION] Source: d2/d11, Destination: d1/d11
[RECURSION] Source: d2/d12, Destination: d1/d12
[RECURSION] Source: d2/d12/d12, Destination: d1/d12/d12
[RECURSION] Source: d1/d12/d12, Destination: d2/d12/d12
[RECURSION] Source: d2/d13, Destination: d1/d13
xuzh@ubuntu > ~/Projects/ShellDesign/DirSync > } master > tree d{1,2}
d1
└── d11
   └── d12
      ├── f122
      ├── f123
      └── f124
   └── d13
      ├── f1
      ├── f2
      ├── f3
      ├── f4
      ├── f5
      └── f6
d2
└── d11
   └── d12
      ├── f122
      ├── f123
      └── f124
   └── d13
      ├── f1
      ├── f2
      ├── f3
      ├── f4
      ├── f5
      └── f6
8 directories, 18 files
xuzh@ubuntu > ~/Projects/ShellDesign/DirSync > } master >

```

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

系统中根据不同的权限分为三类用户：管理员、教师、学生，简要说明如下：

1. 管理员：

创建、修改、删除、显示 (list) 教师帐号；教师帐户包括教师工号、教师姓名，教师用户以教师工号登录。

创建、修改、删除课程；绑定（包括添加、删除）课程与教师用户。课程名称以简单的中文或英文命名。

2. 教师：

对某门课程，创建或导入、修改、删除学生帐户，根据学号查找学生帐号；学生帐号的基本信息包括学号和姓名，学生使用学号登录。

发布课程信息。包括新建、编辑、删除、显示 (list) 课程信息等功能。

布置作业或实验。包括新建、编辑、删除、显示 (list) 作业或实验等功能。

查找、打印所有学生的完成作业情况。

3. 学生：

在教师添加学生账户后，学生就可以登录系统，并完成作业和实验。

基本功能：新建、编辑作业或实验功能；查询作业或实验完成情况。

要求：

1. 用bash脚本实现上述基本功能；其它功能可根据实际情况有添加

2. 可以用一个简单的菜单“作业管理系统”界面。

3. **本实验题要求提供以下文档：**

1. 有需求定义或功能描述文档。题中所提供的功能需求非常简单，把这些需求详细化，允许你扩展和改变

2. 设计文档，包括设计思想、功能模块、数据结构、算法等

3. 源程序。详细的注释和良好编程风格

需求描述

设计文档

用bash编写程序，实现一个简单的**作业管理系统**。使用数据库软件包来实现。系统具备以下的基本功能：

- 仿图形界面的页面性逻辑：在Terminal等纯文字终端实现页面逻辑
- 重要信息的高亮显示
- 打印有用错误信息，方便调试
- 与后端数据库系统的交互衔接
- 系统登陆

1. 教师、学生、管理员都通过其ID和密码登陆，需要事先说明自身身份。
2. 系统内部通过 sha256 加密存储和验证密码。
3. 在系统Banner处显示当前用户身份。

系统中根据不同的权限分为三类用户：管理员、教师、学生，简要说明如下：

- 管理员：

1. 管理管理员账户

1. 创建、修改、删除、显示（进入相关界面直接显示）管理员帐号。
 2. 对修改密码操作单独询问。
 3. 教师帐户包括管理员账号、管理员姓名等。
 4. 可以修改自身账号的信息，或删除自身账号，在下一次登陆时体现修改/删除效果。
 2. 管理教师账户
 1. 创建、修改、删除、显示（进入相关界面直接显示）教师帐号。
 2. 对修改密码操作单独询问。
 3. 教师帐户包括教师工号、教师姓名、性别、职称、注册时间、教师简介等。
 3. 管理学生账户
 1. 创建、修改、删除、显示（进入相关界面直接显示）学生帐号。
 2. 对修改密码操作单独询问。
 3. 学生帐户包括学生学号、学生姓名、性别、录取时间、学生简介等。
 4. 管理课程列表
 1. 创建、修改、删除、显示（进入相关界面直接显示）课程；绑定（包括添加、删除）课程与教师用户。
 2. 课程名称以简单的中文和英文命名，课程列表中包括课程号、课程中文名，课程英文名，课程简介等。
- 教师：
 1. 管理课程中的学生列表
 1. 显示修读课程的学生名单。
 2. 对某门课程，导入或删除学生帐户，根据学号查找学生帐号。
 3. 注：所有可登录账户本身只有管理员有权限管理。
 2. 发布课程信息
 1. 管理课程简介
 1. 显示、修改本课程的简介，支持换行。
 2. 管理课程公告
 1. 管理课程公告，包括新建、编辑、删除、显示（进入相关界面直接显示）课程信息等功能，公告内容支持换行。。
 2. 对于课程公告可以选择添加附件。
 3. 显示课程公告的附件情况。
 3. 布置作业或实验
 1. 管理课程作业/实验
 1. 包括新建、编辑、删除、显示（进入相关界面直接显示）作业或实验等功能，添加时可设定作业/实验截止时间。
 2. 作业/实验简介支持换行。
 3. 对于实验/作业可以选择添加附件。
 4. 显示课程实验/作业的附件情况。
 2. 查看作业/实验的完成情况
 1. 显示全部修读学生的完成情况。
 2. 单独查询某个同学的作业完成情况，并查看其所有提交内容。
 - 学生：
 1. 查看自己修读的课程列表
 2. 总体查看作业/实验的完成情况，列举提交的次数等
 3. 管理课程作业的提交
 1. 对已经布置的课程作业/实验新建、编辑、删除、显示（进入相关界面直接显示）提交
 2. 根据设定的作业/实验截止时间判断学生是否真的可以创建/修改/删除提交。

设计思想

数据库交互

我们通过MySQL命令来直接执行数据库操作，这也是本实验的核心内容

您需要有一个 版本号至少为5.7.*的MySQL数据库，并且您需要对其有管理权限

我们通过设置文件的方式使得MySQL不会抱怨直接在命令行输入密码不安全：

```
1 mysql: [Warning] Using a password on the command line interface can be insecure.
```

- 注意：您可以修改程序运行目录下的 `.mysql.cnf` 文件来设置自己的数据库登陆信息
- 第一次使用本软件时请运行当前目录下的 `table.sql` 来初始化数据库中的表

必须运行的部分是所有的 `create table`

后面的 `insert` 内容是可选的，但是至少要有一个管理员账户，否则本软件没有什么意义

样例初始化语句（假设您知道root密码）：`mysql -uroot -p < tables.sql`：此语句会要求您输入root密码

- 请保证MySQL已经在本机正确安装，且 `.mysql.cnf` 已经被正确配置

您需要在 `.mysql.cnf` 中设置您的登录名/密码/服务器，并设置数据库名称(和您在MySQL中使用的相同)

例如您在MySQL中创建了 `ShellDesigner` 这个用户，密码为 `ShellDesigner`，并打算使用 `ShellDesign` 这个数据库来管理本软件涉及到的内容

登陆root用户后，可使用如下操作修改密码

```
1 ALTER USER 'user'@'hostname' IDENTIFIED BY 'newPass';
```

可以通过如下操作创建新用户

```
1 create user ShellDesigner identified by 'ShellDesigner';
2 create database ShellDesign;
3 grant all on ShellDesign.* to ShellDesigner;
```

`.mysql.cnf` 就将有类似如下的内容

```
1 [client]
2 user=ShellDesigner
3 password=ShellDesigner
4 host=localhost
5 database=ShellDesign
```

下列是我们默认的一些设置

```
1 mysql_u_default="ShellDesigner"
2 mysql_p_default="ShellDesigner"
3 mysql_h_default="localhost"
4 mysql_d_default="ShellDesign"
5 mysql_f=".mysql.cnf"
6
7 # 类似调用alias，我们在下面的Shell语句中执行MySQL调用时都会使用$mysql_prefix来开头
8 mysql_prefix="mysql --defaults-extra-file=$mysql_f"
```

我们采用了命令行调用MySQL数据库的方式实现此管理系统的主要功能。

为了方便复用和嵌套，我们将所有的SQL查询语句存储在字符串变量中（容易遭到SQL Injection攻击，后面会提到如何防御）

注意在每一次事件循环后我们都会尽量更新一次查询语句的变量内容（除非此语句是固定的）。

```
1 query_id="select cid from take where sid=$sid"
2 query_course="select id 课程号, name_zh 中文名称, name_en 英文名称 from course where id
in ($query_id)"
```

第一层括号将返回结果当作数组处理，第二层 `$()` 是执行了一个Bash语句，在此是执行了一个MySQL查询

- 在本程序中，我们将结果存入变量时基本都会采用这种调用MySQL的方式，我们会使用 `-se` 选项，其中 `-e` 代表执行，`-s --slient`，安静模式，在此的效果是去除列名
- 在直接执行MySQL并原封不动的打印信息时，我们会使用`-e`选项，代表执行

值得注意的是，在命令行直接调用MySQL时，会打印列分隔符，而将结果存入变量则不会打印（列分隔符自动会得到删除）

```
1 # 重定向标准输出到文件并打印文件
2 xuzh@ubuntu ~ /Projects/ShellDesign master ● mysql -uShellDesigner -pShellDesigner ShellDesign -e "select * from admin;" > temp.txt; cat temp.txt
3 mysql: [Warning] Using a password on the command line interface can be insecure.
4 name      id      password_hash
5 root      1      53175bcc0524f37b47062fafdda28e3f8eb91d519ca0a184ca71bbebe72f969a
6 admin     2      fc8252c8dc55839967c58b9ad755a59b61b67c13227ddae4bd3f78a38bf394f7
7
8 # 直接执行语句，打印到标准输出
9 xuzh@ubuntu ~ /Projects/ShellDesign master ● mysql -uShellDesigner -pShellDesigner ShellDesign -e "select * from admin;" 
10 mysql: [Warning] Using a password on the command line interface can be insecure.
11 +-----+
12 | name   | id    | password_hash |
13 +-----+
14 | root   | 1    | 53175bcc0524f37b47062fafdda28e3f8eb91d519ca0a184ca71bbebe72f969a |
15 | admin  | 2    | fc8252c8dc55839967c58b9ad755a59b61b67c13227ddae4bd3f78a38bf394f7 |
16 +-----+
17
18 # 将标准输出重定向到Terminal标准输出
19 xuzh@ubuntu ~ /Projects/ShellDesign master ● mysql -uShellDesigner -pShellDesigner ShellDesign -e "select * from admin;" > /dev/tty
20 mysql: [Warning] Using a password on the command line interface can be insecure.
21 +-----+
22 | name   | id    | password_hash |
23 +-----+
24 | root   | 1    | 53175bcc0524f37b47062fafdda28e3f8eb91d519ca0a184ca71bbebe72f969a |
25 | admin  | 2    | fc8252c8dc55839967c58b9ad755a59b61b67c13227ddae4bd3f78a38bf394f7 |
26 +-----+
27
28 # 重定向到变量并打印标准输出
29 xuzh@ubuntu ~ /Projects/ShellDesign master ● temp=$(mysql -uShellDesigner -pShellDesigner ShellDesign -e "select * from admin;");echo "$temp"
30 mysql: [Warning] Using a password on the command line interface can be insecure.
31 name      id      password_hash
32 root      1      53175bcc0524f37b47062fafdda28e3f8eb91d519ca0a184ca71bbebe72f969a
33 admin     2      fc8252c8dc55839967c58b9ad755a59b61b67c13227ddae4bd3f78a38bf394f7
```

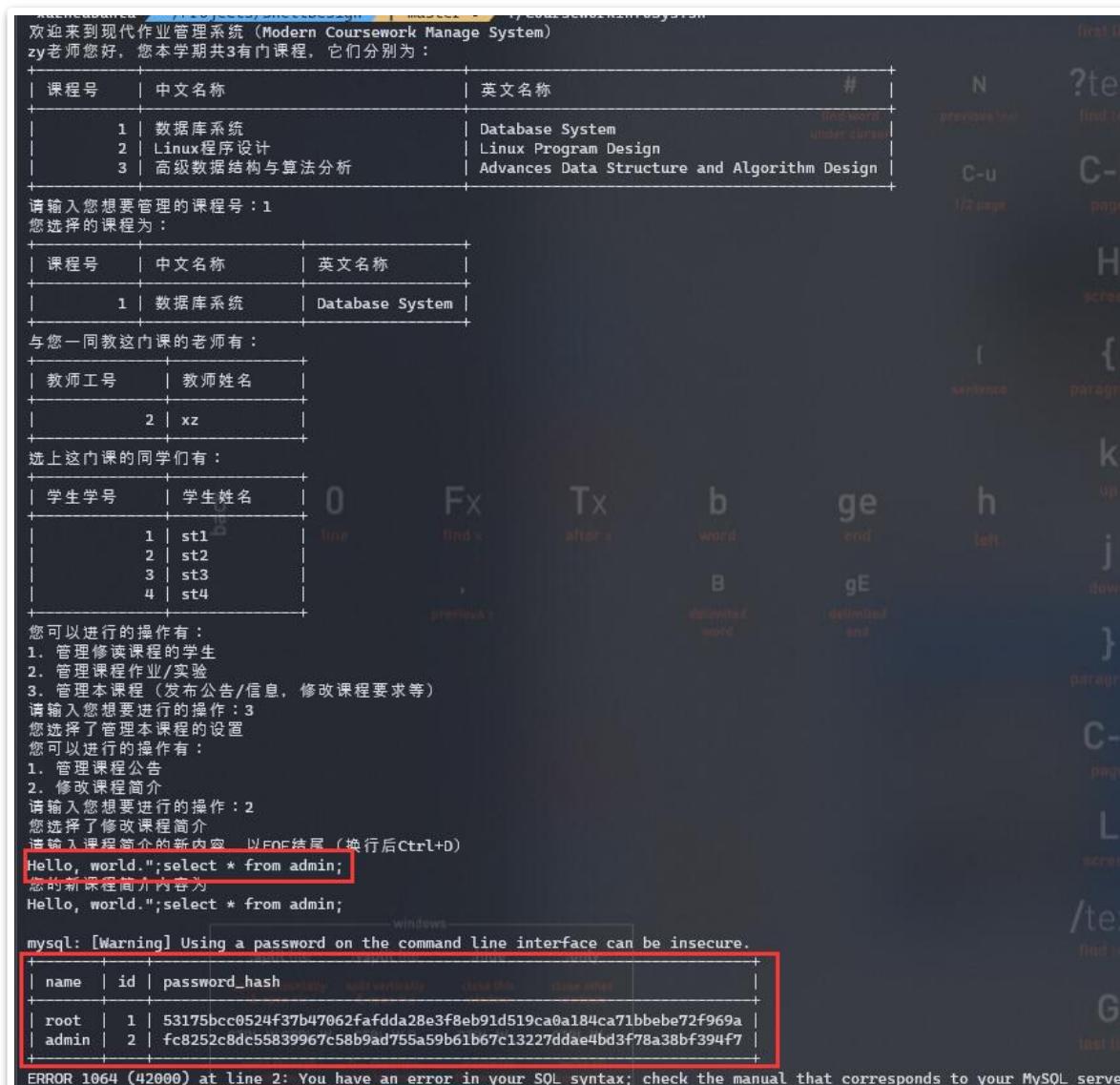
因此当我们想要让此变量获取打印的信息时，我们应直接将返回信息赋值到变量中，当我们想直接使用MySQL的格式化功能时，我们应直接使用命令将输出导入到 `/dev/tty`。

调用举例：

```
1 # 通过变量$mysql_prefix中定义的格式执行MySQL查询：mysql --defaults-extra-file=$mysql_f
2 # 查询的内容是$query_id变量中存储的查询：select cid from take where sid=$sid
3 # 通过-s参数去除列名
4 # 并且将查询结果以数组形式存储到$cids变量中
5 cids=($(mysql_prefix -se "$query_id;"))
6
7 # 直接调用$mysql_prefix变量中定义的内容：mysql --defaults-extra-file=$mysql_f
8 # 查询的内容是$query_course变量中存储的查询：select id 课程号, name_zh 中文名称, name_en
9 英文名称 from course where id in ($query_id)
10 # 不进行输入输出的引导，直接打印到屏幕
11 # 直接调用MySQL并输出到/dev/tty可以使MySQL用分割线打印各种信息
$mysql_prefix -e "$query_course;"
```

同时，为了防止SQL注入攻击，我们设计了如下字符串来过滤敏感字符

不进行过滤直接运行就有可能遭到SQL注入攻击，泄露重要密码HASH值



这一函数会手动转义要插入到SQL命令中的字符串，使得MySQL可以正确解释被转义了的危险字符

```
1 function RemoveDanger() {
2     danger_set=${2:-"[\\\"\\`\\*\\%]"}
```

```

3      danger=$1
4      safe=""
5      for i in $(seq ${#danger}); do
6          thechar="${danger:$i-1:1}"
7          if [[ "$thechar" =~ $danger_set ]]; then
8              # echo "$thechar"
9              safe="$safe""\\\"$thechar"
10         else
11             safe="$safe$thechar"
12         fi
13     done
14     echo "$safe"
15 }

```

桐言的，出于安全性考虑，我们没有在数据库中明文存放密码，而是使用了加密用的 `sha256 hash`

我们将用户密码进行`sha256 hash`后储存

并在登陆时将用户输入的内容进行`sha256 hash`，与数据库内部的`hash`值进行比较，若相等则认为密码正确

- 这种方式可以提高系统的安全性

即使数据库内容被泄露，`sha256`的加密也会让数据偷盗者很难猜出正确的密码

一个解释相关操作的视频

页面逻辑

我们通过一些页面循环来搭建页面逻辑

一个完整页面的结构如下所示

The diagram illustrates the structure of a web page with the following components:

- 横幅: 表明使用者身份** (Header: Indicate user identity) - Points to the top banner area.
- 表格: 一些供参考的列举信息** (Table: Some reference listing information) - Points to a table section containing course assignment/experiment details.
- 您选择了修改以下的课程作业/实验：** (You selected to modify the following course assignments/experiments) - Points to a table showing assignment details.

作业/实验ID	作业/实验简介	创建时间	截止时间
5	实验4 JDBC系统的编写和使用	2020-07-18 16:57:12	2020-07-25 16:57:12

- 本课程作业/实验的附件包括：** (Attachments for this course assignment/experiment include) - Points to a table showing attachments.

附件ID	附件名称	附件URL
2	数据库系统实验报告	https://raw.githubusercontent.com/dendenxu/minisQL/master/xz.tex
4	JDBC接口调用参考与举例	https://raw.githubusercontent.com/dendenxu/DeepOthello/master/MCTS.py

- 您在本课程作业/实验创建的课程作业/实验提交如下所示** (The course assignments/experiments submitted in this course are as follows) - Points to a table showing submitted assignments.

提交ID	提交内容	创建时间	最近修改时间
8	内容呢共软容	2020-07-20 15:54:01	2020-07-20 15:54:01

- 您可以进行的操作有：** (Operations you can perform are) - Points to a list of operations.

 - 发布新的课程作业/实验提交
 - 删除已发布的课程作业/实验提交
 - 修改已发布的课程作业/实验提交
 - 查看已发布的课程作业/实验提交
 - 返回上一级

请输入您想要进行的操作： (Please enter the operation you want to perform:)

- 操作栏: 列举可进行的操作, 最后一项为返回上一级** (Operation bar: List of operations you can perform, the last item is 'Return to previous level') - Points to the bottom right corner of the page.

我们通过类似如下结构的代码来构建上述的事件循环页面：

```

1   function TeacherOPCourse() {

```

```
2     while :; do # 课程操作UI主循环
3         #####
4         #
5         #          打印Banner      #
6         #
7         #####
8         PrintTeacher # 打印Banner
9
10        target="${Green}课程${NoColor}" # 此时的目标字符串为：课程，用绿色显示以方便辨认
11
12        #####
13        #
14        #          打印查询信息      #
15        #
16        #
17        #####
18        query_tid="select tid from teach where cid=$cid"
19        query_teacher="select id 教师工号, name 教师姓名, if(gender='F', \"女\", \"男
\") 性别, registration_time 注册时间, title 职称, brief 简介 from teacher where id in
($query_tid)"
20
21        echo "您选择的${target}为："
22
23        # 此时我们打印课程简介信息，方便用户在后续使用过程中决定是否要修改课程简介信息
24        $mysql_prefix -e "select id 课程号, name_zh 中文名称, name_en 英文名称, brief
课程简介 from course where id=$cid;"
25
26        # 打印除了当前老师外一同教这门课的老师一共用户参考
27        tids=(${mysql_prefix -e "$query_tid and tid > $tid;"})
28        if [ ${#tids[@]} -gt 0 ]; then
29            echo "与您一同教这门课的老师有："
30            $mysql_prefix -e "$query_teacher and id > $tid;"
31        else
32            echo "这门${target}只有您自己在教"
33        fi
34
35        #####
36        #
37        #          操作栏循环      #
38        #
39        #####
40        echo "您可以进行的操作有："
41        echo "1. 管理修读${target}的学生"
42        echo "2. 管理${target}作业/实验"
43        echo "3. 管理本${target}信息（管理公告/简介等）"
44        echo "0. ${ReturnPrev}"
45        while :; do
46            # 输入处理循环，这里比较tidy，因为我们将三个子操作都封装成了函数
47            # 且这里无论选择哪种操作都没有直接清屏返回的必要
48            read -rp "请输入您想要进行的操作：" op
49            case $op in
50                1)
51                    echo "您选择了管理修读该${target}的学生"
52                    TeacherManageStudent
53                    break
54                    ;;
55                2)
56                    echo "您选择了管理本${target}的实验和作业"
```

```

57             TeacherManageHomework
58         break
59     ;;
60     3)
61         echo "您选择了管理本${target}的公告/信息"
62         TeacherManageCourse
63         break
64     ;;
65     0)
66         echo "您选择了${ReturnPrev}"
67         return 0
68     ;;
69     *)
70         echo "您输入的操作$op有误，请输入上面列出的操作"
71     ;;
72     esac
73     done
74 done
75 }

```

我们通过每次主循环都调用这样的一个函数来清空屏幕：

其中的 `clear` 功能可以清空当前页面已打印的信息

接着我们通过 `cat` 打印ASCII ART

```

1 function PrintTeacher() {
2     # TEACHER分隔符，会在老师登陆后的管理界面打印
3     clear
4     cat <<"EOF"
5     -----
6     /\_ _\ /\ ___\ /\ _ \ /\ ___\ /\ \\\ \ /\ ___\ /\ _= \
7     \/_/\ \/\ \ \_ \ \ \_ \ \ \ \ \_ \ \ \ \ \_ \ \ \ \ \_ \ \ \ \ \_ \
8     \ \ \ \ \ \ \ \ \_ \ \ \ \ \ \ \ \ \_ \ \ \ \ \ \ \ \ \_ \ \ \ \ \ \ \ \ \
9     \/_/ \/_/ \/_/ \/_/ \/_/ \/_/ \/_/ \/_/ \/_/ \/_/ \/_/ \/_/ \/_/
10
11 EOF
12 }

```

主循环保证了下一级函数返回后仍然会留在当前页面征求用户的意见

且同时屏幕上的信息会刷新

```

1 while :; do # 屏幕主循环
2
3     # 打印表格内容
4     $mysql_prefix ...
5
6     # 打印可用操作内容
7     echo "... "
8     while :; do # 操作循环
9         case $choice in
10             1)
11                 # 用户输入了正确的选项
12                 # 在此执行下一步的功能
13                 ...
14                 # 运行结束后重新开始主循环，刷新数据
15             break
16             ;;

```

```

17
18     0)
19         # 通过return命令直接返回上一级函数调用/或退出运行
20         return 0
21     ;;
22
23     *)
24         # 用户输入有误
25         # 不调用break命令直接进行操作循环
26         ;;
27     done
28 done

```

页面交互

我们通过设置颜色，字体，以及精心调教read函数和嵌套循环，构成了一套较为流畅的UI导航交互逻辑

- 通过调用 `tput` 命令我们会将重要信息高亮显示，加快用户的定位过程

我们通过初始化这样的语句来定义颜色命令，以后只需要调用相关变量就可以完成颜色的改变

```

1 Red=$(tput setaf 1)
2 Green=$(tput setaf 2)
3 Yellow=$(tput setaf 3)
4 Blue=$(tput setaf 4)
5 Magenta=$(tput setaf 5)
6 Cyan=$(tput setaf 6)
7 Bold=$(tput bold)
8 NoColor=$(tput sgr0)

```

使用样例：

```

1 # 这些变量打印出来都是有颜色或重量的
2 # 每次刷新页面时都要清空目标变量
3 target="${Green}${Bold}课程实验/作业${NoColor}"
4 # 内容未发布提示信息
5 no_publication="${Red}本课程还没有已发布的${NoColor}${target}"
6
7 echo "Target is $target"
8 echo "No publication infomation is $no_publication"

```

```

xuzh@ubuntu:~/Projects/ShellDesign$ master> bash
xuzh@ubuntu:~/Projects/ShellDesign$ Red=$(tput setaf 1)
xuzh@ubuntu:~/Projects/ShellDesign$ Green=$(tput setaf 2)
xuzh@ubuntu:~/Projects/ShellDesign$ Yellow=$(tput setaf 3)
xuzh@ubuntu:~/Projects/ShellDesign$ Blue=$(tput setaf 4)
xuzh@ubuntu:~/Projects/ShellDesign$ Magenta=$(tput setaf 5)
xuzh@ubuntu:~/Projects/ShellDesign$ Cyan=$(tput setaf 6)
xuzh@ubuntu:~/Projects/ShellDesign$ Bold=$(tput bold)
xuzh@ubuntu:~/Projects/ShellDesign$ NoColor=$(tput sgr0)
xuzh@ubuntu:~/Projects/ShellDesign$ # 这些变量打印出来都是有颜色或重量的
xuzh@ubuntu:~/Projects/ShellDesign$ # 每次刷新页面时都要清空目标变量
xuzh@ubuntu:~/Projects/ShellDesign$ target="${Green}${Bold}课程实验/作业${NoColor}"
xuzh@ubuntu:~/Projects/ShellDesign$ # 内容未发布提示信息
xuzh@ubuntu:~/Projects/ShellDesign$ no_publication="${Red}本课程还没有已发布的${NoColor}${target}"
xuzh@ubuntu:~/Projects/ShellDesign$ echo "Target is $target"
Target is 课程实验/作业
xuzh@ubuntu:~/Projects/ShellDesign$ echo "No publication infomation is $no_publication"
No publication infomation is 本课程还没有已发布的课程实验/作业
xuzh@ubuntu:~/Projects/ShellDesign$ 

```

- 通过嵌套循环，我们让用户有很多试错机会

```

1  while :; do
2      read -rp "请输入您想要管理的课程号：" cid
3
4      # 注意到我们使用正则表达式展开数组来进行元素检查
5      # 因此表达式右侧的值应用引号括起以保证完全匹配
6      # 我们使用了ShellCheck工具，而此工具会对=~右侧的表达式报错，因此我们使用了
7      # shellcheck disable=SC2076
8      # 来关闭这一报错
9      [[ "${cids[*]}" =~ "${cid}" ]] && break
10     echo "您输入的课程号$cid有误，请输入上表中列举出的某个课程号"
11 done

```

上一部分描述的嵌套循环也是一个例子。

- 通过调教 `read` 命令，我们给了用户在清屏前观察屏幕的机会，配合高亮，可以快速定位操作中的错误

```

1  function ContinueWithKey() {
2      # 按任意键继续 ...
3      # 有的时候我们会在清空屏幕之前打印一些信息，我们决定给用户一些时间来看看这些信息是什么
4      read -n 1 -rp "${Blue}${Bold}按任意键继续 ... ${NoColor}" -s
5  }

```



即使换成不同的终端，显示效果依然不错。



- 通过调教 `read` 命令，我们使得用户的明文密码不会得到显示

同时，我们会对错误的登录请求添加1s的超时惩罚，以防止暴力破解密码的操作

```

1 while :; do
2     # todo: 使用cat命令可以清楚密码变量，提高安全性，但是我们还没发现该如何换行
3     # 所以暂时使用了变量来存储密码
4     read -rp "请输入您的密码：" -s password
5     echo ""
6     password_hash=$(echo "$password" | sha256sum - | tr -d " ")
7     echo "验证中....."
8     [ "$password_hash" = "$right_hash" ] && break
9     sleep 1s # 为了防止暴力登录攻击，每次密码错误都要得到1s的时间惩罚
10    echo "验证失败，请重新输入"
11 done

```

- 通过嵌套循环，我们使得用户无需提前输入一些稍显冗余的数量信息

例如在附件添加的过程中，用户无需实现输入要添加的附件数目

```

1 # 这里我们通过Bash内部计数来减少一次MySQL链接
2 attachment_count=0
3 while :; do
4     # 我们根据用户回答来修改程序运行流程
5     # 用户无需提前知道需要添加的附件数量
6     # 他/她只需要不断输入Y并添加内容
7     read -rp "请输入您是否需要为${target}添加附件 ( Y/n ) :" need_attach
8     if [[ $need_attach =~ ^[1Yy] ]]; then # 正则表达式匹配
9         attachment_count+=1
10
11     echo "您选择了添加附件"
12     read -rp "请输入您想要添加的附件名称：" attach_name
13     attach_name=$(RemoveDanger "$attach_name") # 可能包含危险字符
14     echo "您的附件名称为：$attach_name"
15
16     read -rp "请输入您想要添加的附件URL：" attach_url
17     # 对于URL，我们使用不同的转义策略
18     attach_url=$(RemoveDanger "$attach_url" "[\\\"\\.\\\\*;]")
19     echo "您的附件URL为：$attach_url"
20
21 # 添加附件到附件相关表，并修改attach_to表来对应附件和Content的关系

```

```

22          # 我们暂时只使用了attach_to表格的一部分功能，在日后的开发中我们可以将一个附件分
配给多个不同的Content
23          # todo: 可以重用已经上传过的附件，建立多对多的附加/带附件内容的对应
24          query_insert_attach="insert into attachment(name, url) value
25          ("$attach_name", "$attach_url")"
26          query_insert_attach_to="insert into attach_to(aid, uid) value
27          (last_insert_id(), $subid)"
28
29          # 同样的，我们利用了Transaction功能
30          attach_id=$(mysql_prefix -se "set
31          autocommit=0;$query_insert_attach;select
32          last_insert_id();$query_insert_attach_to;commit;set autocommit=1;" )
33
34      echo "您刚刚添加的附件ID为：$attach_id"
35      else
36          break
37      fi
38  done

```

- 通过使用ASCII ART，我们让用户很容易的认识到自己的身份

上面打印的STUDENT和TEACHER BANNER就是一个例子

我们还在程序的登陆界面打印了CourseworkManager的字样以方便辨识

```

1  # 以下几个Print函数都是用于打印ASCII Art的
2  # 同时，它们通过调用clear函数来进行假GUI的页面模拟功能
3  # 我们使用ASCII Art的初衷是让用户能在程序的不同Section中更快的找到自己想要的信息
4  # 后来我们发现通过调用clear函数可以达到模拟GUI的功能
5  function PrintBanner() {
6      # 程序的主横幅：CourseworkManger
7      # 会在初始登陆界面打印
8      clear
9      cat <<"EOF"
10     -----
11     -----
12     \_   ___ \   ___  -  -----  -----  -----  -  ----- |  | _  /
13     \ \_  -----  -----  -----  -----  /  \_  /  \_  /  \_  /  \_  /  \_  /  /
14     \ /  \ \_ \ /  \ \_ \ /  \ \_ \ /  \ \_ \ /  \ \_ \ /  \ \_ \ /  \ \_ \ /  \ \_ \ /  /
15     \ \_ \ \_ \ \_ \ \_ \ \_ \ \_ \ \_ \ \_ \ \_ \ \_ \ \_ \ \_ \ \_ \ \_ \ \_ \ \_ \ \_ \ \_ \ \_ \ \_ \ \_ \
16
17  EOF
18 }

```

- 通过清屏功能，我们避免打印太多冗余信息，并模拟了GUI式的交互性操作

功能模块

数据库定义

为了方便用户，我们定义了一个SQL脚本文件，用于快速初始化用户的数据库。

用户在初次运行程序之前可以通过如下的脚本设置数据库（假设您知道root密码）：

```
1 mysql -uroot -p < tables.sql
```

- 用户/数据库定义部分

```
1 # note: we should define the default charset of the database before creating
2 # the tables without explicitly
3 # defining charset
4
5 drop database if exists ShellDesign;
6 drop user if exists ShellDesigner;
7 create user ShellDesigner identified by 'ShellDesigner';
8 create database ShellDesign;
9 grant all on ShellDesign.* to ShellDesigner;
10 alter database ShellDesign character set utf8mb4 collate utf8mb4_unicode_ci;
11 use ShellDesign;
12
13 drop table if exists `take`;
14 drop table if exists `info`;
15 drop table if exists `teach`;
16 drop table if exists `attach_to`;
17 drop table if exists `attachment`;
18 drop table if exists `submission`;
19 drop table if exists `homework`;
20 drop table if exists `content`;
21 drop table if exists `teacher`;
22 drop table if exists `student`;
23 drop table if exists `admin`;
24 drop table if exists `course`;
```

- 表建立部分

```
1 create table `teacher`
2 (
3     name          varchar(100),
4     id            bigint primary key auto_increment,
5     brief         varchar(2000),
6     gender        enum ('F', 'M') default 'F', # F for female and M for
7     male
8     registration_time datetime,
9     title         varchar(500)    default 'Professor',
10    password_hash varchar(64)
11 );
12
13 create table `student`
14 (
15     name          varchar(100),
16     id            bigint primary key auto_increment,
17     brief         varchar(2000),
18     gender        enum ('F', 'M') default 'F', # F for female and M for male
19     enroll_time   datetime,
20     password_hash char(64)
21 );
22
23 create table `admin`
24 (
25     name          varchar(100),
26     id            bigint primary key auto_increment,
27     password_hash char(64)
```

```
27 );
28
29 create table `course`
30 (
31     name_zh  varchar(100),
32     name_en  varchar(100),
33     brief    varchar(2000),
34     syllabus varchar(4000),
35     id       bigint primary key auto_increment
36 );
37
38 create table `teach`
39 (
40     tid bigint,
41     cid bigint,
42     foreign key (`tid`) references teacher (`id`) on delete cascade on update
43     cascade,
44     foreign key (`cid`) references course (`id`) on delete cascade on update
45     cascade
46 );
47
48 create table `take`
49 (
50     cid bigint,
51     sid bigint,
52     foreign key (`sid`) references student (`id`) on delete cascade on update
53     cascade,
54     foreign key (`cid`) references course (`id`) on delete cascade on update
55     cascade
56 );
57
58 # this is a dummy class so that we can ensure foreign key references from
59 # attachments to both submissions and homework
60 create table `content`
61 (
62     id bigint primary key auto_increment
63 );
64
65 create table `info`
66 (
67     id      bigint primary key,
68     content varchar(2000),
69     cid     bigint,
70     release_time datetime,
71     foreign key (`cid`) references course (`id`) on delete cascade on update
72     cascade,
73     foreign key (`id`) references content (`id`) on delete cascade on update
74     cascade
75 );
76
77 create table `homework`
```

```

78     type          enum ('H', 'E') default 'H', # H for homework and e for
    experiment
79     foreign key (`id`) references content (`id`) on delete cascade on update
    cascade,
80     foreign key (`tid`) references teacher (`id`) on delete cascade on update
    cascade,
81     foreign key (`cid`) references course (`id`) on delete cascade on update
    cascade
82 );
83
84 create table `submission`
85 (
86     id          bigint primary key auto_increment,
87     sid         bigint,
88     hid         bigint,
89     submission_text  varchar(2000),
90     creation_time   datetime,
91     latest_modification_time  datetime,
92     foreign key (`id`) references content (`id`) on delete cascade on update
    cascade,
93     foreign key (`sid`) references student (`id`) on delete cascade on update
    cascade,
94     foreign key (`hid`) references homework (`id`) on delete cascade on
    update cascade
95 );
96
97 create table `attachment`
98 (
99     id      bigint primary key auto_increment,
100    name    varchar(100),
101    url     varchar(800),
102    brief   varchar(2000)
103 );
104
105 create table `attach_to`
106 (
107     aid  bigint,
108     uid  bigint,
109     foreign key (`aid`) references attachment (`id`) on delete cascade on
    update cascade,
110     foreign key (`uid`) references content (`id`) on delete cascade on update
    cascade
111 );

```

- Dummy内容插入部分

```

1  insert into `course`(`id, name_zh, name_en`)
2  values (1, '数据库系统', 'Database System'),
3      (2, 'Linux程序设计', 'Linux Program Design'),
4      (3, '高级数据结构与算法分析', 'Advances Data Structure and Algorithm
Design'),
5      (4, '计算机图形学', 'Computer Graphics'),
6      (5, '视觉识别中的深度卷积神经网络', 'Convolutional Neural Network for
Visual Recognition'),
7      (6, 'iOS开发', 'iOS Software Development');
8
9  insert into `teacher`(`id, name, password_hash, registration_time)

```

```

10    values (1, 'zy',
11              '49aabdaa1b0f6c3506f54521ef81fe5b5fe835d268f1f86e1021a342b59d43bc', now()), #
12      password is zy
13      (2, 'xz',
14              'b44f7d6b5283a44ee5f2bd98f84087a04810092122d75e8fbf8ad85f8f2981f1', now()); #
15      password is xz
16
17  insert into `admin`(id, name, password_hash)
18  values (1, 'root',
19          '53175bcc0524f37b47062fafdda28e3f8eb91d519ca0a184ca71bbebe72f969a'), #
20      password is root
21      (2, 'admin',
22          'fc8252c8dc55839967c58b9ad755a59b61b67c13227ddae4bd3f78a38bf394f7'); #
23      password is admin
24
25  insert into `student`(id, name, password_hash, enroll_time)
26  values (1, 'st1',
27          '2238ead9c048f351712c34d22b41f6eec218ea9a9e03e48fad829986b0dafc11', now()), #
28      password is same as name
29      (2, 'st2',
30          '5e61d026a7889d9fc72e17f1b25f4d6d48bfe17046fea845aa8c5651ec89c333', now()),
31      (3, 'st3',
32          'bbb977f8e93feb5dbd79e0688b822115b5acf774dd8a1fe6964e03d6b9579384', now()),
33      (4, 'st4',
34          '6133396ebcd382b137088d2ea91d60637744e404b4376e4635b45784b718db72', now()),
35      (5, 'st5',
36          'd691a62aa63f1be970582902d0ff78df29899f09c5dd540b1447cdd051dcfc8d', now()),
37      (6, 'st6',
38          'a7a287ffc9cb27131b9dc54199ba96cef87e753968bc620d714af212ef0f7a8c', now()),
39      (7, 'st7',
40          '73d0daf13c6159a1fbdeb37b6972325b6e29c312371a0f3d427bd35c0c87b928', now()),
41      (8, 'st8',
42          '4ce70fc1eef7303879a2ef33996db2f85058ae06e8590521267ae8d46ec59793', now());
43
44  insert into `teach`(cid, tid)
45  values (1, 1),
46          (1, 2),
47          (2, 1),
48          (3, 1),
49          (4, 2),
50          (5, 2);
51
52  insert into `take`(cid, sid)
53  values (1, 1),
54          (1, 2),
55          (1, 3),
56          (1, 4),
57          (2, 3),
58          (2, 4),
59          (2, 5),
60          (2, 6),
61          (3, 7),
62          (3, 8),
63          (4, 1),
64          (4, 3),
65          (4, 5),
66          (5, 2),
67          (5, 4),
68
69
```

```
51      (5, 6),  
52      (5, 8),  
53      (6, 1),  
54      (6, 7),  
55      (6, 8);
```

```
insert into content(id)  
values (1),  
       (2),  
       (3),  
       (4),  
       (5),  
       (6),  
       (7);  
  
insert into homework(id, cid, tid, intro, creation_time, end_time, type)  
values (5, 1, 1, '实验4 JDBC系统的编写和使用', now(), now() + interval 7 day, 'E'),  
       (6, 1, 1, '第五周数据库系统作业', now(), now() + interval 10 day, 'H'),  
       (7, 1, 2, '课程大作业 MiniSQL的编写与使用', now(), now() + interval 20 day, 'H');  
  
insert into attachment(id, name, url)  
values (1, 'Linux Shell Program Design 3rd Edition.pdf',  
       'https://raw.githubusercontent.com/dendenxu/minisQL/master/minisQL.tex'),  
       (2, '数据库系统实验报告', 'https://raw.githubusercontent.com/dendenxu/minisQL/master/xz.tex'),  
       (3, '蒙特卡洛树搜索实现', 'https://raw.githubusercontent.com/dendenxu/DeepOthello/master/MCTS.py'),  
       (4, 'JDBC接口调用参考与举例', 'https://raw.githubusercontent.com/dendenxu/DeepOthello/master/MCTS.py');  
  
insert into info(id, content, cid, release_time)  
values (1, '作业1的提交就要截止啦！请大家及时关注。', 1, NOW()),  
       (2, '实验5的验收将在本周六下午4点开始，请需要验收的组长搜索“数据库系统”钉钉群并加入，钉  
钉群二维码详见附件', 1, NOW()),  
       (3, 'ADS考试将在6月24日以线上/机房同时考试的形式进行，YDS老师的复习视频已上传到学在浙  
大系统，详见附件', 3, NOW()),  
       (4, '明天的实验内容为样条插值（Spline）以及贝塞尔曲线的拟合（Bezier Path），请同学们提前  
预习相关内容，PPT已上传附件并开放下载', 4, NOW());  
  
insert into attach_to(aid, uid)  
values (1, 1),  
       (1, 2),  
       (1, 3),  
       (2, 1),  
       (2, 5),  
       (2, 6),  
       (4, 5),  
       (3, 1);
```

```
1 ##### 初始化模块  
2  
3 我们设计了两个初始化函数，用以定义一些在程序运行过程中全局使用的变量：  
4  
5 - 颜色变量，用以打印有色UI  
6  
7
```

```

8      ```bash
9      function DefineColor() {
10         # 我们使用tput命令来定义颜色信息
11         # 各类颜色常数，通过echo调用可以改变Shell的输出样式
12         # 例如echo "${Red}Hello${NoColor}"，world."会打印红色的Hello和原色的World
13         # 上述例子会展开成echo "$tput setaf 1>Hello$(tput sgr0)，world."
14         # ! consider more about this colorization
15         Red=$(tput setaf 1)
16         Green=$(tput setaf 2)
17         Yellow=$(tput setaf 3)
18         Blue=$(tput setaf 4)
19         Magenta=$(tput setaf 5)
20         Cyan=$(tput setaf 6)
21         Bold=$(tput bold)
22         NoColor=$(tput sgr0)
23         ReturnPrev="${Yellow}${Bold}返回上一级${NoColor}"
24     }

```

- 数据库变量，用以操作MySQL
 - 数据库操作变量，用以通过CMD调用MySQL
 - 数据库登陆定义，一些用户，密码等的提前设置

```

1  function DefineMySQL() {
2      # 下列是我们默认的一些设置
3      mysql_u_default="ShellDesigner"
4      mysql_p_default="ShellDesigner"
5      mysql_h_default="localhost"
6      mysql_d_default="ShellDesign"
7      mysql_f=".mysql.cnf"
8
9      # 若.mysql.cnf在当前目录不存在，我们会创建一个并将默认内容写入
10     if [ ! -f "$mysql_f" ]; then
11         echo "Automatically generating configuration file..." >&2
12         echo "[client]" >$mysql_f
13         echo "user=$mysql_u_default" >>$mysql_f
14         echo "password=$mysql_p_default" >>$mysql_f
15         echo "host=$mysql_h_default" >>$mysql_f
16         echo "database=$mysql_d_default" >>$mysql_f
17     fi
18
19     # 类似调用alias，我们在下面的Shell语句中执行MySQL调用时都会使用$mysql_prefix来开头
20     mysql_prefix="mysql --defaults-extra-file=$mysql_f"
21 }
22

```

登陆模块

正如前面描述的，我们在登陆模块采用了一些防范攻击的方法：

- 去除可能造成SQL注入的危险字符
- 登陆失败的操作会受到1s的惩罚时间
- 每次登陆至少等待100ms防止攻击
- 密码不使用明文显示
- 数据库中用sha256sum储存和验证密码

```
1  # 初始界面登陆逻辑
```

```

2   function LoginInUI() {
3       while :; do
4           PrintBanner # 打印一个好看的小Banner: CourseworkManger
5
6           # 获取用户的身份/因为我们使用了有可能会重复的ID
7           # todo: 可以通过构建一个Dummy Table来储存所有用户的相关信息来提供统一认证接口
8           # 当然，这种方式给了用户手动退出系统的接口，否则我们很难定义一个什么特殊值来表示用户希望退出系统
9           while :; do
10              read -rp "请输入您的身份 (T/S/A) 或输入0退出系统：" identity
11              case $identity in
12                  [Tt])
13                      identity="teacher"
14                      break
15                  ;;
16                  [Ss])
17                      identity="student"
18                      break
19                  ;;
20                  [Aa])
21                      identity="admin"
22                      break
23                  ;;
24          0)
25              echo "Bye"
26              return 0
27              ;;
28          *) echo "请输入T, S, A或0" ;;
29      esac
30  done
31
32  # 我们会在密码判断前进行账号检测
33  while :; do
34      read -rp "请输入您的登录账号：" user_id
35      echo "检查中..."
36      sleep 0.1s # 防止暴力登录攻击，100ms的惩罚时间
37
38      # * 防止SQL注入攻击，转义危险字符，详见StudentManageSubmission逻辑
39      user_id=$(RemoveDanger "$user_id")
40
41      # * MySQL调用方式详见StudentUI逻辑
42      query_all_hash="select id, name, password_hash from $identity"
43      query_right_hash="select password_hash from ($query_all_hash) all_hash
where id=\"$user_id\""
44      right_hash=$(mysql_prefix -se "$query_right_hash;")
45      [ -z "$right_hash" ] || break
46      echo "用户不存在，请重新输入"
47  done
48
49  # 我们不会在数据库中储存明文密码
50  # 我们将用户密码进行sha256 hash后储存
51  # 并在登陆时将用户输入的内容进行sha256 hash，与数据库内部的hash值进行比较，若相等则认为密码正确
52  # * 这种方式可以提高系统的安全性
53  # 即使数据库内容被泄露，sha256的加密也会让数据偷盗者很难猜出正确的密码
54  # https://www.youtube.com/watch?v=7U-RbOKanYs
55  while :; do
56      # todo: 使用cat命令可以清楚密码变量，提高安全性，但是我们还没发现该如何换行

```

```

57          # 所以暂时使用了变量来存储密码
58          read -rp "请输入您的密码：" -s password
59          echo ""
60
61
62          password_hash=$(echo "$password" | sha256sum - | tr -d " ")
63          echo "验证中....."
64          sleep 0.1s # 防止暴力登录攻击 · 100ms的惩罚时间
65          [ "$password_hash" = "$right_hash" ] && break
66          sleep 1s # 为了防止暴力登录攻击 · 每次密码错误都要得到1s的时间惩罚
67          echo "验证失败 · 请重新输入"
68      done
69      echo "验证成功"
70      query_name="select name from $identity where id=$user_id"
71      name=$(mysql_prefix -se "$query_name")
72      case $identity in
73          "teacher")
74              TeacherUI "$user_id" "$name"
75              # 这里没有选项循环 · 因此不需要调用break命令
76              # * 详见StudentUI中的逻辑描述
77              ;;
78          "student")
79              StudentUI "$user_id" "$name"
80              ;;
81          "admin")
82              AdminUI "$user_id" "$name"
83              ;;
84
85      esac
86  done
87 }

```

学生操作模块

1. 管理课程

输入要管理的课程号

1. 管理课程作业

输入要管理的作业号

1. 发布新的提交

输入要发布的作业提交内容 · 添加附件等

2. 删除已发布的提交

输入要删除的提交号

3. 修改已发布的提交

输入要修改的提交号

输入新的作业提交内容 · 添加附件等

4. 查看已发布的提交

输入要查看的作业号

5. 返回上一级

2. 返回上一级

2. 查看所有作业完成情况

3. 返回上一级

教师操作模块

1. 管理课程

输入要管理的课程号

1. 管理修读课程的学生

1. 向课程名单中添加学生

输入要添加的学生的学号

2. 从课程名单中移除学生

输入要移除的学生的学号

3. 返回上一级

2. 管理课程作业/实验

1. 发布新的课程作业/实验

输入新的作业/实验内容·截止日期·添加附件等

2. 删除已发布的课程作业/实验

输入要删除的作业/实验号码

3. 修改已发布的课程作业/实验

输入要修改的作业/实验号码

输入新的作业/实验内容·截止日期·添加附件等

4. 查看已发布的作业/实验内容

输入要查看的作业/实验号码

■ 单独查看已完成情况

■ 输入要查看完成情况的学生的学号

■ 输入要查看的提交的提交号码

5. 返回上一级

3. 管理课程简介/公告

1. 管理课程公告

1. 发布新的课程公告

输入新的公告内容·添加附件等

2. 删除已发布的课程公告

输入要删除的公告号码

3. 修改已发布的课程公告

输入要修改的公告号码

输入新的公告内容·添加附件等

4. 查看已发布的公告内容

输入要查看的公告号码

5. 返回上一级

2. 修改课程简介

输入新的课程简介内容

3. 返回上一级
4. 返回上一级
2. 返回上一级

管理员操作模块

1. 管理管理员账户
 1. 添加管理员账户
 2. 删除管理员账户
 3. 修改管理员账户
 4. 返回上一级
2. 管理教师账户
 1. 添加教师账户
 2. 删除教师账户
 3. 修改教师账户
 4. 返回上一级
3. 管理学生账户
 1. 添加学生账户
 2. 删除学生账户
 3. 修改学生账户
 4. 返回上一级
4. 管理课程列表
 1. 添加课程
 2. 删除课程
 3. 修改课程

输入课程的中文、英文名称·添加课程简介等

4. 管理课程讲师

1. 向课程名单中添加课程讲师

输入要添加的讲师的工号

2. 从课程名单中移除课程讲师

输入要删除的讲师的工号

3. 返回上一级

5. 返回上一级

5. 返回上一级

Gadgets小部件

- 清除危险字符模块

可以读取字符串，并检测其全部的字符内容，与给出的 `$danger_set` 变量所示的正则表达式做匹配

对于匹配成功的字符，通过调用 `safe="$safe""\\\"$thechar"` 将其内容添加到末尾

使用时，通过第一个参数 `$1` 传入目标字符串，通过第二个参数传入自定义的 `$2` 正则表达式

```
1  function RemoveDanger() {
2      danger_set=${2:-"\\".\.\*;%}""
3      danger=$1
4      safe=""
5      for i in $(seq ${#danger}); do
6          thechar="${danger:$i-1:1}"
7          if [[ "$thechar" =~ $danger_set ]]; then
8              # echo "$thechar"
9              safe="$safe""\\\"$thechar"
10         else
11             safe="$safe$thechar"
12         fi
13     done
14     echo "$safe"
15 }
```

- 打印附件信息模块

通过预先设定的一些参数（包括 `SQL` 语句和是否存在附件的 `Bool` 值等）

```
1  function PrintAttachment() {
2      # 用于打印附件信息的小函数，可以提高代码可读性
3      # 这个函数认为：
4      # 1. $attachment_count可以用于判断是否有附件需要打印（不一定要是精确的附件数目）
5      # 2. $target是目标内容的字符串描述，例如“课程作业/实验”
6      # 3. $mysql_prefix可以正确执行MySQL命令，$query_attachment可以正确打印相关附件
7      if [ "$attachment_count" -gt 0 ]; then
8          echo "本${target}的附件包括："
9          $mysql_prefix -e "$query_attachment;"
10     else
11         # 我们是用红色显示来让用户快速定位这一提示
12         echo "${Red}本${target}${Red}还没有附件${NoColor}"
13     fi
14 }
```

- 打印各类 ASCII ART

1. Teacher

```

1  function PrintTeacher() {
2      # TEACHER分隔符 · 会在老师登陆后的管理界面打印
3      clear
4      cat <<"EOF"
5      -----
6      /\_ _\ / \ _\ \ / \ _\ \ / \ _\ \ / \ _\ \ / \ _\ \ / \ _\ \
7      == \
8      \/\_/\ \/\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
9      -<
10     \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
11     \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
12     EOF
    }
```

2. Student

```

1  function PrintStudent() {
2      # STUDENT分隔符 · 会在学生登陆后的管理界面打印
3      clear
4      cat <<"EOF"
5      -----
6      /\_ _\ \ / \ _\ \ / \ _\ \ / \ _\ \ / \ _\ \ / \ _\ \ / \ _\ \
7      -\ \
8      \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
9      \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
10     \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
11     EOF
    }
```

3. Admin

```

1  function PrintAdmin() {
2      # ADMIN分隔符 · 会在管理员登陆后的管理界面打印
3      clear
4      cat <<"EOF"
5      -----
6      /\_ _\ \ / \ _\ \ / \ _\ \ / \ _\ \ / \ _\ \ / \ _\ \ / \ _\ \
7      -\ \
8      \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
9      \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
10     \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
11     EOF
    }
```

- 继续运行 按键模块

```
1 function ContinueWithKey() {
2     # 按任意键继续 ...
3     # 有的时候我们会在清空屏幕之前打印一些信息，我们决定给用户一些时间来看看这些信息是什么
4     read -n 1 -rp "${Blue}${Bold}按任意键继续 ... ${NoColor}" -s
5 }
```

主程序

我们通过函数来设计程序：原因是Bash会在读入整个函数的所有内容后运行，这意味着修改脚本的同时运行脚本是可以进行的（原函数已经在内存中了）

一个关于这个问题的讨论

主程序从这里开始，上面定义的都是可供调用的函数

请查看对程序的注释来理解本软件的工作原理

```
1 DefineColor
2 DefineMySQL
3 LoginInUI
```

完整源码

Bash

```
1#!/bin/bash
2# CourseworkInfoSys
3# Author: Xu Zhen 徐震 3180105504
4# shellcheck disable=SC2076
5# 这是一个现代教务管理系统，主要面向作业管理
6# 我们通过编写Shell程序，调用MySQL数据库来管理作业系统
7# ! 您的MySQL版本要至少为5.7
8# ! 您的运行环境最好要有至少150列的字符宽度，因为我们使用了ASCII ART，且很多查询语句的宽度
9# 会较大
10# * 版本的MySQL会在执行tables.sql中的语句时出现问题
11# * 由于许多管理逻辑都是重复的，但将代码集合为一个函数又会显得过于刻意/不灵活，我们会将注释主
12# * 要写在第一次遇到相关逻辑的部分
13# * 阅读源码的最好方式是从头开始，因为我们将主要函数都放在了开头(StudentUI,
14# StudentManageSubmission)
15
16function DefineColor() {
17    # 我们使用tput命令来定义颜色信息
18    # 各类颜色常数，通过echo调用可以改变Shell的输出样式
19    # 例如echo "${Red}Hello${NoColor}"，world."会打印红色的Hello和原色的World
20    # 上述例子会展开成echo "$(tput setaf 1)Hello$(tput sgr0)"，world."
21    # ! consider more about this colorization
22    Red=$(tput setaf 1)
23    Green=$(tput setaf 2)
24    Yellow=$(tput setaf 3)
25    Blue=$(tput setaf 4)
26    Magenta=$(tput setaf 5)
27    Cyan=$(tput setaf 6)
28    Bold=$(tput bold)
29    NoColor=$(tput sgr0)
30    ReturnPrev="${Yellow}${Bold}返回上一级${NoColor}"
31}
```

```
30     function DefineMySQL() {
31         # 我们通过mysql命令来直接执行数据库操作，这也是本实验的核心内容
32         # 我们通过设置文件的方式使得MySQL不会抱怨直接在命令行输入密码不安全：
33         # mysql: [Warning] Using a password on the command line interface can be
34             # * 注意：您可以修改程序运行目录下的.mysql.cnf文件来设置自己的数据库登陆信息
35
36         # ! 第一次使用本软件时请运行当前目录下的table.sql来初始化数据库中的表
37         # 必须运行的部分是所有的create table
38         # 后面的insert内容是可选的，但是至少要有一个管理员账户，否则本软件没有什么意义
39         # 样例初始化语句（假设您知道root密码）：mysql -uroot -p < tables.sql
40
41         # ! 请保证MySQL已经在本机正确安装，且.mysql.cnf已经被正确配置
42         # 您需要在.mysql.cnf中设置您的登录名/密码/服务器，并设置数据库名称（和您在MySQL中使用的
43             # 相同）
44             # 例如您在MySQL中创建了ShellDesigner这个用户，密码为ShellDesigner，并打算使用
45             # ShellDesign这个数据库来管理本软件涉及到的内容
46
47             # 登陆root用户后，可使用如下操作修改密码
48             # ALTER USER 'user'@'hostname' IDENTIFIED BY 'newPass';
49             # 可以通过如下操作创建新用户
50             # create user ShellDesigner identified by 'ShellDesigner';
51             # create database ShellDesign;
52             # grant all on ShellDesign.* to ShellDesigner;
53
54             # .mysql.cnf就将有类似如下的内容
55             # [client]
56             # user=ShellDesigner
57             # password=ShellDesigner
58             # host=localhost
59             # database=ShellDesign
60
61             # 下列是我们默认的一些设置
62             mysql_u_default="ShellDesigner"
63             mysql_p_default="ShellDesigner"
64             mysql_h_default="localhost"
65             mysql_d_default="ShellDesign"
66             mysql_f=".mysql.cnf"
67
68             # 若.mysql.cnf在当前目录不存在，我们会创建一个并将默认内容写入
69             if [ ! -f "$mysql_f" ]; then
70                 echo "Automatically generating configuration file..." >&2
71                 echo "[client]" >$mysql_f
72                 echo "user=$mysql_u_default" >>$mysql_f
73                 echo "password=$mysql_p_default" >>$mysql_f
74                 echo "host=$mysql_h_default" >>$mysql_f
75                 echo "database=$mysql_d_default" >>$mysql_f
76             fi
77
78             # 类似调用alias，我们在下面的Shell语句中执行MySQL调用时都会使用$mysql_prefix来开头
79             mysql_prefix="mysql --defaults-extra-file=$mysql_f"
80
81     }
82
83     # 初始界面登陆逻辑
84     function LoginInUI() {
85         while :; do
86             PrintBanner # 打印一个好看的小Banner: CourseworkManger
87
88
89
90
91
92
93
94
```

```
85      # 获取用户的身份/因为我们使用了有可能会重复的ID
86      # todo: 可以通过构建一个Dummy Table来储存所有用户的相关信息来提供统一认证接口
87      # 当然，这种方式给了用户手动退出系统的接口，否则我们很难定义一个什么特殊值来表示用户
希望退出系统
88      while :; do
89          read -rp "请输入您的身份 ( T/S/A ) 或输入0退出系统：" identity
90          case $identity in
91              [Tt])
92                  identity="teacher"
93                  break
94                  ;;
95              [Ss])
96                  identity="student"
97                  break
98                  ;;
99              [Aa])
100                 identity="admin"
101                 break
102                 ;;
103             0)
104                 echo "Bye"
105                 return 0
106                 ;;
107             *) echo "请输入T, S, A或0" ;;
108             esac
109         done
110
111     # 我们会在密码判断前进行账号检测
112     while :; do
113         read -rp "请输入您的登陆账号：" user_id
114         echo "检查中 ..."
115         sleep 0.1s # 防止暴力登录攻击，100ms的惩罚时间
116
117         # * 防止SQL注入攻击，转义危险字符，详见StudentManageSubmission逻辑
118         user_id=$(RemoveDanger "$user_id")
119
120         # * MySQL调用方式详见StudentUI逻辑
121         query_all_hash="select id, name, password_hash from $identity"
122         query_right_hash="select password_hash from ($query_all_hash)
all_hash where id=\"$user_id\""
123         right_hash=$(mysql_prefix -se "$query_right_hash;")
124         [ -z "$right_hash" ] || break
125         echo "用户不存在，请重新输入"
126     done
127
128     # 我们不会在数据库中储存明文密码
129     # 我们将用户密码进行sha256 hash后储存
130     # 并在登陆时将用户输入的内容进行sha256 hash，与数据库内部的hash值进行比较，若相等
则认为密码正确
131     # * 这种方式可以提高系统的安全性
132     # 即使数据库内容被泄露，sha256的加密也会让数据偷盗者很难猜出正确的密码
133     # https://www.youtube.com/watch?v=7U-RbOKanYs
134     while :; do
135         # todo: 使用cat命令可以清楚密码变量，提高安全性，但是我们还没发现该如何换行
136         # 所以暂时使用了变量来存储密码
137         read -rp "请输入您的密码：" -s password
138         echo ""
139
```

```

140
141         password_hash=$(echo "$password" | sha256sum - | tr -d " ")
142         echo "验证中....."
143         sleep 0.1s # 防止暴力登录攻击·100ms的惩罚时间
144         [ "$password_hash" = "$right_hash" ] && break
145         sleep 1s # 为了防止暴力登录攻击·每次密码错误都要得到1s的时间惩罚
146         echo "验证失败·请重新输入"
147         done
148         echo "验证成功"
149         query_name="select name from $identity where id=$user_id"
150         name=$(mysql_prefix -se "$query_name")
151         case $identity in
152             "teacher")
153                 TeacherUI "$user_id" "$name"
154                 # 这里没有选项循环·因此不需要调用break命令
155                 # * 详见StudentUI中的逻辑描述
156                 ;;
157             "student")
158                 StudentUI "$user_id" "$name"
159                 ;;
160             "admin")
161                 AdminUI "$user_id" "$name"
162                 ;;
163
164             esac
165         done
166     }
167
168     function RemoveDanger() {
169         danger_set=${2:-"[\\'\\.\\*;%]"}
170         danger=$1
171         safe=""
172         for i in $(seq ${#danger}); do
173             thechar="${danger:$i-1:1}"
174             if [[ "$thechar" =~ $danger_set ]]; then
175                 # echo "$thechar"
176                 safe="$safe"\\""$thechar"
177             else
178                 safe="$safe$thechar"
179             fi
180         done
181         echo "$safe"
182     }
183
184     # 以下几个Print函数都是用于打印ASCII Art的
185     # 同时·它们通过调用clear函数来进行假GUI的页面模拟功能
186     # 我们使用ASCII Art的初衷是让用户能在程序的不同Section中更快的找到自己想要的信息
187     # 后来我们发现通过调用clear函数可以达到模拟GUI的功能
188     function PrintBanner() {
189         # 程序的主横幅: CourseworkManger
190         # 会在初始登陆界面打印
191         clear
192         cat <<"EOF"
193         -----
194         \_   __ \ \_   _----- \_   _----- - _-----| | _ _ /
```



```
249      \ \_\\ \_\_ \ \_\_-- \ \_\\ \_\_ \ \_\_ \ \_\_\\ " \_\_
250      \/_/\_/_/ \_\_--/_/ \ \_/_/ \ \_/_/ \ \_/_/ \ \_/_/
251
252  EOF
253 }
254
255 function ContinueWithKey() {
256     # 按任意键继续 ...
257     # 有的时候我们会在清空屏幕之前打印一些信息，我们决定给用户一些时间来看看这些信息是什么
258     read -n 1 -rp "${Blue}${Bold}按任意键继续 ... ${NoColor}" -s
259 }
260
261 function StudentUI() {
262     # 学生UI主界面，为了方便测试我们为sid, name变量加入了默认值
263     sid=${1:-"1"}
264     name=${2:-"st1"}
265     while :; do          # 学生主界面UI循环
266         PrintStudent # 打印Banner
267
268         # 无内容提示信息
269         no_publication="${Red}您本学期没有课程${NoColor}"
270
271         # 为了方便复用和嵌套，我们将所有的SQL查询语句存储在字符串变量中（容易遭到SQL
272         # Injection攻击，后面会提到如何防御）
273         # 注意在每一次事件循环后我们都会尽量更新一次查询语句的变量内容（除非此语句是固定
274         # 的）。
275         query_id="select cid from take where sid=$sid"
276         query_course="select id 课程号, name_zh 中文名称, name_en 英文名称 from
277         course where id in ($query_id)"
278
279         # 第一层括号将返回结果当作数组处理，第二层$()是执行了一个Bash语句，在此是执行了一个
280         # MySQL查询
281         # ! 在本程序中，我们将结果存入变量时基本都会采用这种调用MySQL的方式，我们会使用-se
282         # 选项，其中-e代表执行，-s --slient，安静模式，在此的效果是去除列名
283         # ! 在直接执行MySQL并原封不动的打印信息时，我们会使用-e选项，代表执行
284
285         # * 值得注意的是，在命令行直接调用MySQL时，会打印列分隔符，而将结果存入变量则不会打
286         # 印（列分隔符自动会得到删除）
287
288         # 重定向标准输出的到文件并打印文件
289         # xuzh@ubuntu □ ~/Projects/ShellDesign □ □ master • □ mysql -
290         uShellDesigner -pShellDesigner ShellDesign -e "select * from admin;" > temp.txt;
291         cat temp.txt
292         # mysql: [Warning] Using a password on the command line interface can be
293         # insecure.
294         # name      id      password_hash
295         # root      1
296         53175bcc0524f37b47062fafdda28e3f8eb91d519ca0a184ca71bbebe72f969a
297         # admin      2
298         fc8252c8dc55839967c58b9ad755a59b61b67c13227ddae4bd3f78a38bf394f7
299
300         # 直接执行语句，打印到标准输出
301         # xuzh@ubuntu □ ~/Projects/ShellDesign □ □ master • □ mysql -
302         uShellDesigner -pShellDesigner ShellDesign -e "select * from admin;"
303         # mysql: [Warning] Using a password on the command line interface can be
304         # insecure.
305         # +-----+-----+
306         -----+
```

```

293     # | name | id | password_hash
294     |
295     # +-----+-----+
296     -----+
297     # | root | 1 |
298     53175bcc0524f37b47062fafdda28e3f8eb91d519ca0a184ca71bbebe72f969a |
299     # | admin | 2 |
300     fc8252c8dc55839967c58b9ad755a59b61b67c13227ddae4bd3f78a38bf394f7 |
301     # +-----+-----+
302     -----+
303     # | name | id | password_hash
304     |
305     # +-----+-----+
306     -----+
307     # | root | 1 |
308     53175bcc0524f37b47062fafdda28e3f8eb91d519ca0a184ca71bbebe72f969a |
309     # | admin | 2 |
310     fc8252c8dc55839967c58b9ad755a59b61b67c13227ddae4bd3f78a38bf394f7 |
311     # +-----+-----+
312     -----+
313     # 重定向到变量并打印标准输出
314     # xuzh@ubuntu ~/Projects/ShellDesign master ● mysql -
315     uShellDesigner -pShellDesigner ShellDesign -e "select * from admin;" > /dev/tty
316     # mysql: [Warning] Using a password on the command line interface can be
317     # insecure.
318     # name      id      password_hash
319     # root      1
320     53175bcc0524f37b47062fafdda28e3f8eb91d519ca0a184ca71bbebe72f969a
321     # admin      2
322     fc8252c8dc55839967c58b9ad755a59b61b67c13227ddae4bd3f78a38bf394f7
323
324     # * 因此当我们想要让此变量获取打印的信息时，我们应直接将返回信息赋值到变量中
325     # * 当我们想直接使用MySQL的格式化功能时，我们应直接使用命令将输出导入到/dev/tty
326     cids=$(($mysql_prefix -se "$query_id;"))
327
328     echo "$name同学您好，欢迎来到现代作业管理系统 ( Modern Coursework Manage
329     System )"
330     if [ ${#cids[@]} -eq 0 ]; then
331         echo "$no_publication"
332     else
333         echo "您本学期共${#cids[@]}门课程，它们分别为："
334     fi
335     echo "您可以进行的操作有："
336     echo "1. 管理课程 ( 提交/修改/删除作业 )"
337     echo "2. 查看所有的作业/实验"
338     echo "0. ${ReturnPrev}"
339     while :; do # 操作循环UI，直到获得正确的输入
340         read -rp "请输入您想要进行的操作：" op

```

```

332         case $op in
333             1)
334                 echo "您选择了管理课程"
335                 if [ ${#cids[@]} -eq 0 ]; then
336                     echo "$no_publication"
337                     ContinueWithKey
338                     break
339                 fi
340                 # 直接调用MySQL并输出到/dev/tty可以使MySQL用分割线打印各种信息
341                 $mysql_prefix -e "$query_course;""
342                 while :; do
343                     read -rp "请输入您想要管理的课程号：" cid
344
345                     # 注意到我们使用正则表达式展开数组来进行元素检查
346                     # 因此表达式右侧的值应用引号括起以保证完全匹配
347                     # 我们使用了ShellCheck工具·而此工具会对=~右侧的表达式报错·因此我们
348                     使用了
349                         # shellcheck disable=SC2076
350                         # 来关闭这一报错
351                         [[ "${cids[@]}" =~ "${cid}" ]] && break
352                         echo "您输入的课程号$cid有误·请输入上表中列举出的某个课程号"
353                         done
354
355                     # 每次调用新的函数代表我们将要进入一个新的页面·我们不想让用户在下一页刷新时每次都重复选择某一门课程的过程
356                     # 因此我们将选择好的课程号存储到cid变量中·隐式传递到函数StudentOPCourse
357                     中
358                     StudentOPCourse
359                     break
360                     ;;
361             2)
362                 # 查看所有作业及其完成情况
363                 # 这波·这波是个SQL题·这种长长的还不能格式化的SQL Query也是让人头大
364                 # 我们调用了许多MySQL内置功能·例如UNIX_TIMESTAMP还有IF语句等·还嵌套了
365                 Linux的命令以及变量
366                 # 值得注意的是·对于双引号需要加上转移符号·防止Bash解释它们
367                 echo "您选择了查看所有的作业和实验"
368                 query_all_hw="select sub.hid 作业ID, sub.intro 作业简介,
369                 sub.creation_time 发布时间, sub.end_time 截止时间, if(unix_timestamp(sub.end_time) <$(date +%s),\"是\",\"否\") 是否截止, if(count(sub.id)>0,\"是\",\"否\") 是否完成,
370                 count(sub.id) 创建的提交数目 from (select S.sid, S.id, H.id hid, H.intro,
371                 H.creation_time, H.end_time from (select * from submission where sid=$sid) S
372                 right join homework H on S.hid=H.id where H.cid in (select cid from take where
373                 sid=$sid)) sub group by sub.hid"
374
375                 $mysql_prefix -e "$query_all_hw;""
376
377                 # 我们打印了一些信息·让用户确认一下
378                 ContinueWithKey
379                 break
380                 ;;
381             0)
382                 echo "您选择了${ReturnPrev}"
383                 return 0
384                 ;;
385             *)
386                 echo "您输入的操作$op有误·请输入上面列出的操作"
387                 # 此时不进行Break而是继续请求用户的选择

```

```

380          ;;
381      esac
382      done
383  done
384 }
385
386 function StudentOPCourse() {
387     while :; do
388         # 打印STUDENT Banner
389         PrintStudent
390
391         # target代指我们想要管理的内容的字符串，可以是课程或课程实验/作业。用于格式化打印
392         # 每次刷新页面时都要清空
393         target="${Green}课程实验/作业${NoColor}"
394         # 内容未发布提示信息
395         no_publication="${Red}本课程还没有已发布的${NoColor}${target}"
396
397         # 课程教师查询语句
398         query_tid="select tid from teach where cid=$cid"
399         query_teacher="select id 教师工号, name 教师姓名, if(gender='F', \"女\",
\"男\") 性别, registration_time 注册时间, title 职称, brief 简介 from teacher where
id in ($query_tid)"
400
401         # 课程信息查询语句
402         query_course="select id 课程号, name_zh 中文名称, name_en 英文名称, brief
课程简介 from course where id=$cid"
403
404         echo "您选择的课程为："
405         $mysql_prefix -e "$query_course;"
406
407         echo "教这门课的老师有："
408         $mysql_prefix -e "$query_teacher;"
409
410         # 相关作业/实验查询
411         query_hid="select id from homework where cid=$cid"
412         query_hw="select id 作业ID, intro 作业简介, creation_time 作业发布时间,
end_time 作业截止时间 from homework where cid=$cid"
413
414         # 以数组形式存入变量
415         hids=($(mysql -e "$query_hid;"))
416
417         # 根据数量显示不同的提示
418         if [ ${#hids[@]} -gt 0 ]; then
419             echo "本课程已有的${target}如下图所示"
420             $mysql_prefix -e "$query_hw;"
421         else
422             echo "$no_publication"
423         fi
424
425         echo "您可以进行的操作有："
426         echo "1. 管理${target}"
427         echo "0. ${ReturnPrev}"
428         while :; do
429             read -rp "请输入您想要进行的操作：" op
430             case $op in
431                 1)
432                     echo "您选择了管理本课程的${target}"
433                     # 根据数量显示不同的提示

```

```

434             if [ ${#hids[@]} -eq 0 ]; then
435                 echo "$no_publication"
436                 ContinueWithKey
437                 break
438             fi
439             while :; do
440                 read -rp "请输入您想要管理的${target}ID：" hid
441                 [[ "${hids[@]}" =~ "$hid" ]] && break
442                 echo "您输入的${target}ID$hid有误，请输入上表中列举出的某个
443                     ${target}ID"
444             done
445             # 每次调用新的函数代表我们将要进入一个新的页面，我们不想让用户在下一页刷
446             # 新时每次都重复选择某一项课程作业/实验
447             # 因此我们将选择好的课程号存储到hid变量中，隐式传递到函数中
448             StudentManageSubmission
449
450             break
451         ;;
452     0)
453         echo "您选择了${ReturnPrev}"
454         return 0
455     ;;
456     *)
457         echo "您输入的操作$op有误，请输入上面列出的操作"
458         ;;
459     esac
460     done
461 done
462 function PrintAttachment() {
463     # 用于打印附件信息的小函数，可以提高代码可读性
464     # 这个函数认为：
465     # 1. $attachment_count可以用于判断是否有附件需要打印（不一定要是精确的附件数目
466     # 2. $target是目标内容的字符串描述，例如“课程作业/实验”
467     # 3. $mysql_prefix可以正确执行MySQL命令，$query_attachment可以正确打印相关附件
468     if [ "$attachment_count" -gt 0 ]; then
469         echo "本${target}的附件包括："
470         $mysql_prefix -e "$query_attachment;"
471     else
472         # 我们是用红色显示来让用户快速定位这一提示
473         echo "${Red}本${target}${Red}还没有附件${NoColor}"
474     fi
475 }
476
477 function StudentManageSubmission() {
478     while :; do # 管理作业提交的UI界面主循环，每次重新运行这一循环都会清空界面，退出循环后
479         PrintStudent
480
481         # “提交”的上一级为：“课程作业/实验”
482         upper="${Green}课程作业/实验${NoColor}"
483         target="$upper${Green}提交${NoColor}"
484
485         # 用红色显示的没有提交的信息，方便用户定位
486         no_publication="${Red}您在本${NoColor}$upper${Red}下还没有
487         ${NoColor}${target}"
```

```
488         echo "您选择了修改以下的$upper : "
489         query_course_homework="select id \`作业/实验ID\`, intro \`作业/实验简介\`,
creation_time 创建时间, end_time 截止时间 from homework where id=$hid"
490         query_attachment="select A.id 附件ID, A.name 附件名称, A.url 附件URL from
attachment A join attach_to T on A.id=T.aid where T.uid=$hid"
491         query_count_attachment="select count(1) from attachment join attach_to
on id=aid where uid=$hid"
492         $mysql_prefix -e "$query_course_homework;"
493
494         # 我们通过MySQL Query直接确定相关附件数量的值
495         attachment_count=$(($mysql_prefix -se "$query_count_attachment"))
496
497         # 暂时替换$temp和$upper
498         temp=${target}
499         target=$upper
500         PrintAttachment # 这里我们打印的是upper的附件 · 但PrintAttachment会通过$temp
打印名称
501         target=$temp
502
503         # subid: submission_id : 提交ID
504         query_subids="select id from submission where sid=$sid and hid=$hid"
505         query_subs="select id 提交ID, submission_text 提交内容, creation_time 创建
时间, latest_modification_time 最近修改时间 from submission where id in
($query_subids)"
506
507         subids=$(($mysql_prefix -se "$query_subids;"))
508         if [ ${#subids[@]} -gt 0 ]; then
509             echo "您在本$upper创建的${target}如下所示"
510             $mysql_prefix -e "$query_subs;"
511         else
512             echo "$no_publication"
513             # 这里不可调用break · 会直接退出此界面
514         fi
515
516         query_end_time="select unix_timestamp(end_time) from homework where
id=$hid"
517         end_time=$(($mysql_prefix -se "$query_end_time;"))
518         if [ "$end_time" -lt "$(date +%s)" ]; then
519             echo "${Red}本作业已经截止提交${NoColor}"
520             # ContinueWithKey
521             # break
522         fi
523
524         echo "您可以进行的操作有 : "
525         echo "1. 发布新的${target}"
526         echo "2. 删除已发布的${target}"
527         echo "3. 修改已发布的${target}"
528         echo "4. 查看已发布的${target}"
529         echo "0. ${ReturnPrev}"
530         while :; do # 操作循环
531             read -rp "请输入您想要进行的操作 :" op
532             case $op in
533                 1)
534                     echo "您选择了发布新的${target}"
535                     if [ "$end_time" -lt "$(date +%s)" ]; then
536                         echo "${Red}本作业已经截止提交${NoColor}"
537                         ContinueWithKey
538                         break
```

```
539         fi
540         echo "请输入${target}的简介内容·以EOF结尾(换行后Ctrl+D)"
541
542         # 我们通过连续读取内容直到遇到EOF·也就是Ctrl+D来获取可换行的简介/描述
543         # 注意EOF必须在NewLine后直接输入才有效
544         # 注意到read函数只会读入除了换行符以外的部分·因此换行符需要手动加入
545         # read在遇到EOF后会返回非True值
546         full_string=""
547         while read -r temp; do
548             full_string+="$temp$'\n'"
549         done
550
551         # 我们设计了RemoveDanger函数来减少受到SQL注入攻击的可能性
552         # 简单来讲这一函数的作用就是找到可疑的字符·例如.;*!"等·并对他们进行手动
553         转义
554         # MySQL在处理Query时候会重新解释读入的字符串·原本已经被转义的字符在重新
555         # 解释后很可能不再得到转义·也就给了不法分子可乘之机。
556         full_string=$(RemoveDanger "$full_string")
557
558         echo -e "您的${target}的简介内容为\n$full_string"
559
560         # 由于我们需要保证在Content中与其他具体类型中的标号相同·我们使用数据库的
561         Transaction功能
562         # 通过构建事务·我们保证在Content中添加内容后·submission会获取到相同的
563         ID值·以保证数据完整性和对应性
564         query_insert_content="insert into content value ()"
565         query_insert_submission="insert into submission value
566         (last_insert_id(), $sid, $hid, \"$full_string\", now(), now())"
567
568         # 我们可以通过;串联SQL语句来让它们在同一个MySQL Connection中执行
569         # 注意到我们调用了select last_insert_id()这一语句·这也是这一连串执行中
570         # 唯一有打印内容的一个(返回上次插入的信息)
571         subid=$(mysql_prefix -se "set
572         autocommit=0;$query_insert_content;select
573         last_insert_id();$query_insert_submission;commit;set autocommit=1;")
574
575         echo "您刚刚添加的${target}ID为:$subid"
576
577         # 这里我们通过Bash内部计数来减少一次MySQL链接
578         attachment_count=0
579         while :; do
580             # 我们根据用户回答来修改程序运行流程
581             # 用户无需提前知道需要添加的附件数量
582             # 他/她只需要不断输入Y并添加内容
583             read -rp "请输入您是否需要为${target}添加附件(Y/n)：" need_attach
584             if [[ $need_attach =~ ^[1Yy] ]]; then # 正则表达式匹配
585                 attachment_count+=1
586
587                 echo "您选择了添加附件"
588                 read -rp "请输入您想要添加的附件名称：" attach_name
589                 attach_name=$(RemoveDanger "$attach_name") # 可能包含危险
590
591                 echo "您的附件名称为:$attach_name"
592
593                 read -rp "请输入您想要添加的附件URL：" attach_url
594                 # 对于URL·我们使用不同的转义策略
595                 attach_url=$(RemoveDanger "$attach_url" "[\\'\\.\\*;]")
596
```

```

587                     echo "您的附件URL为 : $attach_url"
588
589                     # 添加附件到附件相关表 · 并修改attach_to表来对应附件和Content的
590                     # 关系
591                     # 我们暂时只使用了attach_to表格的一部分功能 · 在日后的开发中我们
592                     # 可以将一个附件分配给多个不同的Content
593                     # todo: 可以重用已经上传过的附件 · 建立多对多的附加/带附件内容的
594                     # 对应
595                     query_insert_attach="insert into attachment(name, url)
596                     value (\\"$attach_name\\", \\"$attach_url\\")"
597                     query_insert_attach_to="insert into attach_to(aid, uid)
598                     value (last_insert_id(), $subid)"
599
600                     # 同样的 · 我们利用了Transaction功能
601                     attach_id=$(mysql_prefix -se "set
602                     autocommit=0;$query_insert_attach;select
603                     last_insert_id();$query_insert_attach_to;commit;set autocommit=1;")
604
605                     echo "您刚刚添加的附件ID为 : $attach_id"
606                     else
607                     break
608                     fi
609                     done
610
611                     # 打印一些信息 · 让用户得到应有的反馈
612                     echo "您刚刚对课程号为$cid的课程的ID为$hid的$upper发布了如下
613                     ${target} :"
614                     query_course_submission="select id 提交ID, submission_text 提交内
615                     容, creation_time 创建时间, latest_modification_time 最近修改时间 from submission
616                     where id=$subid"
617                     query_attachment="select A.id 附件ID, A.name 附件名称, A.url 附件
618                     URL from attachment A join attach_to T on A.id=T.aid where T.uid=$subid"
619                     $mysql_prefix -e "$query_course_submission;"
620                     PrintAttachment
621
622                     # 下面调用break后就会清空屏幕 · 因此我们给用户一个回顾当下的机会
623                     ContinueWithKey
624                     # 清空屏幕
625                     break
626                     ;;
627
628 2)
629                     echo "您选择了删除已发布的${target}"
630                     if [ "$end_time" -lt "$(date +%s)" ]; then
631                         echo "${Red}本作业已经截止提交${NoColor}"
632                         ContinueWithKey
633                         break
634                     fi
635                     # 若根本没有发布内容 · 删除就是完全无用的
636                     if [ ${#subids[@]} -eq 0 ]; then
637                         echo "$no_publication"
638                         ContinueWithKey
639                         break
640                     fi
641
642                     # 逻辑同上述的while read 循环
643                     while :; do
644                         read -rp "请输入您想要删除的${target}ID : " subid
645                         [[ "${subids[*]}" =~ "${subid}" ]] && break

```

```

634                     echo "您输入的${target}ID$subid有误，请输入上表中列举出的某个
635                     ${target}ID"
636
637                     # 我们对各类Foreign Key使用了on update cascade on delete cascade 功
638                     # 能，就无需显式的删除其他有可能引用到相关内容的东西
639                     query_delete_content="delete from content where id=$subid"
640                     $mysql_prefix -e "$query_delete_content;"
641
642                     break
643
643 3)
644         echo "您选择了修改已发布的${target}"
645         if [ "$end_time" -lt "$(date +%s)" ]; then
646             echo "${Red}本作业已经截止提交${NoColor}"
647             ContinueWithKey
648             break
649         fi
650         # 若根本没有发布内容，修改就是完全无用的
651         if [ ${#subids[@]} -eq 0 ]; then
652             echo "$no_publication"
653             ContinueWithKey
654             break
655         fi
656
657         # 逻辑同上述的while read 循环
658         while :; do
659             read -rp "请输入您想要修改的${target}ID：" subid
660             [[ "${subids[*]}" =~ "$subid" ]] && break
661             echo "您输入的${target}ID$subid有误，请输入上表中列举出的某个
661 ${target}ID"
662
663         done
664
664         echo "您选择修改的${target}为："
665
666         # 让用户观察自己选择修改的内容
667         query_course_submission="select id 提交ID, submission_text 提交内
668         容, creation_time 创建时间, latest_modification_time 最近修改时间 from submission
669         where id=$subid"
670
670         query_attachment="select A.id 附件ID, A.name 附件名称, A.url 附件
671         URL from attachment A join attach_to T on A.id=T.aid where T.uid=$subid"
672         $mysql_prefix -e "$query_course_submission;"
673
673         query_count_attachment="select count(1) from attachment join
674         attach_to on id=aid where uid=$subid"
675         attachment_count=$(($mysql_prefix -se "$query_count_attachment"))
676
676         PrintAttachment
677
677         # 对于full_string的处理同上
678         echo "请输入${target}的简介内容，以EOF结尾（换行后Ctrl+D）"
679         full_string=""
680
680         while read -r temp; do
681             full_string+="$temp$'\n'"
682         done
683
683         full_string=$(RemoveDanger "$full_string")
684
684         echo -e "您的${target}的简介内容为\n$full_string"
685
685         # 同上

```

```

684                 query_modify_submission="update submission set
685                 submission_text=\"$full_string\", latest_modification_time=now() where
686                 id=$subid"
687
688                 $mysql_prefix -e "$query_modify_submission;"
689                 echo "您刚刚修改的${target}ID为 :$subid"
690                 while :; do
691                     read -rp "请输入您是否需要为${target}添加附件 ( Y/n ) : "
692
693                     need_attach
694
695                     if [[ $need_attach =~ ^[1Yy] ]]; then
696                         echo "您选择了添加附件"
697                         read -rp "请输入您想要添加的附件名称 :" attach_name
698                         attach_name=$(RemoveDanger "$attach_name")
699                         echo "您的附件名称为 :$attach_name"
700                         read -rp "请输入您想要添加的附件URL :" attach_url
701                         # 对于URL，我们使用不同的转义策略
702                         attach_url=$(RemoveDanger "$attach_url" "[\\\"\\.\\\\*;]")
703                         echo "您的附件URL为 :$attach_url"
704                         query_insert_attach="insert into attachment(name, url)
705                         value ('$attach_name', '$attach_url')"
706                         query_insert_attach_to="insert into attach_to(aid, uid)
707                         value (last_insert_id(), $subid)"
708                         attach_id=$(($mysql_prefix -se "set
709                         autocommit=0;$query_insert_attach;select
710                         last_insert_id();$query_insert_attach_to;commit;set autocommit=1;")
711                         echo "您刚刚添加的附件ID为 :$attach_id"
712                         else
713                             break
714                         fi
715                         done
716
717                         echo "您刚刚对课程号为$cid的课程的ID为$hid的$upper修改了如下的
718                         ${target} : "
719                         $mysql_prefix -e "$query_course_submission;"
720
721                         attachment_count=$(($mysql_prefix -se "$query_count_attachment"))
722                         PrintAttachment
723                         ContinueWithKey
724                         break
725                         ;;
726
727                         4)
728                         echo "您选择了查询已发布的${target}"
729
730                         # 几乎相同的逻辑
731                         if [ ${#subids[@]} -eq 0 ]; then
732                             echo "$no_publication"
733                             ContinueWithKey
734                             break
735                         fi
736
737                         while :; do
738                             read -rp "请输入您想要查询的作业/实验提交ID :" subid
739                             [[ "${subids[*]}" =~ "$subid" ]] && break
740                             echo "您输入的提交ID$subid有误，请输入上表中列举出的某个提交ID"
741                         done
742
743                         echo "您选择查询的提交为 :"

```

```
732                 query_course_submission="select id 提交ID, submission_text 提交内
容, creation_time 创建时间, latest_modification_time 最近修改时间 from submission
where id=$subid"
733                 query_attachment="select A.id 附件ID, A.name 附件名称, A.url 附件
URL from attachment A join attach_to T on A.id=T.aid where T.uid=$subid"
734                 $mysql_prefix -e "$query_course_submission;"
735
736                 # 没有了添加附件的过程，我们通过调用MySQL接口来进行手动计数
737                 query_count_attachment="select count(1) from attachment join
attach_to on id=aid where uid=$subid"
738                 attachment_count=$(($mysql_prefix -se "$query_count_attachment"))
739                 PrintAttachment
740
741                 # 同样的，打印信息后不直接返回而是继续进行调用
742                 ContinueWithKey
743
744                 # 这里使用了break，因为我们有一个检测命令是否正确的指令
745                 break
746                 ;;
747             0)
748                 echo "您选择了${ReturnPrev}"
749                 return 0
750                 ;;
751             *)
752                 echo "您输入的操作$op有误，请输入上面列出的操作"
753                 ;;
754             esac
755         done
756     done
757 }
758
759 function TeacherUI() {
760     # 同样的，我们使用默认值以方便调试
761     tid=${1:-"1"}
762     name=${2:-"zy"}
763
764     while :; do      # 页面主循环
765         PrintTeacher # 打印TEACHER BANNER提示用户
766
767         no_publication="${Red}您本学期没有课程${NoColor}"
768         query_id="select cid from teach where tid=$tid"
769         query_course="select id 课程号, name_zh 中文名称, name_en 英文名称 from
course where id in ($query_id)"
770
771         # 所有课程数目
772         cids=$(($mysql_prefix -se "$query_id"))
773
774         echo "$name老师您好，欢迎来到现代作业管理系统 ( Modern Coursework Manage
System )"
775         if [ ${#cids[@]} -eq 0 ]; then
776             echo "您本学期没有课程"
777         else
778             echo "您本学期共${#cids[@]}门课程，它们分别为："
779             $mysql_prefix -e "$query_course"
780         fi
781
782         # 虽然只有一个有效选项，但这样处理可以让用户有返回上一级的机会
783         echo "您可以进行的操作有："
```

```

784         echo "1. 管理课程"
785         echo "0. ${ReturnPrev}"
786         while :; do # 错误输入的处理循环，这里只能输入0或者1
787             read -rp "请输入您想要进行的操作：" op
788             case $op in
789             1)
790                 echo "您选择了管理课程"
791                 if [ ${#cids[@]} -eq 0 ]; then
792                     echo "您本学期没有课程"
793                     ContinueWithKey
794                     break
795                 fi
796                 while :; do
797                     read -rp "请输入您想要管理的课程号：" cid
798                     [[ "${cids[*]}" =~ "${cid}" ]] && break
799                     echo "您输入的课程号$cid有误，请输入上表中列举出的某个课程号"
800                 done
801
802             TeacherOPCourse
803             # 若操作过程中没有显式的打印+清屏操作，我们不会让用户按任意键继续
804             break
805             ;;
806         0)
807             echo "您选择了${ReturnPrev}"
808             return 0
809             ;;
810         *)
811             echo "您输入的操作$op有误，请输入上面列出的操作"
812             ;;
813         esac
814     done
815   done
816 }
817
818 function TeacherOPCourse() {
819     while :; do      # 课程操作UI主循环
820         PrintTeacher # 打印Banner
821
822         target="${Green}课程${NoColor}" # 此时的目标字符串为：课程，用绿色显示以方便辨
823         认
824         query_tid="select tid from teach where cid=$cid"
825         query_teacher="select id 教师工号, name 教师姓名, if(gender='F', \"女\",
826         \"男\") 性别, registration_time 注册时间, title 职称, brief 简介 from teacher where
827         id in ($query_tid)"
828
829         echo "您选择的${target}为："
830
831         # 此时我们打印课程简介信息，方便用户在后续使用过程中决定是否要修改课程简介信息
832         $mysql_prefix -e "select id 课程号, name_zh 中文名称, name_en 英文名称,
833         brief 课程简介 from course where id=$cid;"
834
835         # 打印除了当前老师外一同教这门课的老师一共用户参考
836         tids=($(($mysql_prefix -e "$query_tid and tid > $tid;"))
837         if [ ${#tids[@]} -gt 0 ]; then
838             echo "与您一同教这门课的老师有："
839             $mysql_prefix -e "$query_teacher and id > $tid;"
840         else
841             echo "这门${target}只有您自己在教"

```

```
838         fi
839
840         echo "您可以进行的操作有："
841         echo "1. 管理修读${target}的学生"
842         echo "2. 管理${target}作业/实验"
843         echo "3. 管理本${target}信息（管理公告/简介等）"
844         echo "0. ${ReturnPrev}"
845         while :; do
846             # 输入处理循环，这里比较tidy，因为我们将三个子操作都封装成了函数
847             # 且这里无论选择那种操作都没有直接清屏返回的必要
848             read -rp "请输入您想要进行的操作：" op
849             case $op in
850                 1)
851                     echo "您选择了管理修读该${target}的学生"
852                     TeacherManageStudent
853                     break
854                     ;;
855                 2)
856                     echo "您选择了管理本${target}的实验和作业"
857                     TeacherManageHomework
858                     break
859                     ;;
860                 3)
861                     echo "您选择了管理本${target}的公告/信息"
862                     TeacherManageCourse
863                     break
864                     ;;
865                 0)
866                     echo "您选择了${ReturnPrev}"
867                     return 0
868                     ;;
869             *)*
870                 echo "您输入的操作$op有误，请输入上面列出的操作"
871                 ;;
872             esac
873         done
874     done
875 }
876
877 function TeacherManageCourse() {
878     # 和上一个函数有些类似，基本不涉及MySQL操作，因此只是嵌套了一层子菜单
879     while :; do
880         PrintTeacher
881
882         target1="${Green}课程公告${NoColor}"
883         target2="${Green}课程简介${NoColor}"
884         echo "您可以进行的操作有："
885         echo "1. 管理课程$target1"
886         echo "2. 修改课程$target2"
887         echo "0. ${ReturnPrev}"
888         while :; do
889             read -rp "请输入您想要进行的操作：" op
890             case $op in
891                 1)
892                     echo "您选择了管理$target1"
893                     TeacherManageCourseInfo
894                     break
895                     ;;
```

```

896         2)
897             echo "您选择了修改${target2}"
898             TeacherManageCourseBrief
899             break
900             ;;
901         0)
902             echo "您选择了${ReturnPrev}"
903             return 0
904             ;;
905         *)
906             echo "您输入的操作$op有误，请输入上面列出的操作"
907             ;;
908         esac
909     done
910   done
911 }
912
913 function TeacherManageCourseBrief() {
914     # 管理课程简介内容
915     # 因为课程简介只有一个，用户进入这一阶段就一定是为了修改它，因此这一界面没有任何重复性的
916     # 提示信息
917     target="${Green}课程简介${NoColor}"
918     echo "${target}的原内容为"
919     $mysql_prefix -e "select brief 课程简介 from course where id=$cid"
920
921     # 类似的，我们会通过转义危险字符来减少受到MySQL攻击的可能性
922     echo "请输入${target}的新内容，以EOF结尾（换行后Ctrl+D）"
923     # 这种读取方式在前面已经介绍过
924     full_string=""
925     while read -r temp; do
926         full_string+="$temp$'\n'"
927     done
928     full_string=$(RemoveDanger "$full_string")
929
930     echo -e "您的新${target}内容为\n$full_string"
931     query_brief_update="update course set brief = \"$full_string\" where
932     id=$cid"
933
934     # 我们增加了字符串处理函数以减少受到SQL注入攻击的可能性。
935     # we can easily perform SQL injection if the string is not carefully treated
936     # update course set brief = "Hello, world.";select * from admin;" where
937     id=$cid
938     $mysql_prefix -e "$query_brief_update;"
939
940     # 但值得注意的是，课程简介的管理会打印信息，且函数返回后将直接清屏，我们会让用户有机会再
941     # 看一眼
942     ContinueWithKey
943 }
944
945
946 function TeacherManageCourseInfo() {
947     # 管理公告的逻辑和学生管理作业提交的逻辑十分类似
948     # 但细节处又有不少不一样的地方，提取为一个单独的General Purpose函数会显得很Messy
949     while :; do
950         PrintTeacher
951
952         target="${Green}课程公告${NoColor}"
953         no_publication="${Red}本课程没有已发布的${NoColor}${target}"
954
955         query_iid="select id from info where cid=$cid"

```

```
950         query_info="select id 公告ID, release_time 公告发布时间, content 公告内容
951         from info where cid=$cid"
952
953
954     # 惯例：打印一下已有的公告来供用户参考
955     if [ ${#iids[@]} -gt 0 ]; then
956         echo "本课程已有的${target}如下图所示"
957         $mysql_prefix -e "$query_info;"
958     else
959         echo "$no_publication"
960     fi
961
962     echo "您可以进行的操作有："
963     echo "1. 发布新的${target}"
964     echo "2. 删除已发布的${target}"
965     echo "3. 修改已发布的${target}"
966     echo "4. 查询已发布的${target}"
967     echo "0. ${ReturnPrev}"
968
969     while :; do
970         read -rp "请输入您想要进行的操作：" op
971         case $op in
972             1)
973                 echo "您选择了发布新的${target}"
974
975                 # todo：这一段操作可以考虑封装成函数
976                 echo "请输入${target}的新内容，以EOF结尾（换行后Ctrl+D）"
977                 full_string=""
978                 while read -r temp; do
979                     full_string+="$temp$'\n'"
980                 done
981                 full_string=$(RemoveDanger "$full_string")
982                 echo -e "您的新${target}内容为\n$full_string"
983
984                 # 这里的逻辑在上面也有体现
985                 # 由于我们需要保证在Content中与其他具体类型中的标号相同，我们使用Commit
986                 query_insert_content="insert into content value ()"
987                 query_insert_info="insert into info(id, content, cid,
988 release_time) value (last_insert_id(), \"$full_string\", $cid, now())"
989
990                 iid=$(mysql_prefix -se "set
991                 autocommit=0;$query_insert_content;select
992                 last_insert_id();$query_insert_info;commit;set autocommit=1;")
993
994                 echo "您刚刚发布的${target}ID为：$iid"
995                 attachment_count=0
996                 while :; do
997                     read -rp "请输入您是否需要为${target}添加附件 (Y/n)：" need_attach
998                     if [[ $need_attach =~ ^[1Yy] ]]; then
999                         attachment_count+=1
1000                         echo "您选择了添加附件"
1001                         read -rp "请输入您想要添加的附件名称：" attach_name
1002                         attach_name=$(RemoveDanger "$attach_name")
```

```

1003                     attach_url=$(RemoveDanger "$attach_url" "[\\\"\\.\\*;]")
1004                     echo "您的附件URL为 : $attach_url"
1005                     query_insert_attach="insert into attachment(name, url)
1006                         value ('$attach_name', '$attach_url')"
1007                     query_insert_attach_to="insert into attach_to(aid, uid)
1008                         value (last_insert_id(), $iid)"
1009                     attach_id=$(mysql_prefix -se "set
1010             autocommit=0;$query_insert_attach;select
1011             last_insert_id();$query_insert_attach_to;commit;set autocommit=1;")
1012                     echo "您刚刚添加的附件ID为 : $attach_id"
1013                     else
1014                         break
1015                     fi
1016                     done
1017
1018                     echo "您刚刚对课程号为$cid的课程发布了如下的${target} : "
1019                     query_course_info="select I.id 公告ID, I.content 公告内容,
1020                         I.release_time 公告发布时间 from (info I join course C on I.cid=C.id) where
1021                         I.id=$iid;"
1022                     query_attachment="select A.id 附件ID, A.name 附件名称, A.url 附件
1023                         URL from attachment A join attach_to T on A.id=T.aid where T.uid=$iid"
1024                     $mysql_prefix -e "$query_course_info;"
1025
1026                     PrintAttachment
1027                     ContinueWithKey
1028
1029                     break
1030                     ;;
1031
1032 2)
1033 # 完全类似的逻辑
1034 echo "您选择了删除已发布的${target}"
1035 if [ ${#iids[@]} -eq 0 ]; then
1036     echo "$no_publication"
1037     ContinueWithKey
1038     break
1039 fi
1040 while :; do
1041     read -rp "请输入您想要删除的${target}ID：" iid
1042     [[ "${iids[*]}" =~ "${iid}" ]] && break
1043     echo "您输入的${target}ID$iid有误，请输入上表中列举出的某个
1044     ${target}ID"
1045     done
1046     query_delete_content="delete from content where id=$iid"
1047     $mysql_prefix -e "$query_delete_content;"
1048     break
1049     ;;
1050
1051 3)
1052 # 同上
1053 echo "您选择了修改已发布的${target}"
1054 if [ ${#iids[@]} -eq 0 ]; then
1055     echo "$no_publication"
1056     ContinueWithKey
1057     break
1058 fi
1059 while :; do
1060     read -rp "请输入您想要修改的${target}ID：" iid
1061     [[ "${iids[*]}" =~ "${iid}" ]] && break

```

```

1052         echo "您输入的${target}ID${iid}有误，请输入上表中列举出的某个
1053         ${target}ID"
1054         done
1055
1056         # 修改内容前让用户有确认的机会
1057         echo "您选择了修改以下的${target} : "
1058         query_course_info="select I.id 公告ID, I.content 公告内容,
I.release_time 公告发布时间 from (info I join course C on I.cid=C.id) where
I.id=$iid;"
1059         query_attachment="select A.id 附件ID, A.name 附件名称, A.url 附件
URL from attachment A join attach_to T on A.id=T.aid where T.uid=$iid"
1060         query_count_attachment="select count(1) from attachment join
attach_to on id=aid where uid=$iid"
1061         $mysql_prefix -e "$query_course_info;"
1062         attachment_count=$(($mysql_prefix -se "$query_count_attachment"))
1063         PrintAttachment
1064
1065         # 同上
1066         echo "请输入${target}的新内容，以EOF结尾 ( 换行后Ctrl+D ) "
1067         full_string=""
1068         while read -r temp; do
1069             full_string+="$temp$'\n'"
1070         done
1071         full_string=$(RemoveDanger "$full_string")
1072         echo -e "您的新${target}内容为\n$full_string"
1073
1074         query_insert_info="update info set content=\"$full_string\""
1075         where id=$iid"
1076
1077         $mysql_prefix -se "$query_insert_info;"
1078
1079         echo "您刚刚修改的${target}ID为 : $iid"
1080
1081         # 同上
1082         while :; do
1083             read -rp "请输入您是否需要为${target}添加新的附件 ( Y/n ) : "
need_attach
1084             if [[ $need_attach =~ ^[1Yy] ]]; then
1085                 echo "您选择了添加附件"
1086                 read -rp "请输入您想要添加的附件名称 : " attach_name
1087                 attach_name=$(RemoveDanger "$attach_name")
1088                 echo "您的附件名称为 : $attach_name"
1089                 read -rp "请输入您想要添加的附件URL : " attach_url
1090                 # 对于URL，我们使用不同的转义策略
1091                 attach_url=$(RemoveDanger "$attach_url" "[\\\"\\.\\*;]")
1092                 echo "您的附件URL为 : $attach_url"
1093                 query_insert_attach="insert into attachment(name, url)
value ('$attach_name', '$attach_url')"
1094                 query_insert_attach_to="insert into attach_to(aid, uid)
value (last_insert_id(), $iid)"
1095                 attach_id=$(($mysql_prefix -se "set
autocommit=0;$query_insert_attach;select
last_insert_id();$query_insert_attach_to;commit;set autocommit=1;")
1096                 echo "您刚刚添加的附件ID为 : $attach_id"
1097                 else
1098                     break
fi
done

```

```

1099
1100     echo "您刚刚对课程号为$cid的课程发布了如下的${target} : "
1101     $mysql_prefix -e "$query_course_info;"
1102
1103     attachment_count=$(($mysql_prefix -se "$query_count_attachment"))
1104     PrintAttachment
1105     ContinueWithKey
1106
1107     break
1108     ;;
1109 4)
1110     echo "您选择了查询已发布的${target}"
1111     if [ ${#iids[@]} -eq 0 ]; then
1112         echo "$no_publication"
1113         ContinueWithKey
1114         break
1115     fi
1116     while :; do
1117         read -rp "请输入您想要查询的${target}ID : " iid
1118         [[ "${iids[*]}" =~ "${iid}" ]] && break
1119         echo "您输入的${target}ID$iid有误，请输入上表中列举出的某个
${target}ID"
1120     done
1121     echo "您选择了查询以下的${target} : "
1122     query_course_info="select I.id 公告ID, I.content 公告内容,
I.release_time 公告发布时间 from (info I join course C on I.cid=C.id) where
I.id=$iid;"
1123     query_attachment="select A.id 附件ID, A.name 附件名称, A.url 附件
URL from attachment A join attach_to T on A.id=T.aid where T.uid=$iid"
1124     query_count_attachment="select count(1) from attachment join
attach_to on id=aid where uid=$iid"
1125     $mysql_prefix -e "$query_course_info;"
1126     attachment_count=$(($mysql_prefix -se "$query_count_attachment"))
1127     PrintAttachment
1128     ContinueWithKey
1129     break
1130     ;;
1131 0)
1132     echo "您选择了${ReturnPrev}"
1133     return 0
1134     ;;
1135 *)
1136     echo "您输入的操作$op有误，请输入上面列出的操作"
1137     ;;
1138 esac
1139 done
1140 done
1141 }
1142
1143 function TeacherManageStudent() {
1144     # 老师管理学生账户
1145     # 添加/删除到课程等
1146     while :; do
1147         PrintTeacher # 打印Banner
1148         target="${Green}学生${NoColor}"
1149         no_publication="${Red}没有${NoColor}${target}${Red}选上这门课${NoColor}"
1150
1151     # 查询已经选上课的同学们

```

```

1152         query_sid="select sid from take where cid=$cid"
1153         query_student="select id 学生学号, name 学生姓名 from student where id in
1154             ($query_sid)"
1155         sids=$(($mysql_prefix -e "$query_sid;"))
1156         if [ ${#sids[@]} -gt 0 ]; then
1157             echo "选上这门课的$target们有："
1158             $mysql_prefix -e "$query_student;"
1159         else
1160             echo "$no_publication"
1161         fi
1162
1163     # 操作
1164     echo "您可以进行的操作有："
1165     echo "1. 向课程名单中添加$target"
1166     echo "2. 从课程名单中移除$target"
1167     echo "0. ${ReturnPrev}"
1168     while :; do
1169         read -rp "请输入您想要进行的操作：" op
1170         case $op in
1171             1)
1172                 echo "您选择了对课程导入新的$target账户"
1173
1174                 # 列举没有导入到课程下 · 但是已经在管理系统注册了账户的学生方便老师导入
1175                 query_all_sids="select id from student where id not in
1176                     ($query_sid)"
1177                 query_all_students="select id 学号, name 姓名 from student where
1178                     id not in ($query_sid)"
1179                 all_sids=$(($mysql_prefix -se "$query_all_sids;"))
1180                 echo "没有被导入该课程但是已经注册的$target有："
1181                 $mysql_prefix -e "$query_all_students;"
1182                 while :; do
1183                     read -rp "请输入您想要添加的$target学号：" sid
1184                     [[ "${all_sids[*]}" =~ "${sid}" ]] && break
1185                     echo "您输入的学号$sid有误 · 请输入上表中列举出的某个$target的学号"
1186                 done
1187
1188             # 打印下老师选择的同学
1189             echo "您选择了将下列$target添加进课程名单："
1190             query_student_info="select id 学号, name 姓名 from student where
1191                 id=$sid"
1192             $mysql_prefix -e "$query_student_info;"
1193
1194             # 给老师一个确认是否添加的机会
1195             read -rp "是否要添加 (Y/n)：" need_insert_student_course
1196             if [[ $need_insert_student_course =~ ^[1Yy] ]]; then
1197                 query_insert_student_course="insert into take(sid, cid)
1198                     value ($sid, $cid)"
1199                 $mysql_prefix -e "$query_insert_student_course;"
1200             fi
1201             break
1202         ;;
1203     2)
1204         echo "您选择了从课程名单中移除$target"
1205         if [ ${#sids[@]} -eq 0 ]; then
1206             echo "$no_publication"
1207             ContinueWithKey
1208             break
1209         fi

```

```

1205             while :; do
1206                 read -rp "请输入您想要删除的$target学号：" sid
1207                 [[ "${sids[*]}" =~ "$sid" ]] && break
1208                 echo "您输入的学号$sid有误，请输入上表中列举出的某个$target的学号"
1209             done
1210             echo "您选择了将下列$target从课程名单中移除："
1211             query_student_info="select id 学号, name 姓名 from student where
1212                 id=$sid"
1213             $mysql_prefix -e "$query_student_info;"
1214             # 类似的，给老师一个确认的机会
1215             read -rp "是否要移除 (Y/n)：" need_delete_student_course
1216
1217             if [[ $need_delete_student_course =~ ^[1Yy] ]]; then
1218
1219                 # * 值得注意的是，虽然我们已经使用了on delete cascade功能来方便
1220                 # MySQL中的外键管理，但此时的删除并不是删除整个学生账户
1221                 # 而是调整账户使其不再在课程内
1222                 # 这里如果处理不当会出现数据不一致的错误
1223                 # todo：想出一种可以从设计上避免数据不一致的数据库定义范式
1224                 # ! 但这里有一个Paradox：若一个学生被移出课程名单，是否需要清除其已
1225                 # 有的提交呢？
1226                 # * 我们现在选择的是移除，也就是说若学生曾经提交过作业，但老师将其从名
1227                 # 单中移除了，后又添加回来了，他的所有提交都会消失
1228                 query_delete_student_course="delete from take where sid=$sid
1229                 and cid=$cid"
1230                 query_delete_student_attach_to="delete from attach_to where
1231                 uid in (select id from submission where sid=$sid and hid in (select id from
1232                 homework where cid=$cid))"
1233                 query_delete_student_submission="delete from submission
1234                 where sid=$sid and hid in (select id from homework where cid=$cid)"
1235
1236                 # 我们使用了commit来尽量保证操作的完整性
1237                 $mysql_prefix -e "set
1238                 autocommit=0;$query_delete_student_course;$query_delete_student_attach_to;$query
1239                 _delete_student_submission;commit;set autocommit=1;"
1240
1241                 fi
1242                 break
1243                 ;;
1244             0)
1245                 echo "您选择了${ReturnPrev}"
1246                 return 0
1247                 ;;
1248             *)
1249                 echo "您输入的操作$op有误，请输入上面列出的操作"
1250                 ;;
1251             esac
1252         done
1253     done
1254 }
1255
1256 function TeacherManageHomework() {
1257     # * 老师管理作业的逻辑和学生管理作业提交的逻辑十分相似
1258     # 详细注释内容请参考：StudentManageSubmission函数
1259     while :; do
1260         PrintTeacher
1261
1262         target="${Green}课程作业/实验${NoColor}"

```

```

1253         no_publication="${Red}本课程还没有已发布的${NoColor}${target}"
1254
1255         query_hid="select id from homework where cid=$cid"
1256         query_hw="select id 作业ID, intro 作业简介, creation_time 作业发布时间,
1257             end_time 作业截止时间 from homework where cid=$cid"
1258
1259         hids=$(mysql_prefix -e "$query_hid")
1260         if [ ${#hids[@]} -gt 0 ]; then
1261             echo "本课程已有的${target}如下图所示"
1262             mysql_prefix -e "$query_hw"
1263         else
1264             echo "$no_publication"
1265         fi
1266
1267         echo "您可以进行的操作有 : "
1268         echo "1. 发布新的${target}"
1269         echo "2. 删除已发布的${target}"
1270         echo "3. 修改已发布的${target}"
1271         echo "4. 查看已发布的${target}"
1272         echo "0. ${ReturnPrev}"
1273
1274         while :; do
1275             read -rp "请输入您想要进行的操作 :" op
1276             case $op in
1277                 1)
1278                     echo "您选择了发布新的${target}"
1279                     echo "请输入课程实验的简介内容 · 以EOF结尾 ( 换行后Ctrl+D ) "
1280                     full_string=""
1281                     while read -r temp; do
1282                         full_string+="$temp$'\n'"
1283                     done
1284
1285                     full_string=$(RemoveDanger "$full_string")
1286
1287                     echo -e "您的${target}的简介内容为\n$full_string"
1288
1289                     read -rp "请输入您想要创建的是作业还是实验 ( H/E ) :" h_or_e
1290                     [[ $h_or_e =~ ^[Hh] ]] && h_or_e="H" || h_or_e="E"
1291
1292                     while :; do
1293                         read -rp "请输入作业的持续时间 ( 天 ) :" days
1294                         [[ $days =~ ^[0-9]+$ ]] && break || echo "请输入整数"
1295                     done
1296
1297                     # 由于我们需要保证在Content中与其他具体类型中的标号相同 · 我们使用Commit
1298                     # 一天有86400秒
1299                     # 数学运算需要用符号$(( ))进行 · 且运算内部的变量不需要使用$符号 · *等也不需
1300                     要转义
1301                     # 当然 · 我们也可以通过调用expr来进行数学运算 · 不同于上面描述的是 · 使用
1302                     expr需要转义和$
1303                     query_insert_content="insert into content value ()"
1304                     query_insert_hw="insert into homework(id,
1305                         cid,tid,intro,creation_time,end_time) value
1306                         (last_insert_id(),$cid,$tid,\"$full_string\",now(),from_unixtime($((date +%s)
1307                         + days * 86400)))"

```

```

1303                 hid=$(mysql_prefix -se "set
1304                 autocommit=0;${query_insert_content};select
1305                 last_insert_id();${query_insert_hw};commit;set autocommit=1;")
1306
1307                 echo "您刚刚添加的${target}ID为 :$hid"
1308                 attachment_count=0
1309                 while :; do
1310                     read -rp "请输入您是否需要为${target}添加附件 ( Y/n ) : "
1311                     need_attach
1312                     if [[ $need_attach =~ ^[1Yy] ]]; then
1313                         attachment_count+=1
1314                         echo "您选择了添加附件"
1315                         read -rp "请输入您想要添加的附件名称 :" attach_name
1316                         attach_name=$(RemoveDanger "$attach_name")
1317                         echo "您的附件名称为 :$attach_name"
1318                         read -rp "请输入您想要添加的附件URL :" attach_url
1319                         # 对于URL，我们使用不同的转义策略 (对百分号需要进行特殊处理)
1320                         attach_url=$(RemoveDanger "$attach_url" "[\"\\.\*\";]")
1321                         echo "您的附件URL为 :$attach_url"
1322                         query_insert_attach="insert into attachment(name, url)
1323                         value (\\"$attach_name\\", \\"$attach_url\\")"
1324                         query_insert_attach_to="insert into attach_to(aid, uid)
1325                         value (last_insert_id(), $hid)"
1326                         attach_id=$(mysql_prefix -se "set
1327                         autocommit=0;${query_insert_attach};select
1328                         last_insert_id();${query_insert_attach_to};commit;set autocommit=1;")
1329                         echo "您刚刚添加的附件ID为 :$attach_id"
1330                     else
1331                         break
1332                     fi
1333                     done
1334
1335                     # 打印全部信息
1336                     echo "您刚刚对课程号为$cid的课程发布了如下的${target} :"
1337                     query_course_homework="select H.id \`作业/实验ID\`, H.intro \`作业/实验简介\`, H.creation_time 创建时间, H.end_time 结束时间 from homework H where
1338                     H.id=$hid"
1339                     query_attachment="select A.id 附件ID, A.name 附件名称, A.url 附件
1340                     URL from attachment A join attach_to T on A.id=T.aid where T.uid=$hid"
1341                     mysql_prefix -e "$query_course_homework;" | PrintAttachment
1342
1343                     ContinueWithKey
1344
1345                     break
1346                     ;;
1347 2) # 同上
1348                     echo "您选择了删除已发布的${target}"
1349                     if [ ${#hids[@]} -eq 0 ]; then
1350                         echo "$no_publication"
1351                         ContinueWithKey
1352                         break
1353                     fi
1354                     while :; do
1355                         read -rp "请输入您想要删除的${target}ID : " hid
1356                         [[ "${hids[@]}" =~ "$hid" ]] && break

```

```
1350             echo "您输入的${target}ID$hid有误·请输入上表中列举出的某个
1351             ${target}ID"
1352             done
1353             query_delete_content="delete from content where id=$hid"
1354             $mysql_prefix -e "$query_delete_content;""
1355             break
1356             ;;
1357         3)
1358             # 同上
1359             echo "您选择了修改已发布的${target}"
1360             if [ ${#hids[@]} -eq 0 ]; then
1361                 echo "$no_publication"
1362                 ContinueWithKey
1363                 break
1364             fi
1365             while :; do
1366                 read -rp "请输入您想要修改的${target}ID：" hid
1367                 [[ "${hids[@]}" =~ "$hid" ]] && break
1368                 echo "您输入的${target}ID$hid有误·请输入上表中列举出的某个
1369                 ${target}ID"
1370                 done
1371             echo "您选择了修改以下的${target} : "
1372             query_course_homework="select id \`作业/实验ID\`, intro \`作业/实
1373             验简介\`, creation_time 创建时间, end_time 截止时间 from homework where id=$hid"
1374             query_attachment="select A.id 附件ID, A.name 附件名称, A.url 附件
1375             URL from attachment A join attach_to T on A.id=T.aid where T.uid=$hid"
1376             query_count_attachment="select count(1) from attachment join
1377             attach_to on id=aid where uid=$hid"
1378             $mysql_prefix -e "$query_course_homework;"
1379             attachment_count=$(($mysql_prefix -se "$query_count_attachment"))
1380             PrintAttachment
1381
1382             echo "请输入${target}简介的新内容·以EOF结尾(换行后Ctrl+D)"
1383             full_string=""
1384             while read -r temp; do
1385                 full_string+="$temp$'\n'"
1386             done
1387
1388             full_string=$(RemoveDanger "$full_string")
1389
1390             echo -e "您的新${target}简介内容为\n$full_string"
1391
1392             query_insert_homework="update homework set
1393             intro=\"$full_string\" where id=$hid"
1394
1395             $mysql_prefix -se "$query_insert_homework;""
1396
1397             while :; do
1398                 read -rp "请输入作业的持续时间(天)：" days
1399                 [[ $days =~ ^[0-9]+\$ ]] && break || echo "请输入整数"
1400             done
1401
1402             query_get_start_time="select unix_timestamp(creation_time) from
1403             homework where id=$hid"
1404             creation_time=$(($mysql_prefix -se "$query_get_start_time;")
```

```

1400                 query_update_end_time="update homework set
1401                 end_time=from_unixtime($((creation_time + days * 86400))) where id=$hid"
1402
1403                 echo "您刚刚修改的课程${target}ID为 : $hid"
1404                 attachment_count=$(mysql_prefix -e "$query_update_end_time;" )
1405                 while :; do
1406                     read -rp "请输入您是否需要为课程${target}添加新的附件 ( Y/n ) : "
1407                     need_attach
1408                     if [[ $need_attach =~ ^[1Yy] ]]; then
1409                         echo "您选择了添加附件"
1410                         read -rp "请输入您想要添加的附件名称 : " attach_name
1411                         attach_name=$(RemoveDanger "$attach_name")
1412                         echo "您的附件名称为 : $attach_name"
1413                         read -rp "请输入您想要添加的附件URL : " attach_url
1414                         # 对于URL，我们使用不同的转义策略
1415                         attach_url=$(RemoveDanger "$attach_url" "[\\\"\\.\\*;]")
1416                         echo "您的附件URL为 : $attach_url"
1417                         query_insert_attach="insert into attachment(name, url)
1418                         value (\\"$attach_name\\", \\"$attach_url\\")"
1419                         query_insert_attach_to="insert into attach_to(aid, uid)
1420                         value (last_insert_id(), $hid)"
1421                         attach_id=$(mysql_prefix -se "set
1422                         autocommit=0;$query_insert_attach;select
1423                         last_insert_id();$query_insert_attach_to;commit;set autocommit=1;")
1424                         echo "您刚刚添加的附件ID为 : $attach_id"
1425                     else
1426                         break
1427                     fi
1428                     done
1429
1430                     echo "您刚刚对课程号为$cid的课程发布了如下的课程${target} : "
1431                     $mysql_prefix -e "$query_course_homework;"
1432                     PrintAttachment
1433                     ContinueWithKey
1434
1435                     break
1436                     ;;
1437
1438             4)
1439                 echo "您选择了查看已发布的${target}的完成情况"
1440                 if [ ${#hids[@]} -eq 0 ]; then
1441                     echo "$no_publication"
1442                     ContinueWithKey
1443                     break
1444                 fi
1445                 while :; do
1446                     read -rp "请输入您想要查看的${target}ID : " hid
1447                     [[ "${hids[*]}" =~ "${hid}" ]] && break
1448                     echo "您输入的${target}ID$hid有误，请输入上表中列举出的某个
1449                     ${target}ID"
1450                     done
1451
1452                     echo "您选择了查询以下的${target} : "
1453                     query_course_homework="select id \`作业/实验ID\`, intro \`作业/实
1454                     验简介\`, creation_time 创建时间, end_time 截止时间 from homework where id=$hid"
1455                     query_attachment="select A.id 附件ID, A.name 附件名称, A.url 附件
1456                     URL from attachment A join attach_to T on A.id=T.aid where T.uid=$hid"

```

```

1448                 query_count_attachment="select count(1) from attachment join
1449                 attach_to on id=aid where uid=$hid"
1450                 $mysql_prefix -e "$query_course_homework;"
1451                 attachment_count=$(($mysql_prefix -se "$query_count_attachment"))
1452                 PrintAttachment
1453
1454                 query_sid="select sid from take where cid=$cid"
1455                 query_finish="select stu.id 学生学号, stu.name 学生姓名,
1456                 if(count(sub.id)>0,\\"是\",\\"否\\") 是否完成, count(sub.id) 创建的提交数目 from
1457                 (select * from submission where hid=$hid) sub right join (select * from student
1458                 where id in ($query_sid)) stu on sub.sid=stu.id group by stu.id"
1459
1460                 $mysql_prefix -e "$query_finish"
1461                 read -rp "请输入您是否单独查询完成情况 (Y/n) :" check_finish
1462                 if [[ $check_finish =~ ^[1Yy] ]]; then
1463                     CheckFinishYet
1464                     fi
1465                     break
1466                     ;;
1467                 0)
1468                     echo "您选择了${ReturnPrev}"
1469                     return 0
1470                     ;;
1471                 *)
1472                     echo "您输入的操作$op有误·请输入上面列出的操作"
1473                     ;;
1474                     esac
1475             done
1476         done
1477     }
1478
1479     function CheckFinishYet() {
1480         # 当学习某门课的学生过多·我们可以单独检查他们的作业完成情况
1481         while :; do
1482             PrintTeacher
1483             query_sid="select sid from take where cid=$cid"
1484             query_finish="select stu.id 学生学号, stu.name 学生姓名,
1485             if(count(sub.id)>0,\\"是\",\\"否\\") 是否完成, count(sub.id) 创建的提交数目 from
1486             (select * from submission where hid=$hid) sub right join (select * from student
1487             where id in ($query_sid)) stu on sub.sid=stu.id group by stu.id"
1488             sids=$(($mysql_prefix -se "$query_sid;"))
1489             $mysql_prefix -e "$query_finish"
1490             while :; do
1491                 if [[ $check_finish =~ ^[1Yy] ]]; then
1492                     while :; do
1493                         read -rp "请输入您想要查询完成情况的学号:" sid
1494                         [[ "${sids[*]}" =~ "${sid}" ]] && break
1495                         echo "您输入的学号$sid有误·请输入上表中列举出的某个$target的学号"
1496                     done
1497                     query_finish_sid="$query_finish having stu.id=$sid"
1498                     $mysql_prefix -e "$query_finish_sid"
1499
1500                     # subid: submission_id : 提交ID
1501                     query_subids="select id from submission where sid=$sid and
1502                     hid=$hid"
1503                     query_subs="select id 提交ID, submission_text 提交内容,
1504                     creation_time 创建时间, latest_modification_time 最近修改时间 from submission where
1505                     id in ($query_subids)"

```

```
1496
1497         subids=$(($mysql_prefix -se "$query_subids;"))
1498         if [ ${#subids[@]} -gt 0 ]; then
1499             echo "本学生在本作业/实验下创建的提交如下所示"
1500             $mysql_prefix -e "$query_subs;"
1501         else
1502             echo "${Red}本学生还没有在该作业上发布提交${NoColor}"
1503             read -rp "请输入您是否单独查询完成情况 (Y/n) :" check_finish
1504             break
1505             # 这里不可调用break，会直接退出此界面
1506         fi
1507
1508         while :; do
1509             read -rp "请输入您想要查询的作业/实验提交ID：" subid
1510             [[ "${subids[*]}" =~ "$subid" ]] && break
1511             echo "您输入的提交ID$subid有误，请输入上表中列举出的某个提交ID"
1512         done
1513
1514         echo "您选择查询的提交为："
1515         query_course_submission="select id 提交ID, submission_text 提交内
容, creation_time 创建时间, latest_modification_time 最近修改时间 from submission
where id=$subid"
1516         query_attachment="select A.id 附件ID, A.name 附件名称, A.url 附件
URL from attachment A join attach_to T on A.id=T.aid where T.uid=$subid"
1517         $mysql_prefix -e "$query_course_submission;"
1518
1519         # 没有了添加附件的过程，我们通过调用MySQL接口来进行手动计数
1520         query_count_attachment="select count(1) from attachment join
attach_to on id=aid where uid=$subid"
1521         attachment_count=$(($mysql_prefix -se "$query_count_attachment"))
1522         PrintAttachment
1523     else
1524         return 0
1525     fi
1526
1527         read -rp "请输入您是否单独查询完成情况 (Y/n) :" check_finish
1528         break
1529         done
1530     done
1531 }
1532
1533 function AdminUI() {
1534     # 同样的，我们使用默认值以方便调试
1535     # me_admin_id=${1:-"1"}
1536     name=${2:-"root"}
1537
1538     while :; do      # 页面主循环
1539         PrintAdmin # 打印ADMIN BANNER提示用户
1540
1541         echo "$name管理员您好，欢迎来到现代作业管理系统 ( Modern Coursework Manage
System ) "
1542
1543         # 此处仅仅是一个菜单
1544         # 为了方便查询和利用屏幕空间，我们仅仅在选定操作类型后才打印相关信息
1545         echo "您可以进行的操作有："
1546         echo "1. 管理管理员账户"
1547         echo "2. 管理教师账户"
1548         echo "3. 管理学生账户"
```

```
1549     echo "4. 管理课程列表"
1550     echo "0. ${ReturnPrev}"
1551     while :; do # 错误输入的处理循环，这里只能输入0或者1
1552         read -rp "请输入您想要进行的操作：" op
1553         case $op in
1554             1)
1555                 echo "您选择了管理管理员账户"
1556                 AdminManageAdmin
1557                 break
1558                 ;;
1559             2)
1560                 echo "您选择了管理教师账户"
1561                 AdminManageTeacher
1562                 break
1563                 ;;
1564             3)
1565                 echo "您选择了管理学生账户"
1566                 AdminManageStudent
1567                 break
1568                 ;;
1569             4)
1570                 echo "您选择了管理课程列表"
1571                 AdminManageCourse
1572                 break
1573                 ;;
1574             0)
1575                 echo "您选择了${ReturnPrev}"
1576                 return 0
1577                 ;;
1578             *)
1579                 echo "您输入的操作$op有误，请输入上面列出的操作"
1580                 ;;
1581             esac
1582         done
1583     done
1584 }
1585
1586 function AdminManageCourse() {
1587     # 课程管理逻辑
1588     # 很多操作已经在前面详细描述过了
1589     # * 我们意识到，很多逻辑都是重复的，但又有很多小细节难以抹平
1590     # * 我们考虑过将许多大量重复的小段代码抽象成单独函数，但这些小代码中往往有break等控制逻辑，很难实现
1591     # * 我们也考虑过直接抽象所有管理逻辑到一个函数中，但那样要处理的细节过多，难以调试，很难定位到底是哪里出错
1592     # 因此现在采用了较为直接的方法，每种逻辑都使用了不同的函数以方便排查错误和提供模块化功能
1593     # （一个模块宕机其他模块也能暂时正常使用）
1594     # todo：重构重复性高的内容到一个大函数中
1595     # mark：没有C++等语言的面向对象特性，这种复杂逻辑的设计其实是极为困难的，或许Shell语言
1596     # 的目的本身就不是如此吧
1597     while :; do
1598         PrintAdmin
1599         target="${Green}课程${NoColor}"
1600         no_course="${Red}系统中没有课程${NoColor}"
1601         query_cid="select id from course"
1602         query_course="select id 课程号, name_zh 中文名称, name_en 英文名称, brief
1603         课程简介 from course"
1604         cids=(${mysql_prefix -se "$query_cid"})
1605 }
```

```
1602
1603     # 打印已有的课程信息
1604     if [ ${#cids[@]} -gt 0 ]; then
1605         echo "系统中已有的${target}如下所示："
1606         $mysql_prefix -e "$query_course;"
1607     else
1608         echo "$no_course"
1609     fi
1610     echo "您可以进行的操作有："
1611     echo "1. 添加${target}"
1612     echo "2. 删除${target}"
1613     echo "3. 修改${target}"
1614     echo "4. 管理${target}讲师" # 就是管理哪个老师可以讲什么课的逻辑
1615     echo "0. ${ReturnPrev}"
1616     while :; do
1617         read -rp "请输入您想要进行的操作：" op
1618         case $op in
1619             1)
1620                 echo "您选择了添加${target}"
1621
1622                 # 值得注意的是，我们没有对用户输入的是中文还是英文作出严格的判断
1623                 # 因此用户可以根据自己的喜好来调整名称的结果，中文名称中也可以有拉丁字母出
1624             现
1625                 # 获取并确认中文内容
1626                 read -rp "请输入您想要的新${target}中文名称：" c_name_zh
1627                 c_name_zh=$(RemoveDanger "$c_name_zh")
1628                 echo "您的${target}名称为：$c_name_zh"
1629
1630                 # 获取并确认英文内容
1631                 read -rp "请输入您想要的新${target}英文名称：" c_name_en
1632                 c_name_en=$(RemoveDanger "$c_name_en")
1633                 echo "您的${target}名称为：$c_name_en"
1634
1635                 # 获取并确认简介内容
1636                 echo "请输入${target}的简介内容，以EOF结尾（换行后Ctrl+D）"
1637                 full_string=""
1638                 while read -r temp; do
1639                     full_string+="$temp$'\n'"
1640                 done
1641                 full_string=$(RemoveDanger "$full_string")
1642                 echo -e "您的${target}的简介内容为\n$full_string"
1643
1644                 query_insert_course="insert into course(name_zh, name_en, brief)
1645                 value (\\"$c_name_zh\\",\\"$c_name_en\\",\\"$full_string\\")"
1646                 query_last_insert_id="select last_insert_id()"
1647
1648                 cid=$(mysql_prefix -se
1649                 "$query_insert_course;$query_last_insert_id;")
1650                 echo "添加成功....."
1651                 query_course_new="$query_course where id=$cid"
1652                 echo "您新添加的$target如下所示"
1653                 $mysql_prefix -e "$query_course_new;"
1654                 ContinueWithKey
1655                 break
1656             ;;
1657         2)
1658             echo "您选择了删除${target}"
1659             if [ ${#cids[@]} -eq 0 ]; then
```

```
1657             echo "$no_course"
1658             ContinueWithKey
1659             break
1660         fi
1661         while :; do
1662             read -rp "请输入您想要删除的${target}ID：" cid
1663             [[ "${cids[*]}" =~ "${cid}" ]] && break
1664             echo "您输入的${target}ID$cid有误，请输入上表中列举出的某个
1665             ${target}ID"
1666             done
1667             echo "您选择了将下列$target从课程名单中移除，${Red}注意：其所有相关信息
1668             都会丢失：${NoColor}：""
1669             query_course_info="$query_course where id=$cid"
1670             $mysql_prefix -e "$query_course_info;""
1671
1672             # 类似的，给老师一个确认的机会
1673             read -rp "是否要移除 (Y/n)：" need_delete
1674             if [[ $need_delete =~ ^[1Yy] ]]; then
1675                 query_delete_course="delete from course where id=$cid"
1676                 $mysql_prefix -e "$query_delete_course"
1677             fi
1678             break
1679         ;;
1680         echo "您选择了修改${target}"
1681         if [ ${#cids[@]} -eq 0 ]; then
1682             echo "$no_course"
1683             ContinueWithKey
1684             break
1685         fi
1686         while :; do
1687             read -rp "请输入您想要修改的${target}ID：" cid
1688             [[ "${cids[*]}" =~ "${cid}" ]] && break
1689             echo "您输入的${target}ID$cid有误，请输入上表中列举出的某个
1690             ${target}ID"
1691             done
1692
1693             read -rp "请输入您想要的新${target}中文名称：" c_name_zh
1694             c_name_zh=$(RemoveDanger "$c_name_zh")
1695             echo "您的${target}名称为：$c_name_zh"
1696
1697             read -rp "请输入您想要的新${target}英文名称：" c_name_en
1698             c_name_en=$(RemoveDanger "$c_name_en")
1699             echo "您的${target}名称为：$c_name_en"
1700
1701             echo "请输入${target}的简介内容，以EOF结尾(换行后Ctrl+D)"
1702             full_string=""
1703             while read -r temp; do
1704                 full_string+="$temp$'\n'"
1705             done
1706             full_string=$(RemoveDanger "$full_string")
1707             echo -e "您的${target}的简介内容为\n$full_string"
1708
1709             query_change_course="update course set name_zh=\"$c_name_zh\",
1710             name_en=\"$c_name_en\", brief=\"$full_string\" where id=$cid"
1711             $mysql_prefix -e "$query_change_course;"
```

```

1711             echo "$target修改成功 ... "
1712             query_course_new="$query_course where id=$cid"
1713             echo "您新添加的$target如下所示"
1714             $mysql_prefix -e "$query_course_new;"
1715             ContinueWithKey
1716
1717             break
1718             ;;
1719         4)
1720             echo "您选择了管理${target}讲师"
1721             if [ ${#cids[@]} -eq 0 ]; then
1722                 echo "$no_course"
1723                 ContinueWithKey
1724                 break
1725             fi
1726             while :; do
1727                 read -rp "请输入您想要管理的${target}ID：" cid
1728                 [[ "${cids[*]}" =~ "${cid}" ]] && break
1729                 echo "您输入的${target}ID$cid有误，请输入上表中列举出的某个
${target}ID"
1730             done
1731
1732             AdminManageTeaching
1733             break
1734             ;;
1735
1736         0)
1737             echo "您选择了${ReturnPrev}"
1738             return 0
1739             ;;
1740         *)
1741             echo "您输入的操作$op有误，请输入上面列出的操作"
1742             ;;
1743         esac
1744         done
1745     done
1746 }
1747
1748 function AdminManageTeaching() {
1749     while :; do
1750         PrintAdmin
1751         target="${Green}教师${NoColor}"
1752         no_teacher="${Red}没有教师教授这门课${NoColor}"
1753
1754         query_tid="select tid from teach where cid=$cid"
1755         query_teacher_basic="select id 教师工号, name 教师姓名, if(gender='F', \"女
\", \"男\") 性别, registration_time 注册时间, title 职称, brief 简介 from teacher"
1756         query_teacher="$query_teacher_basic where id in ($query_tid)"
1757         tids=(${mysql_prefix -se "$query_tid"})
1758
1759         if [ ${#tids[@]} -gt 0 ]; then
1760             echo "系统中已有的教这门课的${target}如下所示："
1761             $mysql_prefix -e "$query_teacher;"
1762         else
1763             echo "$no_teacher"
1764         fi
1765         # 操作
1766         echo "您可以进行的操作有："

```

```

1767     echo "1. 向课程名单中添加${target}"
1768     echo "2. 从课程名单中移除${target}"
1769     echo "${ReturnPrev}"
1770     while :; do
1771         read -rp "请输入您想要进行的操作：" op
1772         case $op in
1773             1)
1774                 echo "您选择了对课程导入新的${target}账户"
1775
1776                 # 列举没有导入到课程下·但是已经在管理系统注册了账户的学生方便老师导入
1777                 query_all_tids="select id from teacher where id not in
1778                     (${query_tid})"
1779                 query_all_teachers="$query_teacher_basic where id not in
1780                     (${query_tid})"
1781                 all_tids=(${mysql_prefix -se "$query_all_tids;"})
1782                 echo "没有被导入该课程但是已经注册的${target}有："
1783                 $mysql_prefix -e "$query_all_teachers;"
1784                 while :; do
1785                     read -rp "请输入您想要添加的${target}ID：" tid
1786                     [[ "${all_tids[@]}" =~ "${tid}" ]] && break
1787                     echo "您输入的ID${tid}有误·请输入上表中列举出的某个${target}的ID"
1788                 done
1789
1790                 # 打印下老师选择的同学
1791                 echo "您选择了将下列${target}添加进课程名单："
1792                 query_teacher_info="$query_teacher_basic where id=${tid}"
1793                 $mysql_prefix -e "$query_teacher_info;"
1794
1795                 # 给老师一个确认是否添加的机会
1796                 read -rp "是否要添加(Y/n)：" need_insert_teacher_course
1797                 if [[ $need_insert_teacher_course =~ ^[1Yy] ]]; then
1798                     query_insert_teacher_course="insert into teach(tid, cid)
1799                         value (${tid}, ${cid})"
1800                     $mysql_prefix -e "$query_insert_teacher_course;"
1801                 fi
1802                 break
1803             ;;
1804         2)
1805             echo "您选择了从课程名单中移除${target}"
1806             if [ ${#tids[@]} -eq 0 ]; then
1807                 echo "$no_teacher"
1808                 ContinueWithKey
1809                 break
1810             fi
1811             while :; do
1812                 read -rp "请输入您想要移除的${target}ID：" tid
1813                 [[ "${all_tids[@]}" =~ "${tid}" ]] && break
1814                 echo "您输入的ID${tid}有误·请输入上表中列举出的某个${target}的ID"
1815             done
1816             echo "您选择了将下列${target}从课程名单中移除"
1817             query_teacher_info="$query_teacher_basic where id=${tid}"
1818             $mysql_prefix -e "$query_teacher_info;"
1819
1820             # 类似的·给老师一个确认的机会
1821             read -rp "是否要移除(Y/n)：" need_delete_teacher_course
1822             if [[ $need_delete_teacher_course =~ ^[1Yy] ]]; then

```



```

1873             echo "您的${target}名称为 : $s_name"
1874
1875             # 为了表示对女性的尊重 · 我们将无法判断为男性的任何情况都设定为女性教师/学生
1876             read -rp "请输入新的${target}对应的性别 ( M/F ) :" s_gender
1877             [[ $s_gender =~ ^[Mm] ]] && s_gender="M" || s_gender="F"
1878             echo "您选择的性别为 : $s_gender"
1879
1880             echo "请输入${target}的简介内容 · 以EOF结尾 ( 换行后Ctrl+D ) "
1881             full_string=""
1882             while read -r temp; do
1883                 full_string+="$temp$'\n'"
1884             done
1885
1886             full_string=$(RemoveDanger "$full_string")
1887
1888             echo -e "您的${target}的简介内容为\n$full_string"
1889
1890             while :; do
1891                 read -rp "请输入您的密码 :" -s password
1892                 echo ""
1893                 password_hash_ori=$(echo "$password" | sha256sum - | tr -d "
1894 -")
1895
1896                 read -rp "请确认您的密码 :" -s password
1897                 echo ""
1898                 password_hash_fi=$(echo "$password" | sha256sum - | tr -d "
1899 -")
1900
1901                 if [ "$password_hash_ori" = "$password_hash_fi" ]; then
1902                     echo "密码设置成功...."
1903                     break
1904                 fi
1905                 echo "您两次输入的密码不一致 · 请重新输入"
1906             done
1907             query_insert_student="insert into student(name, brief, gender,
1908 password_hash, enroll_time) value ('$s_name', '$full_string', '$s_gender',
1909 '$password_hash', now())"
1910             query_last_insert_id="select last_insert_id()"
1911
1912             sid=$(mysql_prefix -se
1913 "$query_insert_student;$query_last_insert_id;")
1914
1915             query_student_new="$query_student where id=$sid"
1916             echo "添加成功....."
1917
1918             echo "您新添加的${target}如下所示"
1919             $mysql_prefix -e "$query_student_new;"
1920             ContinueWithKey
1921             break
1922             ;;
1923
1924             echo "您选择了删除${target}"
1925             if [ ${#sids[@]} -eq 0 ]; then

```

```

1926             echo "您输入的${target}ID$sid有误·请输入上表中列举出的某个
1927             ${target}ID"
1928         done
1929         echo "您选择了将下列$target从系统中移除·${Red}注意：其所有相关信息都
1930         会丢失：${NoColor}"
1931         query_student_info="$query_student where id=$sid"
1932         $mysql_prefix -e "$query_student_info;"
1933
1934         # 类似的·给老师一个确认的机会
1935         read -rp "是否要移除(Y/n)：" need_delete
1936         if [[ $need_delete =~ ^[1Yy] ]]; then
1937             query_delete_student="delete from student where id=$sid"
1938             $mysql_prefix -e "$query_delete_student"
1939         fi
1940         break
1941     ;;
1942     3)
1943         echo "您选择了修改${target}"
1944         if [ ${#sids[@]} -eq 0 ]; then
1945             echo "$no_student"
1946             ContinueWithKey
1947             break
1948         fi
1949         while :; do
1950             read -rp "请输入您想要修改的${target}ID：" sid
1951             [[ "${sids[@]}" =~ "$sid" ]] && break
1952             echo "您输入的${target}ID$sid有误·请输入上表中列举出的某个
1953             ${target}ID"
1954         done
1955
1956         read -rp "请输入您想要的新${target}名称：" s_name
1957         s_name=$(RemoveDanger "$s_name")
1958         echo "您的${target}名称为：$s_name"
1959
1960         read -rp "请输入新的${target}对应的性别(M/F)：" s_gender
1961         [[ $s_gender =~ ^[Mm] ]] && s_gender="M" || s_gender="F"
1962         echo "您选择的性别为：$s_gender"
1963
1964         echo "请输入${target}的简介内容·以EOF结尾(换行后Ctrl+D)"
1965         full_string=""
1966         while read -r temp; do
1967             full_string+="$temp$'\n'"
1968         done
1969
1970         full_string=$(RemoveDanger "$full_string")
1971
1972         echo -e "您的${target}的简介内容为\n$full_string"
1973
1974         # 由于密码比较敏感·我们会首先询问用户是否需要真的修改
1975         # 这里我们与先前提到的内容对应·使用sha256sum来储存密码以提高安全性
1976         # 即使数据库遭到泄露·用户也无法直接获得密码
1977         read -rp "是否要修改${target}密码(Y/n)：" need_change_pw
1978         if [[ $need_change_pw =~ ^[1Yy] ]]; then
1979             while :; do
1980                 read -rp "请输入您的密码：" -s password
1981                 echo ""
1982                 password_hash_ori=$(echo "$password" | sha256sum - | tr
1983                 -d " -")

```

```

1980                     read -rp "请确认您的密码：" -s password
1981                     echo ""
1982                     password_hash_fi=$(echo "$password" | sha256sum - | tr -
1983                         d " -")
1984                     if [ "$password_hash_ori" = "$password_hash_fi" ]; then
1985                         query_change_pw="update student set
1986                         password_hash=\"$password_hash_ori\" where id=$sid"
1987                         $mysql_prefix -e "$query_change_pw;""
1988                         echo "密码修改成功 ... "
1989                         break
1990                     done
1991                 fi
1992
1993                     query_change_student="update student set name=\"$s_name\",
1994                     brief=\"$full_string\", gender=\"$s_gender\" where id=$sid"
1995                     $mysql_prefix -e "$query_change_student;""
1996
1997                     echo "修改成功 ... "
1998                     query_student_new="$query_student where id=$sid"
1999                     echo "您新添加的$target如下所示"
2000                     $mysql_prefix -e "$query_student_new;""
2001                     ContinueWithKey
2002
2003                     break
2004                     ;;
2005             0)
2006                     echo "您选择了${ReturnPrev}"
2007                     return 0
2008                     ;;
2009             *)
2010                     echo "您输入的操作$op有误，请输入上面列出的操作"
2011                     ;;
2012             esac
2013         done
2014     done
2015 }
2016
2017 function AdminManageTeacher() {
2018     # 与管理同学的逻辑十分相似
2019     while :; do
2020         PrintAdmin
2021         target="${Green}教师账户${NoColor}"
2022         no_teacher="${Red}系统中没有教师${NoColor}"
2023
2024         query_tid="select id from teacher"
2025         query_teacher="select id 教师工号, name 教师姓名, if(gender='F', \"女\",
2026             \"男\") 性别, registration_time 注册时间, title 职称, brief 简介 from teacher"
2027         tids=($(${mysql_prefix} -se "$query_tid;"))
2028
2029         if [ ${#tids[@]} -gt 0 ]; then
2030             echo "系统中已有的${target}如下所示："
2031             $mysql_prefix -e "$query_teacher;""
2032         else
2033             echo "$no_teacher"
2034         fi

```

```
2034     echo "您可以进行的操作有 : "
2035     echo "1. 添加${target}"
2036     echo "2. 删除${target}"
2037     echo "3. 修改${target}"
2038     echo "0. ${ReturnPrev}"
2039     while :; do
2040         read -rp "请输入您想要进行的操作 :" op
2041         case $op in
2042             1)
2043                 echo "您选择了添加${target}"
2044
2045                 read -rp "请输入您想要的新${target}名称 :" t_name
2046                 t_name=$(RemoveDanger "$t_name")
2047                 echo "您的${target}名称为 : $t_name"
2048
2049                 read -rp "请输入新的${target}对应的性别 ( M/F ) :" t_gender
2050                 [[ $t_gender =~ ^[Mm] ]] && t_gender="M" || t_gender="F"
2051                 echo "您选择的性别为 : $t_gender"
2052
2053                 read -rp "请输入您的${target}的职称 :" t_title
2054                 t_title=$(RemoveDanger "$t_title")
2055                 echo "您的${target}职称为 : $t_title"
2056
2057                 echo "请输入${target}的简介内容 . 以EOF结尾 ( 换行后Ctrl+D ) "
2058                 full_string=""
2059                 while read -r temp; do
2060                     full_string+="$temp$'\n'"
2061                 done
2062
2063                 full_string=$(RemoveDanger "$full_string")
2064
2065                 echo -e "您的${target}的简介内容为\n$full_string"
2066
2067                 while :; do
2068                     read -rp "请输入您的密码 :" -s password
2069                     echo ""
2070                     password_hash_ori=$(echo "$password" | sha256sum - | tr -d "
2071                     -")
2072
2073                     read -rp "请确认您的密码 :" -s password
2074                     echo ""
2075                     password_hash_fi=$(echo "$password" | sha256sum - | tr -d "
2076                     -")
2077
2078                     if [ "$password_hash_ori" = "$password_hash_fi" ]; then
2079                         echo "密码设置成功...."
2080                         break
2081                     fi
2082                     echo "您两次输入的密码不一致 . 请重新输入"
2083                 done
2084                 query_insert_teacher="insert into teacher(name, brief, title,
2085                 gender, password_hash, registration_time) value
2086                 ('\"$t_name\"', '\"$full_string\"', '\"$t_title\"', '\"$t_gender\"',
2087                 '\"$password_hash\"', now())"
2088
2089                 query_last_insert_id="select last_insert_id()"
2090
2091                 tid=$(mysql_prefix -se
2092                 "$query_insert_teacher;$query_last_insert_id;")
2093                 echo "添加成功....."
2094
2095
```

```

2086             query_teacher_new="$query_teacher where id=$tid"
2087             echo "您新添加的${target}如下所示"
2088             $mysql_prefix -e "$query_teacher_new;" 
2089             ContinueWithKey
2090             break
2091             ;;
2092         2)
2093             echo "您选择了删除${target}"
2094             if [ ${#tids[@]} -eq 0 ]; then
2095                 echo "$no_teacher"
2096                 ContinueWithKey
2097                 break
2098             fi
2099             while :; do
2100                 read -rp "请输入您想要删除的${target}ID：" tid
2101                 [[ "${tids[*]}" =~ "${tid}" ]] && break
2102                 echo "您输入的${target}ID$tid有误，请输入上表中列举出的某个
2103                 ${target}ID"
2104             done
2105             echo "您选择了将下列${target}从课程名单中移除，${Red}注意：其所有相关信息
2106             都会丢失：${NoColor}：" 
2107             query_teacher_info="$query_teacher where id=$tid"
2108             $mysql_prefix -e "$query_teacher_info;" 
2109
2110             # 类似的，给老师一个确认的机会
2111             read -rp "是否要移除(Y/n)：" need_delete
2112             if [[ $need_delete =~ ^[1Yy] ]]; then
2113                 query_delete_teacher="delete from teacher where id=$tid"
2114                 $mysql_prefix -e "$query_delete_teacher"
2115             fi
2116             break
2117         3)
2118             echo "您选择了修改${target}"
2119             if [ ${#tids[@]} -eq 0 ]; then
2120                 echo "$no_teacher"
2121                 ContinueWithKey
2122                 break
2123             fi
2124             while :; do
2125                 read -rp "请输入您想要修改的${target}ID：" tid
2126                 [[ "${tids[*]}" =~ "${tid}" ]] && break
2127                 echo "您输入的${target}ID$tid有误，请输入上表中列举出的某个
2128                 ${target}ID"
2129             done
2130
2131             read -rp "请输入您想要的新${target}名称：" t_name
2132             t_name=$(RemoveDanger "$t_name")
2133             echo "您的${target}名称为：$t_name"
2134
2135             # 为了表示对女性的尊重，我们将无法判断为男性的任何情况都设定为女性教师/学生
2136             read -rp "请输入新的${target}对应的性别(M/F)：" t_gender
2137             [[ $t_gender =~ ^[Mm] ]] && t_gender="M" || t_gender="F"
2138             echo "您选择的性别为：$t_gender"
2139
2140             read -rp "请输入您的${target}的职称：" t_title
2141             t_title=$(RemoveDanger "$t_title")
2142             echo "您的${target}职称为：$t_title"

```

```
2141
2142     echo "请输入${target}的简介内容，以EOF结尾（换行后Ctrl+D）"
2143     full_string=""
2144     while read -r temp; do
2145         full_string+="$temp$'\n'"
2146     done
2147
2148     full_string=$(RemoveDanger "$full_string")
2149
2150     echo -e "您的${target}的简介内容为\n$full_string"
2151
2152     read -rp "是否要修改${target}密码 (Y/n) :" need_change_pw
2153     if [[ $need_change_pw =~ ^[1Yy] ]]; then
2154         while :; do
2155             read -rp "请输入您的密码：" -s password
2156             echo ""
2157             password_hash_ori=$(echo "$password" | sha256sum - | tr
2158             -d " ")
2159             read -rp "请确认您的密码：" -s password
2160             echo ""
2161             password_hash_fi=$(echo "$password" | sha256sum - | tr -
2162             d " ")
2163             if [ "$password_hash_ori" = "$password_hash_fi" ]; then
2164                 query_change_pw="update teacher set
2165                 password_hash=\"$password_hash_ori\" where id=$tid"
2166                 $mysql_prefix -e "$query_change_pw;""
2167                 echo "密码修改成功 ..."
2168                 break
2169             fi
2170             echo "您两次输入的密码不一致，请重新输入"
2171         done
2172     fi
2173     query_change_teacher="update teacher set name=\"$t_name\",
2174     brief=\"$full_string\", gender=\"$t_gender\", title=\"$t_title\" where id=$tid"
2175     $mysql_prefix -e "$query_change_teacher;""
2176     echo "教师账户修改成功 ..."
2177     query_teacher_new="$query_teacher where id=$tid"
2178     echo "您新添加的$target如下所示"
2179     $mysql_prefix -e "$query_teacher_new;""
2180     ContinueWithKey
2181
2182     break
2183 0)
2184     echo "您选择了${ReturnPrev}"
2185     return 0
2186     ;;
2187 *)
2188     echo "您输入的操作$op有误，请输入上面列出的操作"
2189     ;;
2190 esac
2191 done
2192 done
2193 }
2194
```



```

2248             ContinueWithKey
2249             break
2250             ;;
2251         2)
2252             # 各类小操作的逻辑都十分相似
2253             echo "您选择了删除${target}"
2254             if [ ${#admid[@]} -eq 0 ]; then
2255                 echo "$no_admin"
2256                 ContinueWithKey
2257                 break
2258             fi
2259             while :; do
2260                 read -rp "请输入您想要删除的${target}ID：" admid
2261                 [[ "${admid[*]}" =~ "${admid}" ]] && break
2262                 echo "您输入的${target}ID$admid有误，请输入上表中列举出的某个
2263                 ${target}ID"
2264             done
2265             echo "您选择了将下列$target从课程名单中移除："
2266             query_admin_info="select id 管理员账号, name 管理员姓名 from admin
2267             where id=$admid"
2268             $mysql_prefix -e "$query_admin_info"
2269
2270             # 类似的，给老师一个确认的机会
2271             read -rp "是否要移除(Y/n)：" need_delete
2272             if [[ $need_delete =~ ^[1Yy] ]]; then
2273                 query_delete_admin="delete from admin where id=$admid"
2274                 $mysql_prefix -e "$query_delete_admin"
2275             fi
2276             break
2277         3)
2278             echo "您选择了修改${target}"
2279             if [ ${#admid[@]} -eq 0 ]; then
2280                 echo "$no_admin"
2281                 ContinueWithKey
2282                 break
2283             fi
2284             while :; do
2285                 read -rp "请输入您想要修改的${target}ID：" admid
2286                 [[ "${admid[*]}" =~ "${admid}" ]] && break
2287                 echo "您输入的${target}ID$admid有误，请输入上表中列举出的某个
2288                 ${target}ID"
2289             done
2290
2291             read -rp "请输入您想要的新${target}名称：" admin_name
2292             admin_name=$(RemoveDanger "$admin_name")
2293             echo "您的${target}名称为：$admin_name"
2294
2295             query_change_admin_name="update admin set name=\"$admin_name\""
2296             where id=$admid"
2297             $mysql_prefix -e "$query_change_admin_name;"
2298
2299             read -rp "是否要修改${target}密码(Y/n)：" need_change_pw
2300             if [[ $need_change_pw =~ ^[1Yy] ]]; then
2301                 while :; do
2302                     read -rp "请输入您的密码：" -s password
2303                     echo ""

```

```

2301                         password_hash_ori=$(echo "$password" | sha256sum - | tr
2302                         -d " -")
2303                         read -rp "请确认您的密码：" -s password
2304                         echo ""
2305                         password_hash_fi=$(echo "$password" | sha256sum - | tr -
2306                         d " -")
2307                         if [ "$password_hash_ori" = "$password_hash_fi" ]; then
2308                             query_change_pw="update admin set
2309                             password_hash=\"$password_hash_ori\" where id=$admid"
2310                             $mysql_prefix -e "$query_change_pw;""
2311                             echo "密码修改成功...."
2312                             break
2313                         fi
2314                         echo "您两次输入的密码不一致，请重新输入"
2315                         done
2316                         fi
2317                         echo "管理员账户修改成功...."
2318                         query_admin_new="select id 管理员账号, name 管理员姓名 from admin
2319                         where id=$admid"
2320                         echo "您新修改的$target如下所示"
2321                         $mysql_prefix -e "$query_admin_new;""
2322                         ContinueWithKey
2323                         break
2324                         ;;
2325                         0)
2326                         echo "您选择了${ReturnPrev}"
2327                         return 0
2328                         ;;
2329                         *)
2330                         echo "您输入的操作$op有误，请输入上面列出的操作"
2331                         ;;
2332                         esac
2333                         done
2334                         done
2335                         }
2336                         # ! 我们通过函数来设计程序：原因是Bash会在读入整个函数的所有内容后运行，这意味着修改脚本的同
2337                         # 时运行脚本是可以进行的（原函数已经在内存中了）
2338                         # https://www.shellscript.sh/tips/change-running-script/
2339                         # 主程序从这里开始，上面定义的都是可供调用的函数
2340                         # 请查看对程序的注释来理解本软件的工作原理
2341                         DefineColor
2342                         DefineMySQL
2343                         LoginInUI

```

MySQL

```

1  # note: we should define the default charset of the database before creating the
2  # tables without explicitly
3  # defining charset
4
5  drop database if exists ShellDesign;
6  drop user if exists ShellDesigner;
7  create user ShellDesigner identified by 'ShellDesigner';
8  create database ShellDesign;
9  grant all on ShellDesign.* to ShellDesigner;
10 alter database ShellDesign character set utf8mb4 collate utf8mb4_unicode_ci;

```

```
10  use ShellDesign;
11
12  drop table if exists `take`;
13  drop table if exists `info`;
14  drop table if exists `teach`;
15  drop table if exists `attach_to`;
16  drop table if exists `attachment`;
17  drop table if exists `submission`;
18  drop table if exists `homework`;
19  drop table if exists `content`;
20  drop table if exists `teacher`;
21  drop table if exists `student`;
22  drop table if exists `admin`;
23  drop table if exists `course`;
24
25  create table `teacher`
26  (
27      name          varchar(100),
28      id            bigint primary key auto_increment,
29      brief         varchar(2000),
30      gender        enum ('F', 'M') default 'F', # F for female and M for male
31      registration_time datetime,
32      title         varchar(500)    default 'Professor',
33      password_hash varchar(64)
34 );
35
36  create table `student`
37  (
38      name          varchar(100),
39      id            bigint primary key auto_increment,
40      brief         varchar(2000),
41      gender        enum ('F', 'M') default 'F', # F for female and M for male
42      enroll_time   datetime,
43      password_hash char(64)
44 );
45
46  create table `admin`
47  (
48      name          varchar(100),
49      id            bigint primary key auto_increment,
50      password_hash char(64)
51 );
52
53  create table `course`
54  (
55      name_zh       varchar(100),
56      name_en       varchar(100),
57      brief         varchar(2000),
58      syllabus      varchar(4000),
59      id            bigint primary key auto_increment
60 );
61
62  create table `teach`
63  (
64      tid bigint,
65      cid bigint,
66      foreign key (`tid`) references teacher (`id`) on delete cascade on update
cascade,
```

```

67      foreign key (`cid`) references course (`id`) on delete cascade on update
68      cascade
69  );
70
71  create table `take`
72  (
73      cid bigint,
74      sid bigint,
75      foreign key (`sid`) references student (`id`) on delete cascade on update
76      cascade,
77      foreign key (`cid`) references course (`id`) on delete cascade on update
78      cascade
79  );
78  # this is a dummy class so that we can ensure foreign key references from
79  # attachments to both submissions and homework
80  create table `content`
81  (
82      id bigint primary key auto_increment
83  );
84
84  create table `info`
85  (
86      id          bigint primary key,
87      content     varchar(2000),
88      cid         bigint,
89      release_time datetime,
90      foreign key (`cid`) references course (`id`) on delete cascade on update
91      cascade,
92      foreign key (`id`) references content (`id`) on delete cascade on update
93      cascade
94  );
94
94  create table `homework`
95  (
96      id          bigint primary key auto_increment,
97      cid         bigint,
98      tid         bigint,
99      intro       varchar(2000),
100     creation_time datetime,
101     end_time    datetime,
102     type        enum ('H', 'E') default 'H', # H for homework and e for
103     experiment
104     foreign key (`id`) references content (`id`) on delete cascade on update
105     cascade,
106     foreign key (`tid`) references teacher (`id`) on delete cascade on update
107     cascade,
108     foreign key (`cid`) references course (`id`) on delete cascade on update
109     cascade
110  );
110
108  create table `submission`
109  (
110      id          bigint primary key auto_increment,
111      sid         bigint,
112      hid         bigint,
113      submission_text varchar(2000),
114      creation_time datetime,

```

```
115      latest_modification_time datetime,
116      foreign key (`id`) references content (`id`) on delete cascade on update
117      cascade,
118      foreign key (`sid`) references student (`id`) on delete cascade on update
119      cascade,
120      foreign key (`hid`) references homework (`id`) on delete cascade on update
121      cascade
122 );
123
124 create table `attachment`
125 (
126     id    bigint primary key auto_increment,
127     name  varchar(100),
128     url   varchar(800),
129     brief varchar(2000)
130 );
131
132 create table `attach_to`
133 (
134     aid  bigint,
135     uid  bigint,
136     foreign key (`aid`) references attachment (`id`) on delete cascade on update
137     cascade,
138     foreign key (`uid`) references content (`id`) on delete cascade on update
139     cascade
140 );
141
142 insert into `course`(id, name_zh, name_en)
143 values (1, '数据库系统', 'Database System'),
144         (2, 'Linux程序设计', 'Linux Program Design'),
145         (3, '高级数据结构与算法分析', 'Advances Data Structure and Algorithm
146 Design'),
147         (4, '计算机图形学', 'Computer Graphics'),
148         (5, '视觉识别中的深度卷积神经网络', 'Convolutional Neural Network for Visual
149 Recognition'),
150         (6, 'iOS开发', 'iOS Software Development');
151
152 insert into `teacher`(id, name, password_hash, registration_time)
153 values (1, 'zy',
154         '49aabdaa1b0f6c3506f54521ef81fe5b5fe835d268f1f86e1021a342b59d43bc', now(), #
155         password is zy
156         (2, 'xz',
157         'b44f7d6b5283a44ee5f2bd98f84087a04810092122d75e8fbf8ad85f8f2981f1', now()); #
158         password is xz
159
160 insert into `admin`(id, name, password_hash)
161 values (1, 'root',
162         '53175bcc0524f37b47062fafdda28e3f8eb91d519ca0a184ca71bbebe72f969a'), # password
163         is root
164         (2, 'admin',
165         'fc8252c8dc55839967c58b9ad755a59b61b67c13227ddae4bd3f78a38bf394f7'); # password
166         is admin
167
168 insert into `student`(id, name, password_hash, enroll_time)
169 values (1, 'st1',
170         '2238ead9c048f351712c34d22b41f6eec218ea9a9e03e48fad829986b0dafc11', now(), #
171         password is same as name
```

```
155      (2, 'st2',
156      '5e61d026a7889d9fc72e17f1b25f4d6d48bfe17046fea845aa8c5651ec89c333', now()),
157      (3, 'st3',
158      'bbb977f8e93feb5dbd79e0688b822115b5acf774dd8a1fe6964e03d6b9579384', now()),
159      (4, 'st4',
160      '6133396ebcd382b137088d2ea91d60637744e404b4376e4635b45784b718db72', now()),
161      (5, 'st5',
162      'd691a62aa63f1be970582902d0ff78df29899f09c5dd540b1447cdd051dcfc8d', now()),
163      (6, 'st6',
164      'a7a287ffc9cb27131b9dc54199ba96cef87e753968bc620d714af212ef0f7a8c', now()),
165      (7, 'st7',
166      '73d0daf13c6159a1fbdeb37b6972325b6e29c312371a0f3d427bd35c0c87b928', now()),
167      (8, 'st8',
168      '4ce70fc1eef7303879a2ef33996db2f85058ae06e8590521267ae8d46ec59793', now());
169
170
171  insert into `teach`(cid, tid)
172  values (1, 1),
173          (1, 2),
174          (2, 1),
175          (3, 1),
176          (4, 2),
177          (5, 2);
178
179
180  insert into `take`(cid, sid)
181  values (1, 1),
182          (1, 2),
183          (1, 3),
184          (1, 4),
185          (2, 3),
186          (2, 4),
187          (2, 5),
188          (2, 6),
189          (3, 7),
190          (3, 8),
191          (4, 1),
192          (4, 3),
193          (4, 5),
194          (5, 2),
195          (5, 4),
196          (5, 6),
197          (5, 8),
198          (6, 1),
199          (6, 7),
200          (6, 8);
201
202
203  insert into homework(id, cid, tid, intro, creation_time, end_time, type)
204  values (5, 1, 1, '实验4 JDBC系统的编写和使用', now(), now() + interval 7 day, 'E'),
205          (6, 1, 1, '第五周数据库系统作业', now(), now() + interval 10 day, 'H');
```

```

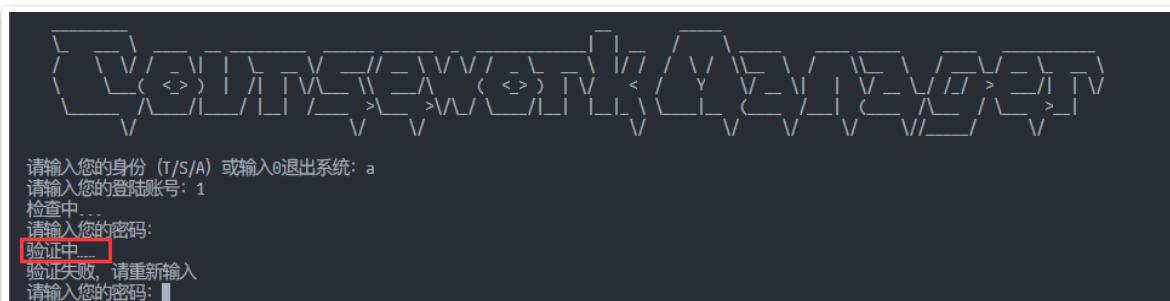
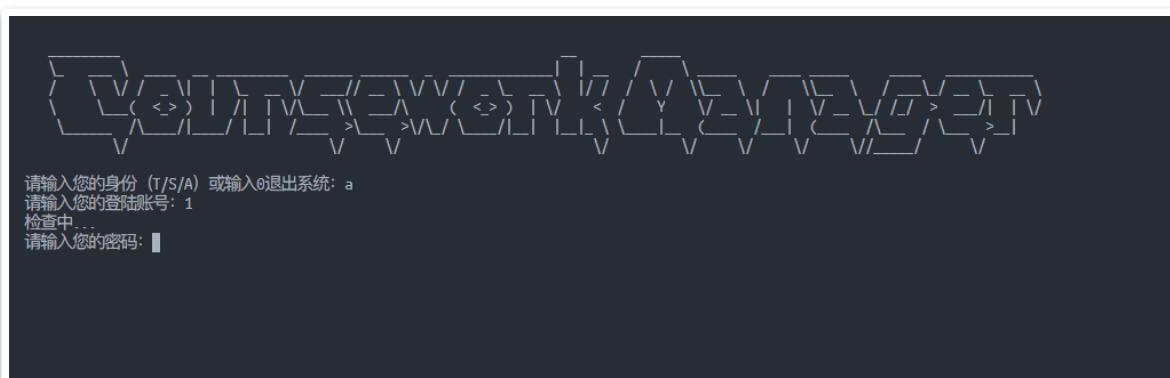
206      (7, 1, 2, '课程大作业 MiniSQL的编写与使用', now(), now() + interval 20 day,
207      'H');
208
209      insert into attachment(id, name, url)
210      values (1, 'Linux Shell Program Design 3rd Edition.pdf',
211              'https://raw.githubusercontent.com/dendenxu/minisQL/master/minisQL.tex'),
212              (2, '数据库系统实验报告',
213               'https://raw.githubusercontent.com/dendenxu/minisQL/master/xz.tex'),
214               (3, '蒙特卡洛树搜索实现',
215                'https://raw.githubusercontent.com/dendenxu/DeepOthello/master/MCTS.py'),
216                (4, 'JDBC接口调用参考与举例',
217                 'https://raw.githubusercontent.com/dendenxu/DeepOthello/master/MCTS.py');
218
219      insert into info(id, content, cid, release_time)
220      values (1, '作业1的提交就要截止啦！请大家及时关注。', 1, NOW()),
221              (2, '实验5的验收将在本周六下午4点开始·请需要验收的组长搜索"数据库系统"钉钉群并加入·
222 钉钉群二维码详见附件', 1, NOW()),
223              (3, 'ADS考试将在6月24日以线上/机房同时考试的形式进行·YDS老师的复习视频已上传到学在
224 浙大系统·详见附件', 3, NOW()),
225              (4, '明天的实验内容为样条插值(Spline)以及贝塞尔曲线的拟合(Bezier Path)·请同学
226 们提前预习相关内容·PPT已上传附件并开放下载', 4, NOW());
227
228      insert into attach_to(aid, uid)
229      values (1, 1),
230              (1, 2),
231              (1, 3),
232              (2, 1),
233              (2, 5),
234              (2, 6),
235              (4, 5),
236              (3, 1);

```

程序运行结果截图

由于可供实验的内容的可能性组合着实太多·我们在此仅仅展示一部分的实验结果

登陆界面：防止恶意登陆的罚时0.1s/1s操作



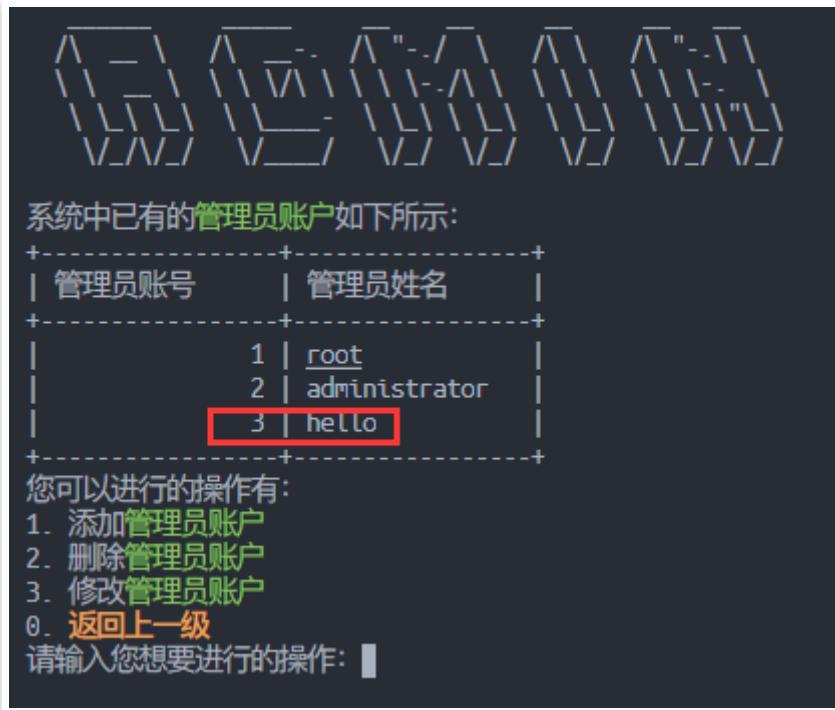
登录成功界面的Banner以及有颜色的返回上一级信息



管理界面需管理信息的高亮显示以及任意键继续功能

添加新的管理员账户测试 (以及登录测试)

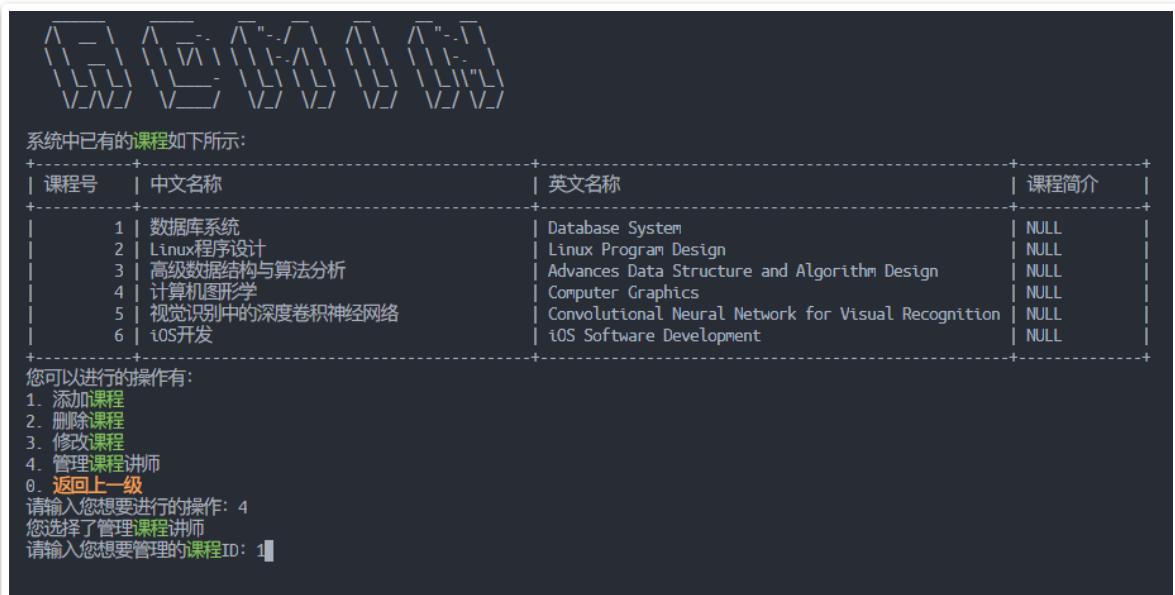




测试后可以用新添加的账号重新登陆



课程教师管理功能测试



系统中已有的教这门课的教师如下所示:

教师工号	教师姓名	性别	注册时间	职称	简介
1	zy	女	2020-07-18 16:57:12	Professor	NULL
2	xz	女	2020-07-18 16:57:12	Professor	NULL

您可以进行的操作有:

1. 向课程名单中添加教师
2. 从课程名单中移除教师
0. 返回上一级

请输入您想要进行的操作: 1

系统中已有的教这门课的教师如下所示:

教师工号	教师姓名	性别	注册时间	职称	简介
1	zy	女	2020-07-18 16:57:12	Professor	NULL
2	xz	女	2020-07-18 16:57:12	Professor	NULL

您可以进行的操作有:

1. 向课程名单中添加教师
2. 从课程名单中移除教师
0. 返回上一级

请输入您想要进行的操作: 1

您选择了对课程导入新的教师账户

没有被导入该课程但是已经注册的教师有:

教师工号	教师姓名	性别	注册时间	职称	简介
4	new	女	2020-07-20 00:24:10	中国工程院院士	

请输入您想要添加的教师ID: 4

您选择了将下列教师添加进课程名单:

教师工号	教师姓名	性别	注册时间	职称	简介
4	new	女	2020-07-20 00:24:10	中国工程院院士	

是否要添加 (y/n) : y



教师管理界面中可以看到刚刚添加的新老师



作业完成情况统计与检查

本课程已有的课程作业/实验如下图所示

作业ID	作业简介	作业发布时间	作业截止时间
5	实验4 JDBC系统的编写和使用	2020-07-18 16:57:12	2020-07-25 16:57:12
6	第五周数据库系统作业	2020-07-18 16:57:12	2020-07-28 16:57:12
7	课程大作业 MiniSQL的编写与使用	2020-07-18 16:57:12	2020-08-07 16:57:12

您可以进行的操作有：

1. 发布新的课程作业/实验
2. 删除已发布的课程作业/实验
3. 修改已发布的课程作业/实验
4. 查看已发布的课程作业/实验
0. 返回上一级

请输入您想要进行的操作：4

您选择了查看已发布的课程作业/实验的完成情况

请输入您想要查看的课程作业/实验ID: 5

您选择了查询以下的课程作业/实验：

作业/实验ID	作业/实验简介	创建时间	截止时间
5	实验4 JDBC系统的编写和使用	2020-07-18 16:57:12	2020-07-25 16:57:12

本课程作业/实验的附件包括：

附件ID	附件名称	附件URL
2	数据库系统实验报告	https://raw.githubusercontent.com/dendenxu/miniSQL/master/xz.tex
4	JDBC接口调用参考与举例	https://raw.githubusercontent.com/dendenxu/DeepHello/master/MCTS.py

请输入您是否单独查询完成情况 (y/n) : ■

请输入您想要查询完成情况的学号：1

学生学号	学生姓名	是否完成	创建的提交数目
1	st1	否	0
2	st2	否	0
3	st3	否	0
4	st4	否	0

本学生还没有在该作业上发布提交

请输入您是否单独查询完成情况 (y/n) : ■

您可以进行的操作有：
1. 管理课程 [课程公告](#)
2. 修改课程 [课程简介](#)
0. [返回上一级](#)

请输入您想要进行的操作: 2

您选择了修改[课程简介](#)

课程简介的原内容为

+-----+	课程简介	+-----+
NULL		+-----+

请输入课程简介的新内容, 以EOF结尾 (换行后Ctrl+D)

```
Hello, world.\";select * from admin;" where id=$cid
```

您的新课程简介内容为

```
Hello, world\";select * from admin;\\\" where id=$cid
```

按任意键继续...

学生管理界面

st1同学您好, 欢迎来到现代作业管理系统 (Modern Coursework Manage System)

您本学期共3门课程, 它们分别为:

您可以进行的操作有:

- 1. 管理课程 ([提交/修改/删除作业](#))
- 2. 查看所有的作业/实验
- 0. [返回上一级](#)

请输入您想要进行的操作: 2

您选择了查看所有的作业和实验

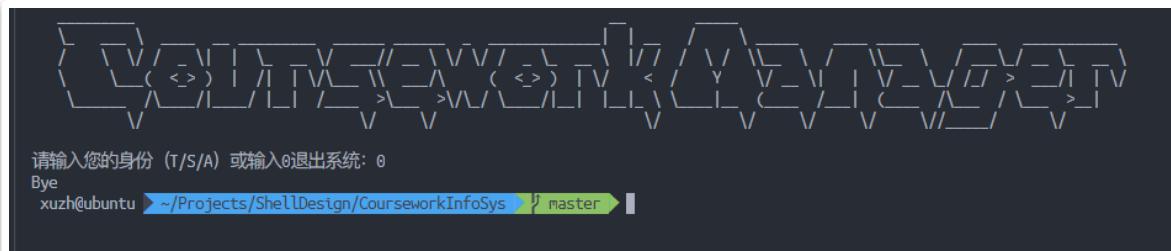
作业ID	作业简介	发布时间	截止时间	是否截止	是否完成	创建的提交数目
5	实验4 JDBC系统的编写和使用	2020-07-18 16:57:12	2020-07-25 16:57:12	是	否	0
6	第五周数据库系统作业	2020-07-18 16:57:12	2020-07-28 16:57:12	是	否	0
7	课程大作业 MySQL的编写与使用	2020-07-18 16:57:12	2020-08-07 16:57:12	否	否	0

按任意键继续...

发布作业提交功能测试



退出后界面



5

使用任何一种程序设计语言实现一个shell 程序的基本功能。

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

1. 这个shell 程序必须支持以下内部命令：`bg`、`cd`、`clr`、`dir`、`echo`、`exec`、`exit`、`environ`、`fg`、`help`、`jobs`、`pwd`、`quit`、`set`、`shift`、`test`、`time`、`umask`、`unset`。

部分命令解释如下：

1. `cd` ——把当前默认目录改变为。如果没有参数，则显示当前目录。如该目录不存在，会出现合适的错误信息。这个命令也可以改变PWD 环境变量。
2. `pwd` ——显示当前目录。
3. `time` ——显示当前时间。
4. `clr` ——清屏。
5. `dir` ——列出目录的内容。
6. `environ` ——列出所有的环境变量。
7. `echo` ——在屏幕上显示并换行（多个空格和制表符可能被缩减为一个空格）。
8. `help` ——显示用户手册，并且使用`more` 命令过滤。
9. `quit` ——退出shell。
10. shell 的环境变量应该包含`shell=/myshell`，其中`/myshell` 是可执行程序shell 的完整路径（不是你的目录下的路径，而是它执行程序的路径）。
2. 其他的命令行输入被解释为程序调用，shell 创建并执行这个程序，并作为自己的子进程。程序的执行的环境变量包含一下条目：
`parent=/myshell`。

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

`myshell batchfile`

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

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

`programname arg1 arg2 < inputfile > outputfile`

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

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

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

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

6. 必须支持管道（|）操作。

7. 命令行提示符必须包含当前路径。

提示：

1. 你可以假定所有命令行参数（包括重定向字符<、>、>>和后台执行字符&）和其他命令行参数用空白空间分开，空白空间可以为一个或多个空格或制表符（见上面（四）中的命令行）。

项目要求：

1. 设计一个简单的全新命令行shell，至少满足上面的要求并且在指定的Linux 平台上执行。拒绝使用已有的shell程序的任何环境及功能。严禁使用开源代码。
2. 写一个关于如何使用shell 的简单的用户手册，用户手册应该包含足够的细节以方便Linux初学者使用。例如：你应该解释I/O 重定向、管道、程序环境和后台程序执行。
3. 源代码必须有很详细的注释，并且有很好的组织结构以方便别人阅读和维护；否则会扣除客观的分数。结构和注释好的程序更加易于理解，并且可以保证批改你作业的人不用很费劲地去读你的代码。

实验报告内容包括：

源代码文件、必要设计文档和用户手册，所有提交的文件以文本形式复制到报告内；还有myshell上测试各种命令的运行结果截图。

需求描述

设计文档

shell 或者命令行解释器是操作系统中最基本的用户接口。我们实现了一个跨平台的简单的shell 程序——**MyShell**，它具有以下属性：

1. 支持的内部指令集：`cd, clr, pwd, dir, echo, exit, quit, jobs, fg, bg, term, environ, set, unset, umask, printio, exec, shift, test, sleep, time, help, verbose`

- `cd`

更改工作目录

`cd [target]`

- 无参数调用时会打印当前工作目录
- 传入一个参数调用时会尝试进入参数所示的目录
- 在各平台上都可正常使用
- 无法进入不存在的目录/或根本不是目录的路径/没有权限进入的路径

- `clr`

清空屏幕

`clr`

- 本指令没有参数
- 本指令需要调用系统相关命令以管理终端屏幕

- `pwd`

打印当前工作目录

`pwd [-a]`

- 无参数调用时会打印当前工作目录，用户根目录以 `~` 显示
- 传入参数 `-a` 调用时会打印当前工作目录完整路径

- `dir`

列举文件夹内容

`dir [target [target ...]]`

- 无参数调用时会显示当前目录下的文件列表
- 传入多个目录时会依次显示目录的列举结果，结果中多个目录间以空行分隔
- 对于每个目录，结果的第一行是下面将要现实的目录路径
- 普通文件加粗显示，可执行文件以红色粗体显示，目录以蓝色粗体显示
- 目录中的文件列表前以 `rwxrwxrwx` 格式显示文件/目录权限
- 目录中的文件列表前显示的时间是最近修改时间
- 若用户参数中有无法显示的目录（不存在/非目录/无权限等），会导致程序运行错误，此时将无法使用管道，但我们会打印出可以显示的那些目录的内容。

- `echo`

打印内容

`echo [-r] [content [content ...]]`

- 无参数调用时会打印空字符串
- 传入多个参数（除了开头的 `-r`）时会用空格分隔它们，并打印
- 传入的参数可以通过双引号包裹，被包裹的内容被视为一个整体
- 参数中可以包含 `~` 字符，会被替换为用户的根目录
- 参数中可以包含 `$...` 代表的变量，会被替换为相应的变量值，变量不存在时替换为空字符串
- 引号可以用于区分变量和普通内容，例如 `echo`

`PATH_TO_SHELL"$MESSAGE"` 只有一个参数，但是变量 `$MESSAGE` 会被正确处理

- 若要打印 `$` 符号，请输入 `\$` 以转义
- 若要打印 `~` 符号，请输入 `\~` 以转义
- 不采用 `-r` 开关时，会尝试转义传入字符中的可转义内容，例如调用 `echo`
- `"\033[1m\033[31mHello, world.\033[0m"` 会以红色粗体打印 `Hello, world.`
- 加入 `-r` 参数后，上面的命令会以普通字体打印 `\033[1m\033[31mHello,`
- `world.\033[0m`

- `exit`

退出MyShell

`exit`

- 我们不会处理任何参数，因为MyShell是一个Python Object，所以没有系统返回值的概念
- 通过调用 `exit/quit/EOF` 退出是最安全的退出方式，因为这种情况下MyShell有机会清空还没有结束的后台工作

- `quit`

同 `exit`

- `jobs`

打印当前任务信息

`jobs`

- 我们不会处理任何参数
- 后台任务的格式为 `[i] status env command` 例如 `[0] suspended env dummy &`
- 已经被清除/已经完成的任务不会被显示
- 尝试读取内容的外部后台程序会直接获得EOF
- 任务信息是管理性质的信息，所以我们会忽略 `exec` 命令的设置，将任务管理结果直接打印到屏幕上

- `fg`

将后台任务提到前台执行

`fg job_number`

- 只接受一个参数
- 对于正在执行的后台任务，提到前台运行
- 通过外部命令的刷出的后台任务仍然不能获取输入，尝试读取内容的外部后台程序会直接获得EOF
- 对于因为获取输入而暂停执行的命令，继续命令的执行并阻塞前台主线程

- `bg`

继续后台程序的执行

`bg [job_number [job_number ...]]`

- 由于所有的暂停的后台任务都是因为尝试获取用户输入，继续在后台执行它们只会得到继续暂停的结果
- MyShell没有对快捷键操作进行处理，因此没有暂停正在运行的外部命令的功能

- `term`

终止后台任务的执行

`term [job_number [job_number ...]]`

- 对于后台任务进程（`multiprocessing.Process`），发出 `SIGTERM` 信号以终止运行；后台任务会自动处理信号并终止自身运行
- 若后台任务不是内部命令，会对其子进程发出 `SIGKILL` 信号以尝试终止运行

- `environ`

打印MyShell全部内部变量

`environ`

- MyShell使用了内部的变量处理机制，在系统环境变量上加了一层额外的接口用以满足更严苛的测试环境。
- `0, 1, 2, 3, 4, 5, 6, 7, 8, 9` 是MyShell的保留变量，不能被修改和删除

- `set`

修改环境变量/设置新的环境变量

`set key=value [key=value ...]`

- 键值对以等于号配对，等于号的周围不允许出现空格，否则无法正常赋值
- `0, 1, 2, 3, 4, 5, 6, 7, 8, 9` 是MyShell的保留变量，不能被修改（它们实际上也不存在）
- 修改 `PS1` 变量会导致命令提示符的提示符号被修改，其默认值为 `$` 美元符号
- 修改 `PWD` 等不会导致当前目录发生改变，但调用 `cd` 命令进入别的命令后 `PWD` 变量就会被修改到目录改变后的地址下

- 修改 `HOME` 变量会导致程序处理 `~` 的方式发生改变
- 修改 `USER` 等变量不会对命令提示符样式有影响，但可能会对其他使用到这些变量的程序有影响
- 修改 `PATH` 可以改变程序搜索可执行文件的路径

- `unset`

删除环境变量

```
unset key [key ...]
```

- `0, 1, 2, 3, 4, 5, 6, 7, 8, 9` 是MyShell的保留变量，不能被删除（它们实际上也不存在）
- 删除 `PS1` 变量会导致命令提示符采用默认值 `$`
- 删除 `HOME` 变量会导致程序无法正确处理 `~`
- 删除 `USER` 等变量不会对命令提示符样式有影响，但可能会对其他使用到这些变量的程序有影响

- `umask`

修改程序的 `umask` 值

```
umask [value]
```

- 在Windows上修改 `umask` 的效果较为奇怪
- 不传入参数的时候会显示当前的 `umask`
- 传入新的 `umask` 会被尝试以八进制解释，并设置为新的 `umask` 值
- Linux上普通文本文件的默认权限是 `0o666`，可执行文件为 `0o777`
- 在MyShell修改的 `umask` 值会影响其后的文件创建

- `printio`

打印当前的输入输出重定向目标

```
printio
```

- 本命令没有参数
- 本命令会打印当前MyShell的 `exec` 指令重定向目标
- 例如执行 `exec < dummy.mysh > result.out` 后调用 `printio` 会打印

```
1 FILE NUMBER OF INPUT FILE: 3, redirecting MyShell input to
<_io.TextIOWrapper name='dummy.mysh' mode='r' encoding='utf-8'>
2 FILE NUMBER OF INPUT FILE: 4, redirecting MyShell output to
<_io.TextIOWrapper name='result.out' mode='w' encoding='utf-8'>
```

- 由于 `printio` 的意义就在于查看当前的重定向路径，我们不会将其输入输出重定向，而是直接打印到屏幕上

- `exec`

调整Shell的默认输入输出源

```
exec [< input] [> output | >> output]
```

- 调用本函数的效果是：若 `exec < input > output` 类似于在下面执行的每一条指令后都调用 `programname < input > output`。但对于输出文件，输出的内容会被累积，而非像调用 `> output` 那样完全覆盖
- 单独调用 `exec` 不会对输入输出产生任何影响，仅仅会调用 `printio` 检测当前IO状态
- 若在某次调用中只有使用 `[< input] [> output]` 的其中之一，另一个不会被改变
- 用户可以通过 `< ""`（传入空字符串）来清空输入源头，同样的，也可以用此方法清空输出源

- 值得注意的是，在手册中标注不会受到 `exec` 影响（保证打印到 `sys.__stdout__`）的程序总是会打印到 `sys.__stdout__`，例如任务管理工作 `jobs` 或者 `exec`, `printio` 本身等。
- 若使用的是 `>` 符号，则会在原有的文件内容基础上添加新的输出内容。
- `shift`

管理特殊环境变量 `1 ... 9`，移动变量的位置（管理命令行参数）

`shift [shamt]`

- 通过 `$0 ... $9` 可以访问脚本/程序执行时候的命令行参数
- `$0` 存储的是当前脚本的路径（脚本模式）/当前MyShell的路径（交互模式）
- `$0 ... $9` 不可以被修改/删除（他们实际上也不存在于环境变量中）
- 不带参数时，本命令可以让 `$1 ... $9` 获取下一个命令行参数，例如 `$1` 会获取 `$2` 的旧值
- 带参数时，移动一定的数量，例如传入参数1的效果与不传入相同，传入2会使得 `$1` 获得 `$3` 的原始值
- 调用 `shift` 命令不会修改 `$0` 的值
- 用户的参数必须要能够转换成整数类型

- `test`

测试表达式结果是否为真或假

`test expression`

- 支持的表达式：
 - `-o`：双目，逻辑或，参数为布尔值
 - `-a`：双目，逻辑与，参数为布尔值
 - `!`：单目，逻辑反，参数为布尔值
 - `-z`：单目，字符串长度零检查，参数为字符串
 - `-n`：单目，字符串长度非零检查，参数为字符串
 - `==`：双目，字符串相等性检擦，参数为字符串
 - `!=`：双目，字符出不等性检查，参数为字符串
 - `-eq`：双目，数值相等性检查，参数为浮点数/整数
 - `-ne`：双目，数值不等性检查，参数为浮点数/整数
 - `-gt`：双目，数值大于性检查 `lhs > rhs`，参数为浮点数/整数
 - `-lt`：双目，数值小于性检查 `lhs < rhs`，参数为浮点数/整数
 - `-ge`：双目，数值大于等于检查 `lhs ≥ rhs`，参数为浮点数/整数
 - `-le`：双目，数值小于等于检查 `lhs ≤ rhs`，参数为浮点数/整数
 - `(`：左括号：被括号包裹的内容会被当成一个表达式来解释，返回布尔值
 - `)`：右括号：被括号包裹的内容会被当成一个表达式来解释，返回布尔值
- 表达式和运算子必须用空白符分隔开
- 支持复杂的嵌套表达式，括号/单目运算符/双目运算符皆可嵌套执行

```
1 test ! -z "" -a ( -n "1" -o 1 -ge 1 ) -o 2 -ne 1 # False, -a -o
   from right to left
2 test ( ! -z "" -a ( -n "1" -o 1 -ge 1 ) ) -o 2 -ne 1 # True
```

- `-a, -o` 从右向左结合，但用户可以通过括号来定制它们的运算顺序

- 用户需要保证输入的内容是合理的可匹配的表达式
 - 括号需匹配完整
 - 运算符能处理的数据类型需要进行合理判断。所有经过运算的表达式结果：布尔值
- **sleep**

等待一定时间

sleep amount

- 在*nix系统下，会尝试调用系统 **sleep** 指令，能够识别很多不同类型的睡眠时长
- 在Windows下，会尝试调用Python 3的 **time.sleep**，支持以秒为单位的睡眠请求

- **time**

获取当前系统时间

time

- 以格式 "%Y-%m-%d %H:%M:%S.%f" 打印时间

- **help**

获取在线帮助信息，通过 **more/less** 指令过滤

help [command]

- 无参数时，打印**MyShell**用户文档
- 有参数时，打印相关指令的帮助文档，找不到**MyShell**内部文档时候会尝试调用系统的 **man** 指令

- **verbose**

调整**MyShell**的调试信息等级

verbose [-e|-w|-i|-d]

- **MyShell**的默认调试等级为：**DEBUG**，会打印程序运行和指令执行中的最详细信息
- 推荐的日常运行等级为：**WARNING**，也就是调用 **-w** 后的结果
- 无参数调用时会打印当前调试等级
- 有参数调用时会尝试切换调试等级
- 也可以在启动**MyShell**时传入类似格式的命令行参数来修改调试信息等级

2. 开发者调试指令集：**dummy**, **check_zombie**, **queues**，用户一般不需要调用这些指令

- **dummy**

输入输出测试程序

dummy

- 用于检查输入请求下的后台程序暂停是否被正常实现
- 用户可以调用一下这个指令，看看是作什么用的

- **check_zombie**

检查僵尸线程状态，打印 **daemon** 下等待主进程退出的进程

check_zombie

- 正常情况下，被手动终止的后台任务会出现在这里
- 类似的，这类任务管理指令会被直接输出到 **sys.__stdout__**

- **queues**

检查程序内部任务管理器的输入队列状态

queues

- 打印当前的后台任务输入队列生存状态
- 是开发者用于检查内存泄露的方式之一

3. MyShell在开始执行后会将环境变量 `SHELL` 设置为 `MyShell` 的运行位置
4. 其他的命令行输入被解释为程序调用，`MyShell`创建并执行这个程序，并作为自己的子进程。程序的执行的环境变量包含一下条目：
`PARENT=<pathname>/MyShell.py`（也就是`MyShell`中 `SHELL` 变量的内容）。
5. `MyShell`能够从文件中提取命令行输入，例如`shell` 使用以下命令行被调用：

```
1      ./MyShell.py dummy.mysh
```

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

6. `MyShell`除了上述的脚本执行，还支持其他运行时命令行参数：

用户可以调用 `./MyShell.py -h` 查看相关内容

```
1      usage: MyShell.py [-h] [-a [A [A ... ]]] [-e] [-w] [-i] [-d] [F]
2
3      MyShell by xudenden@gmail.com
4
5      positional arguments:
6          F                  the batch file to be executed
7
8      optional arguments:
9          -h, --help        show this help message and exit
10         -a [A [A ... ]]  command line arguments to batch file
11         -e                enable error level debugging info log
12         -w                enable warning level debugging info log
13         -i                enable info level debugging info log
14         -d                enable debug(verbose) level debugging info log
```

`MyShell`的调用举例：

```
1      ./MyShell.py -w dummy.mysh -a foo bar foobar hello world linux linus PyTorch
CS231n
```

7. `MyShell`支持I/O 重定向，`stdin` 和`stdout`，或者其中之一，例如命令行为：

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

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

`stdout` 重定向持除了在上面注明需要打印信息（后台任务管理，输入输出重定向查看等）的所有内部指令。

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

对于 `exec` 指令，使用重定向符号会导致`MyShell`的输入输出被调整到指定的文件。

`MyShell`在处理外部程序的调用时，为了方便用户观察结果和控制输入输出，会将 `stderr` 重定向到 `stdout` 一并打印到屏幕/定义的输出文件流。

8. `MyShell`支持后台程序执行。如果在命令行后添加 `&` 字符，在加载完程序后需要立刻返回命令行提示符。

后台程序的主要管理接口为 `jobs, term, fg, bg`

值得注意的是，通过 `subprocess` 调用的外部后台程序的输入端口是关闭的

内部指令的输入请求会触发后台任务的暂停操作

MyShell退出时会尝试清空所有正在运行的后台任务

9. MyShell支持管道 (“|”) 操作。

在MyShell中管道和输出重定向可以同时使用而不冲突

但输入管道和输入重定向不可同时使用

使用管道的指令举例 (请保证 `sha256sum` 指令是可用的) :

```
1      cat < dummy.mysh | wc > /dev/tty | echo "zy" > result.out | sha256sum | tr -d " -" >> result.out | wc | cat result.out | wc | cat result.out
```

应该会打印类似如下的内容

```
1      159      561      2940
2      zy
3      49aabdaa1b0f6c3506f54521ef81fe5b5fe835d268f1f86e1021a342b59d43bc
```

10. MyShell的命令提示符包含以下内容：

```
1      ($CONDA_DEFAULT_ENV) $USER@location $PWD time("%H:%M:%S") $PS1
```

分别为 :

- 括号内的Anaconda环境
- 用户名和登陆位置名
- 当前路径 (用~替换 `$HOME` 的内容)
- 当前时间 (时:分:秒)
- 命令提示符符号

11. MyShell支持详细的调试信息打印，详见 `verbose` 命令的帮助手册

一般来说我们有四种类型的信息打印：

```
1      1. `DEBUG` 调试信息：非开发者可以忽略的调试信息，用于监测MyShell内部运行状态
2      2. `INFO` 一般信息：一般性的记录信息，大部分情况下可以忽略
3      3. `WARNING` 警告信息：一般在警告中出现，子进程非零退出，进程管理以及找不到的环境变量等
4      4. `ERROR` 错误信息：指令格式/运行时错误
```

可以在开启MyShell时通过传入命令行参数开关 `-e, -w, -i, -d` 来调整等级。

也可以在MyShell运行时通过调用 `verbose` 指令来实现

12. MyShell支持颜色/字体调整，我们会调整输出颜色等，使其尽量容易辨识，做到用户友好

例如在命令提示符中，我们会用不同的颜色/字体区分提示符的不同部分

值得注意的是，用户的终端需要支持颜色输出才能正常显示相关字符，否则会有难以预料的输出错误

我们的测试基本都是在Visual Studio Code通过SSH连接Ubuntu下执行的颜色信息的显示较为友好

```
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 14:46:43 $ dir
.
-rw-rw-r-- 2020-07-29 14:28:40 log.log
-rw-rw-r-- 2020-07-29 12:57:15 should.out
-rw-rw-r-- 2020-07-29 12:36:49 MyShell.md
-rw-rw-r-- 2020-07-29 14:46:43 result.out
-rw-rw-r-- 2020-07-25 12:11:10 dummy.py
-rwxrwxr-x 2020-07-25 14:09:29 process.py
-rw-rw-r-- 2020-07-26 17:43:58 sleep10s.py
-rwxrwxr-x 2020-07-29 14:52:17 MyShell.py
-rw-rw-r-- 2020-07-29 12:56:43 dummy.mysh
-rw-rw-r-- 2020-07-22 19:23:17 COLOR.py
-rw-rw-r-- 2020-07-29 10:43:24 MyShellException.py
-rwxrwxr-x 2020-07-29 12:42:10 __pycache__
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 15:11:03 $ verbose -d
2020-07-29 15:11:08 ubuntu __main__ [60688] DEBUG Queue is being cleaned, previous keys are []
2020-07-29 15:11:08 ubuntu __main__ [60688] DEBUG Queue is cleaned, keys are []
2020-07-29 15:11:08 ubuntu __main__ [60688] DEBUG Process is being cleaned, previous keys are []
2020-07-29 15:11:08 ubuntu __main__ [60688] DEBUG Process is cleaned, keys are []
2020-07-29 15:11:08 ubuntu __main__ [60688] DEBUG Status Dict is being cleaned, previous keys are []
2020-07-29 15:11:08 ubuntu __main__ [60688] DEBUG Status Dict is cleaned, keys are []
2020-07-29 15:11:08 ubuntu __main__ [60688] DEBUG Got the varible HOME as /home/xuzh
2020-07-29 15:11:08 ubuntu __main__ [60688] DEBUG Got the varible HOME as /home/xuzh
2020-07-29 15:11:08 ubuntu __main__ [60688] DEBUG Got the varible PS1 as $
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 15:11:08 $
```

13. MyShell支持定制化指令：我们使用Python实现MyShell。并在内部指令中做了统一的接口

```
1  def builtin_foo(self, pipe="", args=[]):
2      # do something
3      # print things that doesn't go to pipe
4      # print to sys._stdout_ to always print to STDOUT
5      # return strings that go into the pipe
6      return result
```

用户只需要定义新的以 `builtin_` (注意下划线) 开头的MyShell方法 (包含 `pipe` 和 `args` 参数) 即可添加内部指令，并无缝融入程序的运行中。

例如：

```
1  def builtin_dummy(self, pipe="", args=[]):
2      # 一个内置的dummy命令，用于测试是否可以正常触发suspension
3      print("builtin_dummy: before any input requirements")
4      print(input("dummy1> "))
5      print(input("dummy2> "))
6      print(input("dummy3> "))
7      print(input("dummyend> "))
8      result = input("dummy_content> ")
9      return result
```

14. MyShell本体可以跨平台运行，后台任务的主体功能也可以在Windows上运行。

可以运行的环境信息在本报告的前半部分已经列举过

在第一次运行时Python 3或许会抱怨有一些包找不到，此时请通过 `pip` 来安装相关缺失的内容

若 `pip` 速度过慢，用户可以使用 清华源 来提速

MyShell在Windows上运行的情况：

```
(D:\condaenvs\anaconda) Xuzh@DESKTOP-XUZH D:\Documents\Materials\DENDEN\PROJ\Linux\ShellDesign\MyShell 15:18:09 $ systeminfo
2020-07-29 15:18:11 DESKTOP-XUZH __main__ [20108] DEBUG Getting user input: systeminfo
2020-07-29 15:18:11 DESKTOP-XUZH __main__ [20108] INFO Executing command systeminfo
2020-07-29 15:18:11 DESKTOP-XUZH __main__ [20108] INFO Arguments are []
2020-07-29 15:18:11 DESKTOP-XUZH __main__ [20108] DEBUG This is not a builtin command.
2020-07-29 15:18:11 DESKTOP-XUZH __main__ [20108] DEBUG Running in subprocess: ['systeminfo']
2020-07-29 15:18:11 DESKTOP-XUZH __main__ [20108] DEBUG EXEC OI controller is: None None
2020-07-29 15:18:11 DESKTOP-XUZH __main__ [20108] DEBUG Got the varible SHELL as D:\Documents\Materials\DENDEN\PROJ\Linux\ShellDesign\MyShell\MyShell.py

Host Name: DESKTOP-XUZH
OS Name: Microsoft Windows 10 Pro
OS Version: 10.0.18363 N/A Build 18363
OS Manufacturer: Microsoft Corporation
OS Configuration: Standalone Workstation
OS Build Type: Multiprocessor Free
Registered Owner: 56295612@qq.com
Registered Organization: N/A
Product ID: 00330-80000-00000-AA645
Original Install Date: 8/21/2019, 12:44:19 PM
System Boot Time: 7/29/2020, 10:14:54 AM
System Manufacturer: Dell Inc.
System Model: Inspiron 7500
System Type: x64-based PC
Processor(s): 1 Processor(s) Installed.
[01]: Intel64 Family 6 Model 158 Stepping 10 GenuineIntel ~2592 Mhz
BIOS Version: Dell Inc. 1.6.0, 2/7/2020
```

```
(D:\condaenvs\anaconda) Xuzh@DESKTOP-XUZH D:\Documents\Materials\DENDEN\PROJ\Linux\ShellDesign\MyShell 15:18:14 $ dir
2020-07-29 15:18:20 DESKTOP-XUZH __main__ [20108] DEBUG Getting user input: dir
2020-07-29 15:18:20 DESKTOP-XUZH __main__ [20108] INFO Executing command dir
2020-07-29 15:18:20 DESKTOP-XUZH __main__ [20108] INFO Arguments are []
2020-07-29 15:18:20 DESKTOP-XUZH __main__ [20108] DEBUG This is a builtin command.
2020-07-29 15:18:20 DESKTOP-XUZH __main__ [20108] DEBUG Listing dir .

rW-rw-rw- 2020-07-22 19:26:43 COLOR.py
rW-rw-rw- 2020-07-28 17:17:15 dummy.mysl
rW-rw-rw- 2020-07-25 10:58:47 dummy.py
rW-rw-rw- 2020-07-26 16:32:22 log.log
rwxrwxrwx 2020-07-29 15:11:50 MyShell.assets
rW-rw-rw- 2020-07-29 15:16:27 MyShell.md
rW-rw-rw- 2020-07-29 12:35:40 MyShell.py
rW-rw-rw- 2020-07-29 12:34:36 MyShellException.py
rW-rw-rw- 2020-07-25 14:24:12 process.py
rW-rw-rw- 2020-07-29 12:34:36 should.out
rW-rw-rw- 2020-07-26 17:55:59 sleep10s.py
rwxrwxrwx 2020-07-29 12:35:19 __pycache__
2020-07-29 15:18:20 DESKTOP-XUZH __main__ [20108] DEBUG Queue is being cleaned, previous keys are []
2020-07-29 15:18:20 DESKTOP-XUZH __main__ [20108] DEBUG Queue is cleaned, keys are []
2020-07-29 15:18:20 DESKTOP-XUZH __main__ [20108] DEBUG Process is being cleaned, previous keys are []
2020-07-29 15:18:20 DESKTOP-XUZH __main__ [20108] DEBUG Process is cleaned, keys are []
2020-07-29 15:18:20 DESKTOP-XUZH __main__ [20108] DEBUG Status Dict is being cleaned, previous keys are []
2020-07-29 15:18:20 DESKTOP-XUZH __main__ [20108] DEBUG Status Dict is cleaned, keys are []
2020-07-29 15:18:20 DESKTOP-XUZH __main__ [20108] DEBUG Got the varible HOME as C:\Users\56295
2020-07-29 15:18:20 DESKTOP-XUZH __main__ [20108] DEBUG Got the varible PS1 as $
```

15. MyShell有较为完备的内置报错系统

在用户调用的命令出错时，我们会通过 `exception` 机制快速定位错误源头，并通过 `logging` 模块以人性化的方式打印相关信息

```
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 11:38:12 $ di
2020-07-30 11:38:13 ubuntu __main__ [65482] ERROR Cannot successfully execute command "di". Exception is:
FileNotFoundException: [Errno 2] No such file or directory: 'di': 'di'
Extra info: {'type': 'subshell'}
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 11:38:13 $ cd ...
2020-07-30 11:38:22 ubuntu __main__ [65482] ERROR Cannot successfully execute command "cd". Exception is:
FileNotFoundException: [Errno 2] No such file or directory: '...'
Extra info: {'type': 'cd'}
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 11:38:22 $ test ttt
True
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 11:38:28 $ test ! ttt
False
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 11:38:33 $ test ! ! ttt
True
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 11:38:38 $ test ! ( ttt
2020-07-30 11:38:41 ubuntu __main__ [65482] ERROR Cannot successfully execute command "test". Exception is:
TestException: Unrecognized test expression, check your syntax. list index out of range
Extra info: None
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 11:38:41 $ set 0=0
2020-07-30 11:38:53 ubuntu __main__ [65482] ERROR Cannot successfully execute command "set". Exception is:
ReservedKeyException: Key 0 is reserved, along with "[0', '1', '2', '3', '4', '5', '6', '7', '8', '9']"
Extra info: None
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 11:38:53 $
```

16. MyShell支持以 # 开头的注释，注意 注释符号前必须是空白字符

```
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:31:05 $ # This is some comment
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:33:20 $ echo This is not a comment#
This is not a comment#
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:33:35 $ echo This is a comment # I'm comment!
This is a comment
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:34:11 $
```

17. MyShell支持输入历史记录的记录，用户可以通过上下方向键访问他们上次输入的内容。

用户的左右方向键也可以被成功识别为移动光标的请求。

用户手册

程序IO重定向

Linux中的命令行程序以输入输出为主要信息交互方式，shell是Linux系列系统的基本接口之一，用户经常需要在各种各样的shell下运行不同的程序，并观察他们的输入输出结果。因此控制输入输出是shell的基本功能之一。

一般情况下，程序会从连接到终端的键盘设备（`stdin: /dev/tty`）读取用户的输入内容，并将输出内容打印到终端的屏幕上（`stdout: /dev/tty`）。

但若用户并不希望某个在shell环境下运行的程序从标准输入中`stdin: /dev/tty`读入内容，它可以通过输入重定向符号`<`来改变shell下程序的输入源。类似的，shell也提供输出重定向功能。一般的，若程序以`programname args < inputfile > outputfile`的形式被调用，它会从`inputfile`中读取内容，并将标准输出导入到`outputfile`中。

对于输入文件流，不同于通过键盘读取的标准输出，在`inputfile`的内容被读完时，程序将会获得`EOF`信号，而非被停止并等待输入。

对于输出文件流，若使用的重定向符号为`>`，则会创建新的`outputfile`文件/覆盖原有内容。若为`>>`，则在`outputfile`已经存在或有文件内容的情况下，会在保留原有文件的基础上在文件末尾添加新的内容。

程序管道

正如上面所说的，管道其实也是输入输出重定向的一种。

不同于引导输入输出到文件，程序管道直接将上一个程序的输出作为本程序的输入，而本程序的输出会被看作下一个程序的输出。

类似的，输入内容耗尽时会获得`EOF`。

相对于通过文件进行交互，程序管道无需显式地对文件进行操作：这意味着其运行速度会快于通过重定向到文件。

`prog1 | prog2 | prog3` 的调用效果相当于：

```
1  prog1 > file1
2  prog2 < file1 > file2
3  prog3 < file2
```

程序的运行环境

在操作系统中，程序的运行环境也是控制程序运行方式的一种重要方式。

例如，我们熟悉的`PATH`环境变量就可以指导shell到相应的文件夹中寻找可执行文件来运行。

在一些深度学习环境中，`CUDA`相关环境变量可以控制相应程序对Nvidia CUDA的操作方式。

`HOME`环境变量还控制着shell对`~`符号的解释。

在Linux相关的shell脚本中，这些环境变量还被当作一般的变量来使用。例如我们可以将一些特殊的颜色字符储存到一个环境变量中，在以后调用相关程序需要打印相关颜色时，可以直接使用 \$COLOR

后台程序执行

许多Linux Shell支持基于任务管理的多线程程序执行功能。我们可以通过在程序命令行末尾添加 & 来让程序在后台执行（特别是一些需要较长时间才能完成的程序），而立刻返回到可交互的命令行来输入其他命令。在程序完成后/状态发生改变时在shell中以一定的方式提示用户。

这种方式理论上可以管理无限多的后台程序。用户可以通过 jobs, bg, fg 等命令来查看/管理正在后台执行的程序。

在一些较为完备的shell中，键盘快捷键得到了很好的支持，用户可以通过 `Ctrl+Z` 来暂停/挂起正在执行的程序，并通过 `bg` 让其在后台恢复运行/ `fg` 让其恢复运行并提到前台。并且支持根据输入的程序暂停功能：在程序读取输入流时自动挂起。

运行结果

1. 复杂重定向和管道操作

```
1      cat < dummy.mysh | wc > /dev/tty | echo "zy" > result.out | sha256sum | tr -d " -" >> result.out | wc | cat result.out | wc | cat result.out
```

```
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:26:07 $ dir
.
-rwxrwxr-x 2020-07-30 12:08:45 hello
-rw-rw-r-- 2020-07-29 14:28:40 log.log
-rw-rw-r-- 2020-07-29 12:57:15 should.out
-rw-rw-r-- 2020-07-29 12:36:49 MyShell.md
-rw-rw-r-- 2020-07-25 12:11:10 dummy.py
-rwxrwxr-x 2020-07-25 14:09:29 process.py
-rw-rw-r-- 2020-07-30 12:08:43 hello.c
-rw-rw-r-- 2020-07-26 17:43:58 sleep10s.py
-rwxrwxr-x 2020-07-29 14:52:17 MyShell.py
-rw-rw-r-- 2020-07-29 12:56:43 dummy.mysh
-rw-rw-r-- 2020-07-22 19:23:17 COLOR.py
-rw-rw-r-- 2020-07-29 10:43:24 MyShellException.py
-rwxrwxr-x 2020-07-29 12:42:10 __pycache__
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:26:09 $ cat < dummy.mysh | wc > /dev/tty | echo "zy" > result.out | sha256sum | tr -d " -" >> result.out | wc | cat result.out | wc | cat result.out
159      561     2940
zy
49aabdaa1b0f6c3506f54521ef81fe5b5fe835d268f1f86e1021a342b59d43bc
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:26:17 $ dir
.
-rwxrwxr-x 2020-07-30 12:08:45 hello
-rw-rw-r-- 2020-07-29 14:28:40 log.log
-rw-rw-r-- 2020-07-29 12:57:15 should.out
-rw-rw-r-- 2020-07-29 12:36:49 MyShell.md
-rw-rw-r-- 2020-07-30 12:26:17 result.out
-rw-rw-r-- 2020-07-25 12:11:10 dummy.py
-rwxrwxr-x 2020-07-25 14:09:29 process.py
-rw-rw-r-- 2020-07-30 12:08:43 hello.c
-rw-rw-r-- 2020-07-26 17:43:58 sleep10s.py
-rwxrwxr-x 2020-07-29 14:52:17 MyShell.py
-rw-rw-r-- 2020-07-29 12:56:43 dummy.mysh
-rw-rw-r-- 2020-07-22 19:23:17 COLOR.py
-rw-rw-r-- 2020-07-29 10:43:24 MyShellException.py
-rwxrwxr-x 2020-07-29 12:42:10 __pycache__
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:26:36 $
```

2. 脚本执行

```
1      ./MyShell.py -w dummy.mysh -a foo bar foobar hello world linux linus PyTorch
CS231n
```

脚本内容 `dummy.mysh`：

```
1      test ! -z ""
2      date +%
```

```
3   time
4   set BOLD="\033[1m"
5   set RED="\033[31m"
6   set BLUE="\033[34m"
7   set RESET="\033[0m"
8
9   set
10  LINE="$BOLD$BLUE#####
11  #####$RESET"
12
13  umask
14  echo "Changing UMASK to 0e777"
15  umask 777
16  umask
17  echo "Changing UMASK to 0e002"
18  umask 002
19
20  echo $LINE
21
22  echo "Hello, my name is $SHELL"
23  set hello_msg="Hello, my name is"
24  echo -r "$hello_msg $USER, and I live in $HOME"
25  echo "Should print sha256sum of zy in the next line"
26  echo "xz" | sha256sum | tr -d " -"
27  echo "Should print 1 1 65 in the following line"
28  echo "zy" | sha256sum | tr -d " -" | wc
29
30  echo $LINE
31
32  dir
33  pwd
34  unset hello_msg
35  echo "Should get empty output"
36  echo $hello_msg
37
38  echo $LINE
39
40  echo "Should print /dev/null"
41  ls /dev | grep null
42  echo "Should print all files containing 1 in /tmp"
43  ls /tmp | grep 1
44
45  echo $LINE
46
47  echo "Should make a file log.log"
48  echo "Hello, I'm your logger." > log.log
49  dir
50  echo "Should see content of log.log"
51  cat < log.log
52  echo "Hello, again..." >> log.log
53  echo "Should display content of log.log"
54  cat log.log
55  echo "Should display word count of log.log"
56  wc < log.log
57
58  echo $LINE
```

```

59
60     echo "Opening some sleepy jobs"
61     echo "And calling command jobs"
62     sleep 2s | echo "Sleeping in $0" &
63     echo "waiting 0.25s"
64     jobs
65     sleep 0.25s
66     sleep 2s | echo "This is some job management" &
67     echo "waiting 0.25s"
68     jobs
69     sleep 0.25s
70     sleep 2s | echo "MyShell is $SHELL" &
71     echo "waiting 0.25s"
72     jobs
73     sleep 0.25s
74     sleep 2s &
75     echo "waiting 0.25s"
76     jobs
77     sleep 0.25s
78     sleep 2s &
79     echo "Getting current runnning jobs ... "
80     jobs
81
82     echo "Getting back to fore ground"
83     echo "Waiting for background jobs to terminate"
84     echo "At the same time I can still do other things like testing ... "
85     test "" -o "a"
86     test ! -z "a" -a ( -n "1" -o 1 -ge 1 ) -a 2 -ne 1
87     test ! -z "" -a ( -n "1" -o 1 -ge 1 ) -o 2 -ne 1 # False, -a -o from
     right to left
88     test ( ! -z "" -a ( -n "1" -o 1 -ge 1 ) ) -o 2 -ne 1 # True
89     sleep 2s
90
91     echo "Should produce empty content"
92     jobs
93
94     echo "Jobs are done ~"
95
96     echo "Spawning dummy built_in job that is trying to read from user (will
     suspend)"
97
98     dummy &
99     dummy &
100    dummy &
101    dummy &
102
103    echo "Counting jobs"
104    jobs
105
106    echo "$RED$BOLD""WE'RE ONLY TERMINATING JOB [0] AND [1], YOU SHOULD SEE
     WARMING IF -w. NO ZOMBIE""$RESET"
107
108    term 0 1
109
110    echo $LINE
111
112    echo "calling environ ... "
113    environ

```

```
114
115 echo "Arg 0 is: $0"
116 echo "Arg 1 is: $1"
117 echo "Arg 2 is: $2"
118 echo "Arg 3 is: $3"
119 echo "Arg 4 is: $4"
120 echo "Arg 5 is: $5"
121 echo "Arg 6 is: $6"
122 echo "Arg 7 is: $7"
123 echo "Arg 8 is: $8"
124 echo "Arg 9 is: $9"
125
126 echo "Shifting number 1"
127 shift
128 echo "Arg 0 is: $0"
129 echo "Arg 1 is: $1"
130 echo "Arg 2 is: $2"
131 echo "Arg 3 is: $3"
132 echo "Arg 4 is: $4"
133 echo "Arg 5 is: $5"
134 echo "Arg 6 is: $6"
135 echo "Arg 7 is: $7"
136 echo "Arg 8 is: $8"
137 echo "Arg 9 is: $9"
138
139 echo "Shifting number 2"
140 shift
141 echo "Arg 1 is: $1"
142
143 echo "Shifting number 3"
144 shift
145 echo "Arg 1 is: $1"
146
147 echo "Shifting number 4"
148 shift
149 echo "Arg 1 is: $1"
150
151 echo "Shifting number 5"
152 shift
153 echo "Arg 1 is: $1"
154 echo "$BOLD$Bye! $RESET"
```

执行结果：

```

xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:28:34 $ exit
xuzh@ubuntu ~/Projects/ShellDesign/MyShell > master > ./MyShell.py -w dummy.mysh -a foo bar foob
ar hello world linux linus PyTorch CS231n
False
1596083355
2020-07-30 12:29:15.712186
#####
0o002
Changing UMASK to 0o777
0o777
Changing UMASK to 0o002
#####
Hello, my name is /home/xuzh/Projects/ShellDesign/MyShell/MyShell.py
Hello, my name is xuzh, and I live in /home/xuzh
Should print sha256sum of zy in the next line
b44f7d6b5283a44ee5f2bd98f84087a04810092122d75e8fbf8ad85f8f2981f1
Should print 1 1 65 in the following line
    1      1      65
#####
.
rwxrwxr-x 2020-07-30 12:08:45 hello
rw-rw-r-- 2020-07-29 14:28:40 log.log
rw-rw-r-- 2020-07-29 12:57:15 should.out
rw-rw-r-- 2020-07-29 12:36:49 MyShell.md
rw-rw-r-- 2020-07-30 12:26:17 result.out
rw-rw-r-- 2020-07-25 12:11:10 dummy.py
rwxrwxr-x 2020-07-25 14:09:29 process.py
rw-rw-r-- 2020-07-30 12:08:43 hello.c
rw-rw-r-- 2020-07-26 17:43:58 sleep10s.py
rwxrwxr-x 2020-07-29 14:52:17 MyShell.py
rw-rw-r-- 2020-07-29 12:56:43 dummy.mysh
rw-rw-r-- 2020-07-22 19:23:17 COLOR.py
rw-rw-r-- 2020-07-29 10:43:24 MyShellException.py
rwxrwxr-x 2020-07-29 12:42:10 __pycache__
~/Projects/ShellDesign/MyShell
Should get empty output
2020-07-30 12:29:15 ubuntu __main__[66739] WARNING Unable to get the varible "hello_msg", assigning e
mpty string
#####

```

...

```

1 False
2 1596083355
3 2020-07-30 12:29:15.712186
4 #####
5 0o002
6 Changing UMASK to 0o777
7 0o777
8 Changing UMASK to 0o002
9 #####
10 Hello, my name is /home/xuzh/Projects/ShellDesign/MyShell/MyShell.py
11 Hello, my name is xuzh, and I live in /home/xuzh
12 Should print sha256sum of zy in the next line
13 b44f7d6b5283a44ee5f2bd98f84087a04810092122d75e8fbf8ad85f8f2981f1
14 Should print 1 1 65 in the following line
15     1      1      65
16 #####
17 .
18 rwxrwxr-x 2020-07-30 12:08:45 hello
19 rw-rw-r-- 2020-07-29 14:28:40 log.log
20 rw-rw-r-- 2020-07-29 12:57:15 should.out
21 rw-rw-r-- 2020-07-29 12:36:49 MyShell.md
22 rw-rw-r-- 2020-07-30 12:26:17 result.out
23 rw-rw-r-- 2020-07-25 12:11:10 dummy.py
24 rwxrwxr-x 2020-07-25 14:09:29 process.py
25 rw-rw-r-- 2020-07-30 12:08:43 hello.c
26 rw-rw-r-- 2020-07-26 17:43:58 sleep10s.py
27 rwxrwxr-x 2020-07-29 14:52:17 MyShell.py

```

```
28 rw-rw-r-- 2020-07-29 12:56:43 dummy.mysh
29 rw-rw-r-- 2020-07-22 19:23:17 COLOR.py
30 rw-rw-r-- 2020-07-29 10:43:24 MyShellException.py
31 rwxrwxr-x 2020-07-29 12:42:10 __pycache__
32 ~/Projects/ShellDesign/MyShell
33 Should get empty output
34 2020-07-30 12:29:15 ubuntu __main__ [66739] WARNING Unable to get the
variable "hello_msg", assigning empty string
35 #####
36 Should print /dev/null
37 null
38 Should print all files containing 1 in /tmp
39 clr-debug-pipe-64261-7288941-in
40 clr-debug-pipe-64261-7288941-out
41 dotnet-diagnostic-64261-7288941-socket
42 pymp-1yf6wit3
43 pymp-i1l0r1ba
44 ssh-3Dl1dZ2MVhMN
45 ssh-5opVjzIOSY1n
46 ssh-F4yJVhHqoOo1
47 ssh-ssJdLj12gJhT
48 systemd-private-32fa36ac417343a4813881b03c5a5a50-bolt.service-9WJsxv
49 systemd-private-32fa36ac417343a4813881b03c5a5a50-colord.service-qxRv3v
50 systemd-private-32fa36ac417343a4813881b03c5a5a50-fwupd.service-WNl3wS
51 systemd-private-32fa36ac417343a4813881b03c5a5a50-ModemManager.service-
w89ayg
52 systemd-private-32fa36ac417343a4813881b03c5a5a50-rtkit-daemon.service-
ebfv3E
53 systemd-private-32fa36ac417343a4813881b03c5a5a50-systemd-
resolved.service-G0nqb0
54 systemd-private-32fa36ac417343a4813881b03c5a5a50-systemd-
timesyncd.service-N7TNb0
55 tmp-64015c9VBFgG403ca.tpl
56 tmp-64015EunG2vwQ362u.tpl
57 tmp-64015jJWFioLuj9YL.tpl
58 vmware-root_1361-3988687315
59 vscode-ipc-0e8469e9-1fc8-467e-b42c-f6a8519ab561.sock
60 vscode-ipc-705a4712-c1f7-43b0-9465-99c771a42a1d.sock
61 vscode-ipc-76bfc346-b068-4fb1-8340-103432500ccb.sock
62 vscode-ipc-77ad59b8-138f-4875-884e-368ed7e31e7d.sock
63 vscode-ipc-846c3b8a-cd23-429d-aa84-b71bb5acbe7b.sock
64 vscode-ipc-a1c81be6-faa1-49b2-8a12-38a172ae37e9.sock
65 vscode-ipc-cb722aa7-90c1-4b82-b595-52d2e62b5d98.sock
66 vscode-ipc-e9b20dcfd-2866-4d0b-bd69-426b2dc6b153.sock
67 vscode-ipc-efde013b-3ec4-4d77-9e0d-b10d25879dda.sock
68 vscode-ipc-typescript1000
69 #####
70 Should make a file log.log
71 .
72 .
73 rwxrwxr-x 2020-07-30 12:08:45 hello
74 rw-rw-r-- 2020-07-30 12:29:15 log.log
75 rw-rw-r-- 2020-07-29 12:57:15 should.out
76 rw-rw-r-- 2020-07-29 12:36:49 MyShell.md
77 rw-rw-r-- 2020-07-30 12:26:17 result.out
78 rw-rw-r-- 2020-07-25 12:11:10 dummy.py
79 rwxrwxr-x 2020-07-25 14:09:29 process.py
80 rw-rw-r-- 2020-07-30 12:08:43 hello.c
```

```
81    rw-rw-r-- 2020-07-26 17:43:58 sleep10s.py
82    rwxrwxr-x 2020-07-29 14:52:17 MyShell.py
83    rw-rw-r-- 2020-07-29 12:56:43 dummy.mysh
84    rw-rw-r-- 2020-07-22 19:23:17 COLOR.py
85    rw-rw-r-- 2020-07-29 10:43:24 MyShellException.py
86    rwxrwxr-x 2020-07-29 12:42:10 __pycache__
87    Should see content of log.log
88    Hello, I'm your logger.
89    Should display content of log.log
90    Hello, I'm your logger.
91    Hello, again ...
92    Should display word count of log.log
93        2       6      40
94    #####
95    Opening some sleepy jobs
96    And calling command jobs
97    waiting 0.25s
98    [0] running env sleep 2s | echo "Sleeping in $0" &
99    waiting 0.25s
100   [0] running env sleep 2s | echo "Sleeping in $0" &
101   [1] running env sleep 2s | echo "This is some job management" &
102   waiting 0.25s
103   [0] running env sleep 2s | echo "Sleeping in $0" &
104   [1] running env sleep 2s | echo "This is some job management" &
105   [2] running env sleep 2s | echo "MyShell is $SHELL" &
106   waiting 0.25s
107   [0] running env sleep 2s | echo "Sleeping in $0" &
108   [1] running env sleep 2s | echo "This is some job management" &
109   [2] running env sleep 2s | echo "MyShell is $SHELL" &
110   [3] running env sleep 2s &
111   Getting current runnning jobs ...
112   [0] running env sleep 2s | echo "Sleeping in $0" &
113   [1] running env sleep 2s | echo "This is some job management" &
114   [2] running env sleep 2s | echo "MyShell is $SHELL" &
115   [3] running env sleep 2s &
116   [4] running env sleep 2s &
117   Getting back to fore ground
118   Waiting for background jobs to terminate
119   At the same time I can still do other things like testing ...
120   True
121   True
122   False
123   True
124   Sleeping in /home/xuzh/Projects/ShellDesign/MyShell/dummy.mysh
125   [0] finished env sleep 2s | echo "Sleeping in $0" &
126   This is some job management
127   [1] finished env sleep 2s | echo "This is some job management" &
128   MyShell is /home/xuzh/Projects/ShellDesign/MyShell/MyShell.py
129   [2] finished env sleep 2s | echo "MyShell is $SHELL" &
130   [3] finished env sleep 2s &
131   [4] finished env sleep 2s &
132   Should produce empty content
133   Jobs are done /home/xuzh
134   Spawning dummy built_in job that is trying to read from user (will
135   suspend)
136   builtin_dummy: before any input requirements
137   [0] suspended env dummy &
138   builtin_dummy: before any input requirements
```

```
138 [1] suspended env dummy &
139 builtin_dummy: before any input requirements
140 [2] suspended env dummy &
141 builtin_dummy: before any input requirements
142 [3] suspended env dummy &
143 Counting jobs
144 [0] suspended env dummy &
145 [1] suspended env dummy &
146 [2] suspended env dummy &
147 [3] suspended env dummy &
148 WE'RE ONLY TERMINATING JOB [0] AND [1], YOU SHOULD SEE WARMING IF -w. NO
ZOMBIE
149 [0] terminated env dummy &
150 2020-07-30 12:29:18 ubuntu __main__[66819] WARNING Terminating job [0]
handler process by signal ...
151 [1] terminated env dummy &
152 2020-07-30 12:29:18 ubuntu __main__[66823] WARNING Terminating job [1]
handler process by signal ...
153 #####calling environ ...
154 SSH_CONNECTION=192.168.28.1 12000 192.168.28.146 22
155 LANG=en_US.UTF-8
156 OLDPWD=/home/xuzh/Projects/ShellDesign/MyShell
157 XDG_SESSION_ID=29
158 USER=xuzh
159 PWD=/home/xuzh/Projects/ShellDesign/MyShell
160 HOME=/home/xuzh
161 SSH_CLIENT=192.168.28.1 12000 22
162 MAIL=/var/mail/xuzh
163 SHELL=/home/xuzh/Projects/ShellDesign/MyShell/MyShell.py
164 SHLVL=2
165 LOGNAME=xuzh
166 DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
167 XDG_RUNTIME_DIR=/run/user/1000
168 PATH=/home/xuzh/.vscode-
server/bin/91899dcef7b8110878ea59626991a18c8a6a1b3e/bin:/usr/local/sbin:/u
sr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/ho
me/xuzh/.local/bin
169 _=/home/xuzh/Projects/ShellDesign/MyShell./MyShell.py
170 VS CODE_IPC_HOOK_CLI=/tmp/vscode-ipc-a1c81be6-faa1-49b2-8a12-
38a172ae37e9.sock
171 TERM_PROGRAM=vscode
172 TERM_PROGRAM_VERSION=1.47.3
173 COLORTERM=truecolor
174 VS CODE_GIT_IPC_HANDLE=/run/user/1000/vscode-git-2fbb053fa5.sock
175 GIT_ASKPASS=/home/xuzh/.vscode-
server/bin/91899dcef7b8110878ea59626991a18c8a6a1b3e/extensions/git/dist/as
kpass.sh
176 VS CODE_GIT_ASKPASS_NODE=/home/xuzh/.vscode-
server/bin/91899dcef7b8110878ea59626991a18c8a6a1b3e/node
177 VS CODE_GIT_ASKPASS_MAIN=/home/xuzh/.vscode-
server/bin/91899dcef7b8110878ea59626991a18c8a6a1b3e/extensions/git/dist/as
kpass-main.js
178 TERM=xterm-256color
179 ZSH=/home/xuzh/.oh-my-zsh
180 PAGER=less
181 LESS=-R
182 LSCOLORS=Gxfxcxdxbxegedabagacad
```

```
184 LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;3
3;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=
34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*
taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;3
1:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=0
1;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;3
1:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.jpg=01;35:
*.jpeg=01;35:*.mjpg=01;35:*.jpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;3
5:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:
*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:
*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:
*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*
nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.av
i=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01
;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;3
6:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36
:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*
opus=00;36:*.spx=00;36:*.xspf=00;36:
185 SSH_AUTH_SOCK=/tmp/ssh-sHY4pYyqIN0I/agent.66323
186 SSH_AGENT_PID=66324
187 PS1=$
188 BOLD=\033[1m
189 RED=\033[31m
190 BLUE=\033[34m
191 RESET=\033[0m
192 LINE=\033[1m\033[34m#####
#####\033[0m
193 Arg 0 is: /home/xuzh/Projects/ShellDesign/MyShell/dummy.mysh
194 Arg 1 is: foo
195 Arg 2 is: bar
196 Arg 3 is: foobar
197 Arg 4 is: hello
198 Arg 5 is: world
199 Arg 6 is: linux
200 Arg 7 is: linus
201 Arg 8 is: PyTorch
202 Arg 9 is: CS231n
203 Shifting number 1
204 Arg 0 is: /home/xuzh/Projects/ShellDesign/MyShell/dummy.mysh
205 Arg 1 is: bar
206 Arg 2 is: foobar
207 Arg 3 is: hello
208 Arg 4 is: world
209 Arg 5 is: linux
210 Arg 6 is: linus
211 Arg 7 is: PyTorch
212 Arg 8 is: CS231n
213 2020-07-30 12:29:19 ubuntu __main__ [66739] WARNING Unable to get the
variable "9", assigning empty string
214 Arg 9 is:
215 Shifting number 2
216 Arg 1 is: foobar
217 Shifting number 3
218 Arg 1 is: hello
219 Shifting number 4
```

```
220 Arg 1 is: world
221 Shifting number 5
222 Arg 1 is: linux
223 Bye!
224 [2] terminated env dummy &
225 [3] terminated env dummy &
226 2020-07-30 12:29:19 ubuntu __main__[66827] WARNING Terminating job [2]
    handler process by signal ...
227 2020-07-30 12:29:19 ubuntu __main__[66831] WARNING Terminating job [3]
    handler process by signal ...
```

3. 复杂 test 命令执行 (同时检查注释功能)

```
1 test ! -z "" -a ( -n "1" -o 1 -ge 1 ) -o 2 -ne 1 # False, -a -o from right
  to left
2 test ( ! -z "" -a ( -n "1" -o 1 -ge 1 ) ) -o 2 -ne 1 # True
```

```
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:30:46 $ test ! -z "" -a ( -n "1" -o 1 -ge 1 ) -o 2 -ne
1 # False, -a -o from right to left
False
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:30:53 $ test ( ! -z "" -a ( -n "1" -o 1 -ge 1 ) ) -o 2
-ne 1 # True
True
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:31:05 $
```

4. cd

```
1 cd ..
2 cd ..
3 cd
4 cd Doesn'tExist
5 cd /dev
6 cd /var/log
7 cd .
8 cd ~
9 cd /
10 cd $HOME
```

```
xuzh@ubuntu ~/Projects/ShellDesign 12:37:02 $ cd ..
xuzh@ubuntu ~/Projects 12:37:07 $ cd ..
xuzh@ubuntu ~ 12:37:07 $ cd
~
xuzh@ubuntu ~ 12:37:07 $ cd Doesn'tExist
2020-07-30 12:37:07 ubuntu __main__[66882] ERROR Cannot successfully execute command "cd". Exception
is:
FileNotFoundException: [Errno 2] No such file or directory: 'Doesn'tExist'
Extra info: {'type': 'cd'}
xuzh@ubuntu ~ 12:37:07 $ cd /dev
xuzh@ubuntu /dev 12:37:07 $ cd /var/log
xuzh@ubuntu /var/log 12:37:07 $ cd .
xuzh@ubuntu /var/log 12:37:07 $ cd ~
xuzh@ubuntu ~ 12:37:07 $ cd /
xuzh@ubuntu / 12:37:07 $ cd $HOME
xuzh@ubuntu ~ 12:37:09 $
```

5. clr

```
1 clr
```

```

File Edit Selection View Go Run Terminal Help
MyShell.py - bash (Workspace) [SSH: 192.168.28.146] - Visual Studio Code
SQL CONSOLE PROBLEMS TERMINAL ... 1: python
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:38:44 $ dr
rwxrwxr-x 2020-07-30 12:08:45 hello
rw-rw-r-- 2020-07-30 12:29:15 log.log
rw-rw-r-- 2020-07-29 12:57:15 should.out
rw-rw-r-- 2020-07-29 12:36:49 MyShell.md
rw-rw-r-- 2020-07-29 12:36:49 MyShell.out
rw-rw-r-- 2020-07-25 12:11:10 dummy.py
rwxrwxr-x 2020-07-25 14:09:29 process.py
rw-rw-r-- 2020-07-30 12:08:43 hello.c
rw-rw-r-- 2020-07-26 17:43:58 sleep10s.py
rwxrwxr-x 2020-07-29 14:52:17 MyShell.py
rw-rw-r-- 2020-07-29 12:56:43 dummy.mysl
rw-rw-r-- 2020-07-22 19:23:17 COLOR.py
rw-rw-r-- 2020-07-22 10:43:24 MyShellException.py
rwxrwxr-x 2020-07-29 12:42:10 __pycache__
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:39:11 $ 

```

```

File Edit Selection View Go Run Terminal Help
MyShell.py - bash (Workspace) [SSH: 192.168.28.146] - Visual Studio Code
SQL CONSOLE PROBLEMS TERMINAL ... 1: python
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:39:15 $ 

```

6. pwd

```

1  pwd
2  pwd -a

```

```

xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:39:15 $ pwd
~/Projects/ShellDesign/MyShell
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:39:36 $ pwd -a
/home/xuzh/Projects/ShellDesign/MyShell
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:39:40 $ 

```

7. dir

```

1  dir
2  cd ..
3  dir MyShell DirSync Doesn't Exist

```

```
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:39:40 $ dir
.
rwxrwxr-x 2020-07-30 12:08:45 hello
rw-rw-r-- 2020-07-30 12:29:15 log.log
rw-rw-r-- 2020-07-29 12:57:15 should.out
rw-rw-r-- 2020-07-29 12:36:49 MyShell.md
rw-rw-r-- 2020-07-30 12:26:17 result.out
rw-rw-r-- 2020-07-25 12:11:10 dummy.py
rwxrwxr-x 2020-07-25 14:09:29 process.py
rw-rw-r-- 2020-07-30 12:08:43 hello.c
rw-rw-r-- 2020-07-26 17:43:58 sleep10s.py
rwxrwxr-x 2020-07-29 14:52:17 MyShell.py
rw-rw-r-- 2020-07-29 12:56:43 dummy.mysh
rw-rw-r-- 2020-07-22 19:23:17 COLOR.py
rw-rw-r-- 2020-07-29 10:43:24 MyShellException.py
rwxrwxr-x 2020-07-29 12:42:10 __pycache__
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:40:37 $ cd ..
xuzh@ubuntu ~/Projects/ShellDesign 12:40:43 $ dir MyShell DirSync Doesn't Exist
MyShell
rwxrwxr-x 2020-07-30 12:08:45 hello
rw-rw-r-- 2020-07-30 12:29:15 log.log
rw-rw-r-- 2020-07-29 12:57:15 should.out
rw-rw-r-- 2020-07-29 12:36:49 MyShell.md
rw-rw-r-- 2020-07-30 12:26:17 result.out
rw-rw-r-- 2020-07-25 12:11:10 dummy.py
rwxrwxr-x 2020-07-25 14:09:29 process.py
rw-rw-r-- 2020-07-30 12:08:43 hello.c
rw-rw-r-- 2020-07-26 17:43:58 sleep10s.py
rwxrwxr-x 2020-07-29 14:52:17 MyShell.py
rw-rw-r-- 2020-07-29 12:56:43 dummy.mysh
rw-rw-r-- 2020-07-22 19:23:17 COLOR.py
rw-rw-r-- 2020-07-29 10:43:24 MyShellException.py
rwxrwxr-x 2020-07-29 12:42:10 __pycache__

DirSync
rwxrwxr-x 2020-07-22 19:23:17 DirSync.sh

Doesn't Exist
2020-07-30 12:40:59 ubuntu __main__ [66882] ERROR Cannot successfully execute command "dir". Exception is:
FileNotFoundException: Cannot list director[y|ies]:
[Errno 2] No such file or directory: 'Doesn't Exist'
Extra info: {'type': 'dir'}
xuzh@ubuntu ~/Projects/ShellDesign 12:40:59 $
```

8. echo

```
1 echo "\033[1m\033[31mHello, world.\033[0m"
2 echo "\033[1m\033[33mMy name is \$SHELL\033[0m"
3 echo -r "\033[1m\031[31mMy name is \$SHELL\033[0m"
4 echo without"\$SHELL"any"\$HOME"space and here come spaces
5 echo "中文测试" # 注释测试
```

```
xuzh@ubuntu ~/Projects/ShellDesign 12:44:38 $ echo "\033[1m\033[31mHello, world.\033[0m"
Hello, world.
xuzh@ubuntu ~/Projects/ShellDesign 12:45:21 $ echo "\033[1m\033[33mMy name is \$SHELL\033[0m"
My name is /home/xuzh/Projects/ShellDesign/MyShell/MyShell.py
xuzh@ubuntu ~/Projects/ShellDesign 12:45:21 $ echo -r "\033[1m\031[31mMy name is \$SHELL\033[0m"
\033[1m\031[31mMy name is /home/xuzh/Projects/ShellDesign/MyShell/MyShell.py\033[0m
xuzh@ubuntu ~/Projects/ShellDesign 12:45:21 $ echo without"\$SHELL"any"\$HOME"space and here come spaces
without/home/xuzh/Projects/ShellDesign/MyShell/MyShell.pyany/home/xuzhspace and here come spaces
xuzh@ubuntu ~/Projects/ShellDesign 12:45:22 $
```

```
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:55:59 $ echo "中文测试" # 注释测试
中文测试
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:08:03 $
```

9. exit

```
1 exit
```

```
xuzh@ubuntu ~/Projects/ShellDesign 12:45:22 $ exit
xuzh@ubuntu ~ ~/Projects/ShellDesign/MyShell
```

10. quit

```
1     quit
```

```
xuzh@ubuntu > ~/Projects/ShellDesign/MyShell > } master > ./MyShell.py -w
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:46:24 $ quit
xuzh@ubuntu > ~/Projects/ShellDesign/MyShell > } master >
```

11. jobs, fg, bg, term, exit 任务管理

```
1     set BOLD="\033[1m"
2     set RED="\033[31m"
3     set BLUE="\033[34m"
4     set RESET="\033[0m"
5
6     set
7     LINE="$BOLD$BLUE#####
8     #####$RESET"
9
10    echo $LINE
11
12    echo "Opening some sleepy jobs"
13    echo "And calling command jobs"
14    sleep 2s | echo "Sleeping in $0" &
15    echo "waiting 0.25s"
16    jobs
17    sleep 0.25s
18    sleep 2s | echo "This is some job management" &
19    echo "waiting 0.25s"
20    jobs
21    sleep 0.25s
22    sleep 2s | echo "MyShell is $SHELL" &
23    echo "waiting 0.25s"
24    jobs
25    sleep 0.25s
26    sleep 2s &
27    echo "waiting 0.25s"
28    jobs
29    sleep 0.25s
30    sleep 2s &
31    echo "Getting current runnning jobs ... "
32    jobs
33
34    echo "Getting back to fore ground"
35    echo "Waiting for background jobs to terminate"
36    echo "At the same time I can still do other things like testing ... "
37    test "" -o "a"
38    test ! -z "a" -a ( -n "1" -o 1 -ge 1 ) -a 2 -ne 1
39    test ! -z "" -a ( -n "1" -o 1 -ge 1 ) -o 2 -ne 1 # False, -a -o from right
40    to left
41    test ( ! -z "" -a ( -n "1" -o 1 -ge 1 ) ) -o 2 -ne 1 # True
42    sleep 2s
43
44    echo "Should produce empty content"
45    jobs
46
47    echo "Jobs are done ~"
```

```

46    echo "Spawning dummy built_in job that is trying to read from user (will
        suspend)"
47
48    dummy &
49    dummy &
50    dummy &
51    dummy &
52
53    echo "Counting jobs"
54    jobs
55
56    echo "$RED$BOLD""WE'RE ONLY TERMINATING JOB [0] AND [1], YOU SHOULD SEE
        WARMING IF -w. NO ZOMBIE""$RESET"
57
58    term 0 1
59
60    echo $LINE
61
62    exit # should see termination message

```

```

xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:55 $ set BOLD="\033[1m"
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:56 $ set RED="\033[31m"
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:56 $ set BLUE="\033[34m"
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:56 $ set RESET="\033[0m"
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:56 $ set LINE="$BOLD$BLUE#####
#####$RESET"
#####
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:56 $
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:56 $ echo $LINE
#####
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:56 $
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:56 $ echo "Opening some sleepy jobs"
Opening some sleepy jobs
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:56 $ echo "And calling command jobs"
And calling command jobs
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:56 $ sleep 2s | echo "Sleeping in $0" &
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:56 $ echo "waiting 0.25s"
waiting 0.25s
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:56 $ jobs
[0] running env sleep 2s | echo "Sleeping in $0" &
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:56 $ sleep 0.25s
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:56 $ sleep 2s | echo "This is some job management" &
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:56 $ echo "waiting 0.25s"
waiting 0.25s
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:56 $ jobs
[0] running env sleep 2s | echo "Sleeping in $0" &
[1] running env sleep 2s | echo "This is some job management" &
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:56 $ sleep 0.25s
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:56 $ sleep 2s | echo "MyShell is $SHELL" &
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:56 $ echo "waiting 0.25s"
waiting 0.25s
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:56 $ jobs
[0] running env sleep 2s | echo "Sleeping in $0" &
[1] running env sleep 2s | echo "This is some job management" &
[2] running env sleep 2s | echo "MyShell is $SHELL" &
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:56 $ sleep 0.25s
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:57 $ sleep 2s &
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:57 $ echo "waiting 0.25s"
waiting 0.25s
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:57 $ jobs
[0] running env sleep 2s | echo "Sleeping in $0" &
[1] running env sleep 2s | echo "This is some job management" &
[2] running env sleep 2s | echo "MyShell is $SHELL" &
[3] running env sleep 2s &
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:57 $ sleep 0.25s
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:57 $ sleep 2s &
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:57 $ echo "Getting current running jobs..."
Getting current running jobs...
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:57 $ jobs
[0] running env sleep 2s | echo "Sleeping in $0" &
[1] running env sleep 2s | echo "This is some job management" &
[2] running env sleep 2s | echo "MyShell is $SHELL" &
[3] running env sleep 2s &
[4] running env sleep 2s &
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:57 $
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:57 $ echo "Getting back to fore ground"
Getting back to fore ground

```

```
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:57 $ echo "Waiting for background jobs to terminate"
Waiting for background jobs to terminate
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:57 $ echo "At the same time I can still do other things like testing..."
At the same time I can still do other things like testing...
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:57 $ test "" -o "a"
True
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:57 $ test ! -z "a" -a ( -n "1" -o 1 -ge 1 ) -a 2 -ne 1
True
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:57 $ test ! -z "" -a ( -n "1" -o 1 -ge 1 ) -o 2 -ne 1 # False,
-a -o from right to left
False
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:57 $ test ( ! -z "" -a ( -n "1" -o 1 -ge 1 ) ) -o 2 -ne 1 # True
True
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:57 $ sleep 2s
Sleeping in /home/xuzh/Projects/ShellDesign/MyShell/MyShell.py
[0] finished env sleep 2s | echo "Sleeping in $0" &
This is some job management
[1] finished env sleep 2s | echo "This is some job management" &
MyShell is /home/xuzh/Projects/ShellDesign/MyShell/MyShell.py
[2] finished env sleep 2s | echo "MyShell is $SHELL" &
[3] finished env sleep 2s &
[4] finished env sleep 2s &
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:59 $
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:59 $ echo "Should produce empty content"
Should produce empty content
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:59 $ jobs
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:59 $
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:59 $
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:59 $ echo "Jobs are done ~"
Jobs are done /home/xuzh
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:59 $
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:59 $ echo "Spawning dummy built_in job that is trying to read from user (will suspend)"
Spawning dummy built_in job that is trying to read from user (will suspend)
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:59 $
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:59 $ dummy &
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:59 $ dummy &
builtin_dummy: before any input requirements
[0] suspended env dummy &
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:59 $ dummy &
builtin_dummy: before any input requirements
[1] suspended env dummy &
builtin_dummy: before any input requirements
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:59 $
[2] suspended env dummy &
builtin_dummy: before any input requirements
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:59 $
[3] suspended env dummy &
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:59 $ echo "Counting jobs"
Counting jobs
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:59 $ jobs
[0] suspended env dummy &
[1] suspended env dummy &
[2] suspended env dummy &
[3] suspended env dummy &
```

```
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:59 $
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:59 $ echo "$RED$BOLD""WE'RE ONLY TERMINATING JOB [0] ANDOMBIE""$RESET"LD SEE WARMING IF -w. NO ZO
WE'RE ONLY TERMINATING JOB [0] AND [1], YOU SHOULD SEE WARMING IF -w. NO ZOMBIE
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:59 $
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:59 $ term 0 1
[0] terminated env dummy &
2020-07-30 12:48:59 ubuntu __main__[67251] WARNING Terminating job [0] handler process by signal...
[1] terminated env dummy &
2020-07-30 12:48:59 ubuntu __main__[67255] WARNING Terminating job [1] handler process by signal...
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:59 $
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:59 $ echo $LINE
#####
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:59 $
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:48:59 $ exit # should see termination message
[2] terminated env dummy &
2020-07-30 12:49:01 ubuntu __main__[67259] WARNING Terminating job [2] handler process by signal...
[3] terminated env dummy &
2020-07-30 12:49:01 ubuntu __main__[67263] WARNING Terminating job [3] handler process by signal...
xuzh@ubuntu ~/Projects/ShellDesign/MyShell > master > []
```

```
1    dummy &
2    dummy &
3    dummy &
4    dummy &
5    jobs
```

```
6    term 0 1
7    bg 2 3
8    term 2 3
9    jobs
10   dummy &
11   dummy &
12   fg 0
13   1
14   2
15   3
16   4
17   5
18   term 1
19   jobs
```

```
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:52:29 $ dummy &
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:52:30 $ dummy &
builtin_dummy: before any input requirements
[0] suspended env dummy &
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:52:30 $ dummy &
builtin_dummy: before any input requirements
[1] suspended env dummy &
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:52:30 $ dummy &
builtin_dummy: before any input requirements
[2] suspended env dummy &
builtin_dummy: before any input requirements
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:52:30 $ jobs
[3] suspended env dummy &
[0] suspended env dummy &
[1] suspended env dummy &
[2] suspended env dummy &
[3] suspended env dummy &
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:52:30 $ term 0 1
[0] terminated env dummy &
2020-07-30 12:52:30 ubuntu __main__[67506] WARNING Terminating job [0] handler process by signal...
[1] terminated env dummy &
2020-07-30 12:52:30 ubuntu __main__[67510] WARNING Terminating job [1] handler process by signal...
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:52:30 $ bg 2 3
[2] continued env dummy &
[2] suspended env dummy &
[3] continued env dummy &
[3] suspended env dummy &
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:52:30 $ term 2 3
[2] terminated env dummy &
2020-07-30 12:52:30 ubuntu __main__[67514] WARNING Terminating job [2] handler process by signal...
[3] terminated env dummy &
2020-07-30 12:52:30 ubuntu __main__[67518] WARNING Terminating job [3] handler process by signal...
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:52:30 $ jobs
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:52:30 $ dummy &
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:52:30 $ dummy &
builtin_dummy: before any input requirements
[0] suspended env dummy &
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:52:30 $ fg 0
builtin_dummy: before any input requirements
[0] continued env dummy &
[1] suspended env dummy &
[0] running env dummy &
dummy1> 1
1
dummy2> 2
dummy3> 3
dummyend> 4
dummy_content> 5
[0] finished env dummy &
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:52:30 $ term 1
[1] terminated env dummy &
2020-07-30 12:52:34 ubuntu __main__[67526] WARNING Terminating job [1] handler process by signal...
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:52:34 $ jobs
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:52:35 $
```

12. [environ, set, unset](#) 变量检查

```

1 echo "Hello, my name is $SHELL"
2 set hello_msg="Hello, my name is"
3 echo -r "$hello_msg $USER, and I live in $HOME"
4 environ | grep hello_msg # should see hello_msg=Hello, my name is
5 unset hello_msg
6 echo "Should get empty output"
7 echo $hello_msg
8 environ | grep hello_msg # should see no hello_msg

```

```

xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:55:26 $ echo "Hello, my name is $SHELL"
Hello, my name is /home/xuzh/Projects/ShellDesign/MyShell/MyShell.py
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:55:58 $ set hello_msg="Hello, my name is"
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:55:58 $ echo -r "$hello_msg $USER, and I live in $HOME"
Hello, my name is xuzh, and I live in /home/xuzh
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:55:58 $ environ | grep hello_msg # should see hello_msg=Hello, my name is
hello_msg>Hello, my name is
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:55:58 $ unset hello_msg
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:55:58 $ echo "Should get empty output"
Should get empty output
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:55:58 $ echo $hello_msg
2020-07-30 12:55:58 ubuntu __main__[67490] WARNING Unable to get the varible "hello_msg", assigning empty string

xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:55:58 $ environ | grep hello_msg # should see no hello_msg
2020-07-30 12:55:59 ubuntu __main__[67490] WARNING The subprocess is not returning zero exit code
2020-07-30 12:55:59 ubuntu __main__[67490] ERROR Cannot successfully execute command "grep". Exception is:
CalledProcessException: None zero return code encountered
Extra info: None
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 12:55:59 $ 

```

13. umask

```

1 umask
2 echo "Changing UMASK to 0o077"
3 umask 077
4 touch text.txt
5 gcc hello.c -o hello
6 dir | grep -E "hello|text"
7 echo "Displaying UMASK"
8 umask
9 echo "Changing UMASK to 0o002"
10 umask 002

```

```

xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:16:31 $ umask
0o077
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:16:31 $ echo "Changing UMASK to 0o002"
Changing UMASK to 0o002
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:16:31 $ umask 002
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:16:34 $
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:16:46 $ umask
0o002
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:16:46 $ echo "Changing UMASK to 0o077"
Changing UMASK to 0o077
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:16:46 $ umask 077
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:16:46 $ touch text.txt
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:16:46 $ gcc hello.c -o hello
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:16:46 $ dir | grep -E "hello|text"
rw----- 2020-07-30 13:16:46 text.txt
rwx----- 2020-07-30 13:16:46 hello
rw-rw-r-- 2020-07-30 12:08:43 hello.c
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:16:46 $ echo "Displaying UMASK"
Displaying UMASK
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:16:46 $ umask
0o077
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:16:46 $ echo "Changing UMASK to 0o002"
Changing UMASK to 0o002
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:16:46 $ umask 002
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:16:46 $ 

```

14. printio, exec

```

1  printio
2  exec < dummy.mysh > result.out
3  exec
4  printio
5  cat
6  exec < result.out > ""
7  printio
8  wc
9  exec < "" > ""
10 printio
11 exec > result.out
12 echo "REPLACING"
13 exec > ""
14 cat result.out
15 exec >> result.out
16 echo "APPENDING"
17 exec > ""
18 cat result.out

```

```

xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:24:17 $ printio
FILE NUMBER OF INPUT FILE: 0, redirecting MyShell input to None
FILE NUMBER OF OUTPUT FILE: 1, redirecting MyShell output to None
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:24:18 $ exec < dummy.mysh > result.out
FILE NUMBER OF INPUT FILE: 4, redirecting MyShell input to <_io.TextIOWrapper name='dummy.mysh' mode='r' encoding='utf-8'>
FILE NUMBER OF OUTPUT FILE: 7, redirecting MyShell output to <_io.TextIOWrapper name='result.out' mode='w' encoding='utf-8'>
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:24:18 $ exec
FILE NUMBER OF INPUT FILE: 4, redirecting MyShell input to <_io.TextIOWrapper name='dummy.mysh' mode='r' encoding='utf-8'>
FILE NUMBER OF OUTPUT FILE: 7, redirecting MyShell output to <_io.TextIOWrapper name='result.out' mode='w' encoding='utf-8'>
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:24:18 $ printio
FILE NUMBER OF INPUT FILE: 4, redirecting MyShell input to <_io.TextIOWrapper name='dummy.mysh' mode='r' encoding='utf-8'>
FILE NUMBER OF OUTPUT FILE: 7, redirecting MyShell output to <_io.TextIOWrapper name='result.out' mode='w' encoding='utf-8'>
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:24:18 $ cat
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:24:18 $ exec < result.out > ""
FILE NUMBER OF INPUT FILE: 8, redirecting MyShell input to <_io.TextIOWrapper name='result.out' mode='r' encoding='utf-8'>
FILE NUMBER OF OUTPUT FILE: 1, redirecting MyShell output to None
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:24:18 $ printio
FILE NUMBER OF INPUT FILE: 8, redirecting MyShell input to <_io.TextIOWrapper name='result.out' mode='r' encoding='utf-8'>
FILE NUMBER OF OUTPUT FILE: 1, redirecting MyShell output to None
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:24:18 $ wc
    159   561 2940
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:24:18 $ exec < "" > ""
FILE NUMBER OF INPUT FILE: 0, redirecting MyShell input to None
FILE NUMBER OF OUTPUT FILE: 1, redirecting MyShell output to None
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:24:18 $ printio
FILE NUMBER OF INPUT FILE: 0, redirecting MyShell input to None
FILE NUMBER OF OUTPUT FILE: 1, redirecting MyShell output to None
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:24:18 $ exec > result.out
FILE NUMBER OF INPUT FILE: 0, redirecting MyShell input to None
FILE NUMBER OF OUTPUT FILE: 4, redirecting MyShell output to <_io.TextIOWrapper name='result.out' mode='w' encoding='utf-8'>
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:24:18 $ echo "REPLACING"
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:24:18 $ exec > ""
FILE NUMBER OF INPUT FILE: 0, redirecting MyShell input to None
FILE NUMBER OF OUTPUT FILE: 1, redirecting MyShell output to None
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:24:18 $ cat result.out
REPLACING
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:24:18 $ exec >> result.out
FILE NUMBER OF INPUT FILE: 0, redirecting MyShell input to None
FILE NUMBER OF OUTPUT FILE: 4, redirecting MyShell output to <_io.TextIOWrapper name='result.out' mode='a' encoding='utf-8'>
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:24:18 $ echo "APPENDING"
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:24:18 $ exec > ""
FILE NUMBER OF INPUT FILE: 0, redirecting MyShell input to None
FILE NUMBER OF OUTPUT FILE: 1, redirecting MyShell output to None
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:24:18 $ cat result.out
APPENDING
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 13:24:19 $ 

```

15. shift

通过命令行： ./MyShell.py -w dummy.mysh -a foo bar foobar hello world linux
linus PyTorch CS231n 运行MyShell

```

1 echo $0 $1 $2 $3 $4 $5 $6 $7 $8 $9
2 shift
3 echo $0 $1 $2 $3 $4 $5 $6 $7 $8 $9
4 shift 1
5 echo $0 $1 $2 $3 $4 $5 $6 $7 $8 $9
6 shift 2
7 echo $0 $1 $2 $3 $4 $5 $6 $7 $8 $9
8 shift 3
9 echo $0 $1 $2 $3 $4 $5 $6 $7 $8 $9
10 shift 4
11 echo $0 $1 $2 $3 $4 $5 $6 $7 $8 $9

```

```

xuzh@ubuntu ~/Projects/ShellDesign/MyShell(master) ➜ ./MyShell.py -w -a foo bar foobar hello world linux linus PyTorch CS23In
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 15:51:42 $ echo $0 $1 $2 $3 $4 $5 $6 $7 $8 $9
/home/xuzh/Projects/ShellDesign/MyShell.py foo bar foobar hello world linux linus PyTorch CS23In
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 15:53:30 $ shift
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 15:53:30 $ echo $0 $1 $2 $3 $4 $5 $6 $7 $8 $9
2020-07-30 15:53:30 ubuntu __main__ [69184] WARNING Unable to get the variable "9", assigning empty string
/home/xuzh/Projects/ShellDesign/MyShell.py bar foobar hello world linux linus PyTorch CS23In
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 15:53:30 $ shift 1
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 15:53:30 $ echo $0 $1 $2 $3 $4 $5 $6 $7 $8 $9
2020-07-30 15:53:30 ubuntu __main__ [69184] WARNING Unable to get the variable "8", assigning empty string
2020-07-30 15:53:30 ubuntu __main__ [69184] WARNING Unable to get the variable "9", assigning empty string
/home/xuzh/Projects/ShellDesign/MyShell.py foobar hello world linux linus PyTorch CS23In
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 15:53:30 $ shift 2
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 15:53:30 $ echo $0 $1 $2 $3 $4 $5 $6 $7 $8 $9
2020-07-30 15:53:30 ubuntu __main__ [69184] WARNING Unable to get the variable "6", assigning empty string
2020-07-30 15:53:30 ubuntu __main__ [69184] WARNING Unable to get the variable "7", assigning empty string
2020-07-30 15:53:30 ubuntu __main__ [69184] WARNING Unable to get the variable "8", assigning empty string
2020-07-30 15:53:30 ubuntu __main__ [69184] WARNING Unable to get the variable "9", assigning empty string
/home/xuzh/Projects/ShellDesign/MyShell.py world linux linus PyTorch CS23In
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 15:53:30 $ shift 3
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 15:53:30 $ echo $0 $1 $2 $3 $4 $5 $6 $7 $8 $9
2020-07-30 15:53:30 ubuntu __main__ [69184] WARNING Unable to get the variable "3", assigning empty string
2020-07-30 15:53:30 ubuntu __main__ [69184] WARNING Unable to get the variable "4", assigning empty string
2020-07-30 15:53:30 ubuntu __main__ [69184] WARNING Unable to get the variable "5", assigning empty string
2020-07-30 15:53:30 ubuntu __main__ [69184] WARNING Unable to get the variable "6", assigning empty string
2020-07-30 15:53:30 ubuntu __main__ [69184] WARNING Unable to get the variable "7", assigning empty string
2020-07-30 15:53:30 ubuntu __main__ [69184] WARNING Unable to get the variable "8", assigning empty string
2020-07-30 15:53:30 ubuntu __main__ [69184] WARNING Unable to get the variable "9", assigning empty string
/home/xuzh/Projects/ShellDesign/MyShell.py PyTorch CS23In
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 15:53:30 $ shift 4
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 15:53:47 $ echo $0 $1 $2 $3 $4 $5 $6 $7 $8 $9
2020-07-30 15:53:52 ubuntu __main__ [69184] WARNING Unable to get the variable "1", assigning empty string
2020-07-30 15:53:52 ubuntu __main__ [69184] WARNING Unable to get the variable "2", assigning empty string
2020-07-30 15:53:52 ubuntu __main__ [69184] WARNING Unable to get the variable "3", assigning empty string
2020-07-30 15:53:52 ubuntu __main__ [69184] WARNING Unable to get the variable "4", assigning empty string
2020-07-30 15:53:52 ubuntu __main__ [69184] WARNING Unable to get the variable "5", assigning empty string
2020-07-30 15:53:52 ubuntu __main__ [69184] WARNING Unable to get the variable "6", assigning empty string
2020-07-30 15:53:52 ubuntu __main__ [69184] WARNING Unable to get the variable "7", assigning empty string
2020-07-30 15:53:52 ubuntu __main__ [69184] WARNING Unable to get the variable "8", assigning empty string
2020-07-30 15:53:52 ubuntu __main__ [69184] WARNING Unable to get the variable "9", assigning empty string
/home/xuzh/Projects/ShellDesign/MyShell.py
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 15:53:52 $ 

```

16. sleep

```

1 sleep 10s

```

```

xuzh@ubuntu ~/Projects/ShellDesign/MyShell 15:56:02 $ sleep 10s
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 15:56:12 $ 

```

17. time

```

1 time

```

```

xuzh@ubuntu ~/Projects/ShellDesign/MyShell 15:56:35 $ time
2020-07-30 15:56:37.154888

```

18. verbose

```
1 verbose
2 verbose -d
3 verbose -e
4 echo $9
5 verbose -w
6 echo $9
```

```
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 15:56:37 $ verbose
Current logging level: WARN, WARNING
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 15:57:08 $ verbose -d
2020-07-30 15:57:11 ubuntu __main__[69184] DEBUG Queue is being cleaned, previous keys are []
2020-07-30 15:57:11 ubuntu __main__[69184] DEBUG Queue is cleaned, keys are []
2020-07-30 15:57:11 ubuntu __main__[69184] DEBUG Process is being cleaned, previous keys are []
2020-07-30 15:57:11 ubuntu __main__[69184] DEBUG Process is cleaned, keys are []
2020-07-30 15:57:11 ubuntu __main__[69184] DEBUG Status Dict is being cleaned, previous keys are []
2020-07-30 15:57:11 ubuntu __main__[69184] DEBUG Status Dict is cleaned, keys are []
2020-07-30 15:57:11 ubuntu __main__[69184] DEBUG Got the varible HOME as /home/xuzh
2020-07-30 15:57:11 ubuntu __main__[69184] DEBUG Got the varible HOME as /home/xuzh
2020-07-30 15:57:11 ubuntu __main__[69184] DEBUG Got the varible PS1 as $
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 15:57:11 $ verbose -e
2020-07-30 15:57:18 ubuntu __main__[69184] DEBUG Getting user input: verbose -e
2020-07-30 15:57:18 ubuntu __main__[69184] INFO Executing command verbose
2020-07-30 15:57:18 ubuntu __main__[69184] INFO Arguments are ['-e']
2020-07-30 15:57:18 ubuntu __main__[69184] DEBUG This is a builtin command.
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 15:57:18 $ echo $9

xuzh@ubuntu ~/Projects/ShellDesign/MyShell 15:57:22 $ verbose -w
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 15:57:26 $ echo $9
2020-07-30 15:57:30 ubuntu __main__[69184] WARNING Unable to get the varible "9", assigning empty string

xuzh@ubuntu ~/Projects/ShellDesign/MyShell 15:57:30 $ █
```

19. help

```
1 help
2 help test
3 help MyShell
4 help jobs
5 help waitpid
6 help Doesn'tExist
```

SQL CONSOLE

PROBLEMS

OUTPUT

TERMINAL

...

1: python

+

田

画

<

x

shell 或者命令行解释器是操作系统中最基本的用户接口。我们实现了一个跨平台的简单的shell 程序—**MyShell**，它具有以下属性：

1. 支持的内部指令集: `cd, clr, pwd, dir, echo, exit, quit, jobs, fg, bg, term, environ, set, unset, umask, printio, exec, shift, test, sleep, time, help, verbose`
请通过`help command`的格式查看指令的帮助

2. 开发者调试指令集: `dummy, check_zombie, queues`，用户一般不需要调用这些指令
请通过`help command`的格式查看指令的帮助

3. MyShell在开始执行后会将环境变量`SHELL`设置为`MyShell`的运行位置

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

`PARENT=<pathname>/MyShell.py` (也就是MyShell中`SHELL`变量的内容)。

5. MyShell能够从文件中提取命令行输入，例如`shell` 使用以下命令行被调用：

```
```shell
./MyShell.py dummy.mysh
```

```

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

6. MyShell除了上述的脚本执行，还支持其他运行时命令行参数：

用户可以调用`./MyShell.py -h`查看相关内容

```
```shell
usage: MyShell.py [-h] [-a [A [A ...]]) [-e] [-w] [-i] [-d] [F]
```

```

MyShell by xudenden@gmail.com

positional arguments:
F the batch file to be executed

optional arguments:

| | |
|-----------------|--|
| -h, --help | show this help message and exit |
| -a [A [A ...]]) | command line arguments to batch file |
| -e | enable error level debugging info log |
| -w | enable warning level debugging info log |
| -i | enable info level debugging info log |
| -d | enable debug(verbose) level debugging info log |

..

MyShell的调用举例：

```
```shell
./MyShell.py -w dummy.mysh -a foo bar foobar hello world linux linus PyTorch CS231n
```

```

7. MyShell支持I/O 重定向，`stdin` 和`stdout`，或者其中之一，例如命令行为：

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

```

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

`stdout` 重定向特除了在上面注明需要打印信息（后台任务管理，输入输出重定向查看等）的所有内部指令。

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

对于`exec`指令，使用重定向符号会导致MyShell的输入输出被调整到指定的文件。

MyShell在处理外部程序的调用时，为了方便用户观察结果和控制输入输出，会将`stderr`重定向到`stdout`一并打印到屏幕/定义的输出文件流。

8. MyShell支持后台程序执行。如果在命令行后添加`&`字符，在加载完程序后需要立刻返回命令行提示符。

后台程序的主要管理接口为`jobs, term, fg, bg`

```
- `test`  
测试表达式结果是否为真或假  
'test expression'  
- 支持的表达式:  
`-o`: 双目, 逻辑或, 参数为布尔值  
`-a`: 双目, 逻辑与, 参数为布尔值  
`!`: 单目, 逻辑反, 参数为布尔值  
`-z`: 单目, 字符串长度零检查, 参数为字符串  
`-n`: 单目, 字符串长度非零检查, 参数为字符串  
`==`: 双目, 字符串相等性检测, 参数为字符串  
`!=`: 双目, 字符串不等性检测, 参数为字符串  
`-eq`: 双目, 数值相等性检查, 参数为浮点数/整数  
`-ne`: 双目, 数值不等性检查, 参数为浮点数/整数  
`-gt`: 双目, 数值大于性检查`lhs > rhs`, 参数为浮点数/整数  
`-lt`: 双目, 数值小于性检查`lhs < rhs`, 参数为浮点数/整数  
`-ge`: 双目, 数值大于等于性检查`lhs >= rhs`, 参数为浮点数/整数  
`-le`: 双目, 数值小于等于性检查`lhs <= rhs`, 参数为浮点数/整数  
`(`: 左括号: 被括号包裹的内容会被当成一个表达式来解释, 返回布尔值  
`)`: 右括号: 被括号包裹的内容会被当成一个表达式来解释, 返回布尔值  
- 表达式和运算子必须用空白符分隔开  
- 支持复杂的嵌套表达式, 括号/单目运算符/双目运算符皆可嵌套执行  


```
```shell
test ! -z "" -a ( -n "1" -o 1 -ge 1 ) -o 2 -ne 1 # False, -a -o from right to left
test ( ! -z "" -a ( -n "1" -o 1 -ge 1 ) ) -o 2 -ne 1 # True
```

```

  
- `|-a, -o`从右向左结合, 但用户可以通过括号来定制它们的运算顺序  
- 用户需要保证输入的内容是合理的可匹配的表达式  
- 括号需匹配完整  
- 运算符能处理的数据类型需要进行合理判断。所有经过运算的表达式结果: 布尔值
```

(END)

```
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 16:34:06 $ help test
- `test`
    测试表达式结果是否为真或假

    `test expression`

- 支持的表达式:
    `~-o`: 双目, 逻辑或, 参数为布尔值
    `~-a`: 双目, 逻辑与, 参数为布尔值
    `~!~`: 单目, 逻辑反, 参数为布尔值
    `~-z`: 单目, 字符串长度空检查, 参数为字符串
    `~-n`: 单目, 字符串长度非空检查, 参数为字符串
    `~=~=`: 双目, 字符串相等性检查, 参数为字符串
    `!=~=`: 双目, 字符串不等性检查, 参数为字符串
    `~-eq`: 双目, 数值相等性检查, 参数为浮点数/整数
    `~-ne`: 双目, 数值不等性检查, 参数为浮点数/整数
    `~-gt`: 双目, 数值大于性检查`lhs > rhs`, 参数为浮点数/整数
    `~-lt`: 双目, 数值小于性检查`lhs < rhs`, 参数为浮点数/整数
    `~-ge`: 双目, 数值大于等于性检查`lhs >= rhs`, 参数为浮点数/整数
    `~-le`: 双目, 数值小于等于性检查`lhs <= rhs`, 参数为浮点数/整数
    `(~`: 左括号: 被括号包裹的内容会被当成一个表达式来解释, 返回布尔值
    `)`~: 右括号: 被括号包裹的内容会被当成一个表达式来解释, 返回布尔值
- 表达式和运算子必须用空白符分隔开
- 支持复杂的嵌套表达式, 括号/单目运算符/双目运算符皆可嵌套执行
```shell
test ! -z "" -a (-n "1" -o 1 -ge 1) -o 2 -ne 1 # False, -a -o from right to left
test (! -z "" -a (-n "1" -o 1 -ge 1)) -o 2 -ne 1 # True
```
- `~-a, -o`从右向左结合, 但用户可以通过括号来定制它们的运算顺序
- 用户需要保证输入的内容是合理的可匹配的表达式
    - 括号需匹配完整
    - 运算符能处理的数据类型需要进行合理判断。所有经过运算的表达式结果: 布尔值
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 16:34:23 $ help Doesn'tExist
2020-07-30 16:34:32 ubuntu __main__ [71296] WARNING Cannot find help page in builtin dict, trying with man command
No manual entry for Doesn'tExist
2020-07-30 16:34:32 ubuntu __main__ [71296] WARNING The subprocess is not returning zero exit code
2020-07-30 16:34:32 ubuntu __main__ [71296] ERROR Cannot successfully execute command "help". Exception is:
HelpException: Cannot find help page for Doesn'tExist
Extra info: None
xuzh@ubuntu ~/Projects/ShellDesign/MyShell 16:34:32 $
```

程序完整源码

```

1 #!/usr/bin/python
2 import os # 大部分os相关功能来自此包
3 import io # 用于构建一个stdin的wrapper以捕捉程序的输入/输出请求
4 import sys # 一些系统调用/常数的来源包
5 import re # 正则表达式
6 import time # 用在builtin_sleep中·针对Windows系统的睡眠操作
7 import platform # 获取平台信息
8 import getpass # 获取用户信息
9 import logging # 用于打印一些详细调试信息
10 import coloredlogs # 用于打印带颜色的调试信息
11 import datetime # 用于取得时间/日期·用于builtin_time
12 import readline # 用于提供人性化的input函数操作·例如输入历史功能等
13 import subprocess # 用于刷出/控制子进程
14 import multiprocessing # 用于后台程序的运行/管理
15 import traceback # 用于打印报错信息的调用堆栈
16 import argparse # 用于解释本程序的命令行参数
17 import codecs # 用于转义字符串·用于builtin_echo
18 import signal # 用于在调用builtin_term后给multiprocessing传递信号
19 import copy # 用于复制MyShell以进行multiprocessing和任务管理
20
```

```
21 from subprocess import Popen, PIPE, STDOUT # 子进程管理
22 from multiprocessing import Process, Queue, Pipe, Pool, Manager, Value # 多进程
23 from COLOR import COLOR # 颜色信息
24 from MyShellException import * # MyShell错误管理
25 log = logging.getLogger(__name__)
26
27 coloredlogs.install(level='DEBUG') # Change this to DEBUG to see more info.
28
29
30 # a decorator for logging function call
31 # 装饰器：用于在函数调用之前打印相关信息，有助于调试或检查函数调用记录
32 def logger(func):
33     def wrapper(*args, **kwargs):
34         # 本装饰器中我们使用logging模块提供详细信息打印功能
35         log.debug(f"CALLING FUNCTION: {COLOR.BOLD(func)}")
36         return func(*args, **kwargs)
37     return wrapper
38
39
40 # a class decorator that can modify all callable class method with a decorator
41 # here we use this to log function call of sys.stdin wrapper
42 # 一个带参数装饰器，用于对类进行修改：为类中的每一个可调用函数添加参数中所示的装饰器
43 def for_all_methods(decorator):
44     def decorate(cls):
45         for attr in cls.__dict__: # there's probably a better way to do this
46             if callable(getattr(cls, attr)):
47                 # 通过批量修改类的内容，避免重复代码，提高拓展性
48                 setattr(cls, attr, decorator(getattr(cls, attr)))
49         return cls
50     return decorate
51
52
53 # a dict that logs itself on every getting item operation
54 # 一个会在调用__getitem__前打印相关信息的函数
55 class LoggingDict(dict):
56     def __getitem__(self, key):
57         log.debug(f"GETTING DICT ITEM {COLOR.BOLD(key)}")
58         return super().__getitem__(key)
59
60
61 class MyShell:
62     def __init__(self, dict_in={}, cmd_args=[]):
63         # 所有的内置功能的函数开头都是builtin_
64         builtin_prefix = "builtin_"
65
66         # builtins is a simple dict containing functions (not called on access)
67         self.builtins = {}
68         for key, value in MyShell.__dict__.items():
69             if key.startswith(builtin_prefix):
70                 self.builtins[key[len(builtin_prefix)::]] = value
71
72         # 环境变量，用到了下面将要提到的OnCallDict的功能
73         # we're using callable dict to evaluate values on call
74         unsupported = {str(i): MyShell.PicklableCMDArgs(self, i) for i in
75 range(10)}
76         self.vars = MyShell.OnCallDict(unsupported=unsupported)
77         self.vars["PS1"] = "$"
```

```
77         self.vars["SHELL"] = os.path.abspath(__file__)
78
79     # 对于命令函数参数，我们为了防止写太多重复代码，就使用了picklable_nested类来自动生
    成相关callable object
80     # ! pickle cannot handle nested function, however a simulating callable
    object is fine
81     # 由于Pickle无法处理nested function，我们使用类来实现相关功能
82     if not cmd_args:
83         cmd_args = [os.path.abspath(__file__)]
84     self.cmd_args = cmd_args
85
86     # update： 我们会在调用多线程/后台执行功能时清空这一部分的变量，能够解决的问题有：
87     # 1. queue can only be pass by inheritance
88     # 2. cannot pickle weakref object
89     # 3. cannot pickle thread lock object
90     # 4. for safety purpose, cannot pickle authentication string
91     # pickle无法正常处理含有multiprocessing.Manager的object，因此我们不会显示储存
    这样的变量
92     # ! damn strange ... when an object has a Manager, multiprocessing refuse
    to spawn it on Windows
93     # self.job_manager = Manager()
94     self.jobs = Manager().dict()
95     self.status_dict = Manager().dict()
96     self.queues = {}
97     self.process = {}
98     self.subp = None
99
100    # 用于builtin_exec对输入输出的调整
101    # None表示从原始输入/输出源接受相关信息
102    self.input_file = None
103    self.output_file = None
104
105    # 初始化MyShell时可以传入初始环境变量
106    for key, value in dict_in.items():
107        # user should not be tampering with the vars already defined
108        # however we decided not to disturb them
109        self.vars[key] = value
110
111    log.debug(f"SHELL var content: {self.vars['SHELL']}")
112    log.debug(f"Built in command list: {self.builtins}")
113    log.debug("MyShell is instanciated.")
114
115    @property
116    def job_counter(self):
117        keys = [int(key) for key in self.jobs.keys()]
118        count = 0
119        while count in keys:
120            count += 1
121        return str(count)
122
123    # 我们考虑过直接使用系统的环境变量接口，但那样或许就少了很多提前控制功能，跨平台性也不一
    定会很好
124    # 所以MyShell会单独管理自己的变量（这里不一定要称为环境变量）
125    # 与MyShell配合使用，用于统一管理环境变量，并与真正的系统环境变量交互的字典
126    # 主要功能为，若字典value为可执行内容，则返回执行后的结果
127
128    class OnCallDict():
129        def keys(self):
```

```
130         return os.environ.keys()
131
132     def __iter__(self):
133         return os.environ.__iter__()
134
135     def __init__(self, unsupported={}):
136         self.unsupported = unsupported
137
138     def __getitem__(self, key):
139         try:
140             if key in self.unsupported:
141                 var = self.unsupported[key]()
142             else:
143                 var = os.environ.__getitem__(key)
144                 log.debug(f"Got the variable {COLOR.BOLD(key + ' as ' + var)}")
145         except (KeyError, IndexError) as e:
146             log.warning(f"Unable to get the variable \'{key}\', assigning
empty string")
147             var = ""
148
149         return var
150
151     def __setitem__(self, key, value):
152         if key in self.unsupported:
153             raise ReservedKeyException(f"Key {key} is reserved, along with
\"{list(self.unsupported.keys())}\"")
154
155         return os.environ.__setitem__(key, value)
156
157     def __delitem__(self, key):
158         if key in self.unsupported:
159             raise ReservedKeyException(f"Key {key} is reserved, along with
\"{list(self.unsupported.keys())}\"")
160
161         # StdinWrapper是一个继承自io.TextWrapper·是本类型对获取输入的job进行线程暂停的一种
162         # 方式
163
164         # 我们考虑过直接使用系统调用·但跨平台性会很难保证·因此对于后台外部命令·我们会直接关闭
165         # 输入PIPE
166
167         @for_all_methods(logger) # using the full class logging system
168         class StdinWrapper(io.TextIOWrapper):
169
170             def __init__(self, queue, count, job, status_dict, *args, **kwargs):
171                 super().__init__(*args, **kwargs)
172                 self.queue = queue
173                 self.job = job
174                 self.status_dict = status_dict
175                 self.count = count
176                 self.ok = False
177
178             # 用于修改job状态并打印相关信息
179             def job_status(self, status):
180                 self.status_dict[self.count] = status
181                 print(f"\n{COLOR.BOLD(COLOR.YELLOW(f'[{self.count}]'))}
{COLOR.BOLD(status)} env {COLOR.BOLD(self.job)}", file=sys.__stdout__)
182
183             # 将第一次输入阻塞·等待主线程的fg命令
184             # there might exists a better way
185             def isok(self):
186                 if not self.ok:
187                     log.debug("Aha! You want to read something? Wait on!")
188
189
```

```
182             self.job_status("suspended")
183
184         log.debug(f"WHAT IS SELF.STATUS_DICT: {self.status_dict}")
185         log.debug(f"WHAT IS SELF.COUNT: {self.count}")
186
187         # queue的功能就是将本线程阻塞
188         content = self.queue.get()
189
190         log.debug(f"WHAT DID YOU GET: {content}")
191
192         self.isok = True
193
194         self.job_status("running")
195
196         # 为了方便调试，我们对很多内容的调用都内置了记录器
197     def __getattribute__(self, name):
198         log.debug(f"GETTING ATTRIBUTE: {COLOR.BOLD(name)}")
199         return super().__getattribute__(name)
200
201     def fileno(self):
202         self.isok()
203         return super().fileno()
204
205         # 以命令提示符方式执行MyShell
206     def __call__(self):
207         log.debug("This is MyShell.")
208
209         # 通过返回值方式传输退出信号
210         exit_signal = False
211         while not exit_signal:
212             exit_signal = self.command_prompt()
213
214         # 用于模仿Nested Function的Object
215         # 可以被pickle
216     class PicklableCMDArgs():
217         # 在我们的使用环境下，shell变量被直接传入了self，也就保留了指针，因此外部对
218         cmd_args变量的改变也会使得这里的结果不同
219         def __init__(self, shell, number):
220             self.shell = shell
221             self.number = number
222
223         def __call__(self):
224             return self.shell.cmd_args[self.number]
225
226     def builtin_cd(self, pipe="", args=[]):
227         # 更换目录功能
228         # 对于多个参数的情况进行警告，并尝试进入第一个参数所示的功能
229         if len(args):
230             if len(args) > 1:
231                 log.warning("Are you trying to cd into multiple dirs? We'll only
232 accept the first argument.")
233             target_dir = args[0]
234         else:
235             # 在一般的shell中，无参数的cd调用会进入用户主目录
236             # 但我们根据作业要求实现了打印当前工作目录的功能
237             # well, we'd like to stay in home
238             # but the teacher says we should pwd instead
239             # target_dir = self.home()
```

```
238         return self.builtin_pwd(pipe=pipe, args=args)
239
240     # 若用户的输入开头为~符号 · 替换为用户主目录内容
241     # if target_dir.startswith("~"):
242     #     target_dir = f"{self.home()}{target_dir[1::]}"
243     # the dir might not exist
244     # 路径不存在则报错
245     try:
246         log.info(f"Changing CWD to {COLOR.BOLD(target_dir)}")
247         os.chdir(target_dir)
248         self.vars["PWD"] = os.getcwd()
249     except (FileNotFoundException, NotADirectoryError, PermissionError) as e:
250         raise FileNotFoundException(e, {"type": "cd"})
251
252     # 清空屏幕功能
253     def builtin_clr(self, pipe="", args=[]):
254         # clear the screen
255         # we're forced to do a system call here ...
256         # Windows系统下为cls命令 · 而*nix下为clear
257         if os.name == "nt":
258             os.system("cls")
259         else:
260             os.system("clear")
261         # return ""
262
263     def builtin_pwd(self, pipe="", args=[]):
264         # 打印当前目录 · 如果使用了-a参数会打印完整的目录 ( 将~符号替换为self.home()的值 )
265         cwd = self.cwd() # 通过系统调用获得当前路径
266
267         # 根据参数判断
268         if len(args):
269             if args[0] == "-a" and cwd.startswith("~"):
270                 cwd = f"{self.home()}{cwd[1::]}"
271             else:
272                 raise ArgException("pwd only accepts -a as argument, otherwise
don't give any argument")
273             return cwd
274
275     def builtin_dir(self, pipe="", args=[]):
276         # 内置get_mode函数 · 用于获取文件权限
277         def get_mode(permissions):
278             map_string = "rwxrwxrwx"
279             map_empty = "-----"
280             # 我们使用列表生成器对bit列表进行判断
281             result = "".join([map_string[i] if permissions[i] else map_empty[i]
for i in range(9)])
282             return result
283
284         # 没有参数时 · 处理当前目录
285         if not len(args):
286             args.append(".")
287
288         # 不存在的目录的报错信息集合 ( 用于最后的raise检查 · 会被调用'\n'.join )
289         not_in_list = []
290         # 储存将要打印的内容 · 最后会作为'\n'.join(result)的参数
291         result = []
292
293         for target_dir in args:
```

```
294     # 第一行是当前打印的目录的名称
295     result.append(target_dir)
296     # if target_dir.startswith("~"):
297     #     target_dir = f"{self.home()}{'target_dir[1::]'}"
298     # the dir might not exist
299     try:
300         # 可能会出现找不到文件的情况，将找不到的文件添加到not_in_list中，继续处理
301         # 下一个目录的打印操作，并在最后raise
302         log.debug(f"Listing dir {COLOR.BOLD(target_dir)}")
303         # 通过os接口获得文件名称
304         dir_list = os.listdir(target_dir)
305         for file_name in dir_list:
306             # 下面的stat等函数调用需要全路径
307             full_name = f"{target_dir}/{file_name}"
308
309             # 文件的访问权限，最后修改时间等信息
310             file_stat = os.stat(full_name)
311             file_mode = file_stat.st_mode
312             file_time = file_stat.st_mtime
313
314             # # guess we're not using this ...
315             # type_code = file_mode >> 12
316
317             # 提取整数的每一位，一共获取9位
318             permissions = [(file_mode >> bit) & 1 for bit in range(9 -
319             1, -1, -1)]
320
321             # 将访问权限和目录信息转换为字符串
322             mode_str = get_mode(permissions)
323             time_str =
324                 datetime.datetime.fromtimestamp(file_time).strftime("%Y-%m-%d %H:%M:%S")
325
326             # 合并成为最终结果的一行的前半部分
327             file_line = f"{mode_str} {time_str} "
328
329             # 根据文件的类型/是否位可执行文件，写入相关颜色
330             if os.path.isdir(full_name):
331                 # path is a directory
332                 file_line += COLOR.BLUE(COLOR.BOLD(file_name))
333             else:
334                 # ! os.access not working properly on windows
335                 # todo: maybe we can recognize char block or others
336                 # this is brutal
337                 if os.access(full_name, os.X_OK):
338                     # 用红色显示可执行文件
339                     file_line += COLOR.RED(COLOR.BOLD(file_name))
340                 else:
341                     # 用普通粗体显示一般文件
342                     file_line += COLOR.BOLD(file_name)
343
344             # 添加到完整的result数组中
345             result.append(file_line)
346     except (FileNotFoundException, NotADirectoryError, PermissionError) as e:
347         not_in_list.append(str(e))
348     finally:
349         # 若用户传入了多个参数，在不同的文件夹间打印一个换行（最终'\n'.join的效果）
350
```

```
347             result.append("")
348
349         # 找不到的目录 ( 可能输入了一个文件 ) 数量为非空
350         if len(not_in_list):
351             # 打印可以打印的内容
352             # 由于raise会使函数停止继续执行，我们在此打印一些信息
353             # ! 注意，这种情况下我们不会返回结果，piping会停止
354             print("\n".join(result), end="")
355             joined = '\n'.join(not_in_list)
356             raise FileNotFoundError(f"Cannot list director[y|ies]:"
357             f"\n{joined}", {"type": "dir"})
358
359         # 若一切正常则以字符串形式直接返回最终结果
360         return "\n".join(result)
361
362     def builtin_echo(self, pipe="", args=[]):
363         # 我们支持-r参数，只能放在开头，如果用户使用了-r: raw input就不会对输入的特殊内容
364         # 进行转义翻译
365         # 但是相应的shell内置参数还是会被尝试替换，例如变量或者表示用户根目录的~符号，这不是
366         # echo函数所能控制的
367         # 没有-r我们会尝试转义其他的escape code
368         # ! if -r is provided, we won't do any escaping
369         result = f"{' '.join(args)}\n"
370         if len(args) >= 1 and args[0] == "-r":
371             result = result[len("-r ")::]
372         else:
373             # 通过codecs包实现较易于拓展的转义功能
374             result = codecs.escape_decode(bytes(result, "utf-8"))
375             [0].decode("utf-8")
376
377         return result
378
379     def builtin_exit(self, pipe="", args=[]):
380         # 我们通过异常退出
381         raise ExitException("Exitting ... ")
382
383     def builtin_quit(self, pipe="", args=[]):
384         # 我们通过异常退出
385         raise ExitException("Quitting ... ")
386
387     # 后台工作的相关内容
388     @staticmethod
389     def job_status_fmt(job_number, status, content):
390         # 以统一的形式获得后台工作的样式
391         return f'{COLOR.BOLD(COLOR.YELLOW(f"[{job_number}]'))}'
392         '{COLOR.BOLD(status)} env {COLOR.BOLD(content)}'
393
394     def job_status(self, job_number, status=None):
395         # 设定后台工作的状态并返回统一的打印样式
396         if status is not None:
397             self.status_dict[job_number] = status
398         else:
399             status = self.status_dict[job_number]
400
401         return self.job_status_fmt(job_number, status, self.jobs[job_number])
402
403     def builtin_jobs(self, pipe="", args=[]):
404         # 打印当前所有后台工作的执行情况
405         log.debug("Trying to get all jobs")
406         log.info(f"Content of jobs {COLOR.BOLD(self.jobs)}")
```

```
400     log.info(f"Content of status_dict {COLOR.BOLD(self.status_dict)}")
401     result = [self.job_status(key) for key in self.jobs.keys()]
402     return '\n'.join(result)
403
404     def builtin_queues(self, pipe="", args=[]):
405         # 开发者测试用命令 · 检查内存泄漏用
406         log.debug("Trying to get all queues")
407         log.info(f"Content of queues {COLOR.BOLD(self.queues)}")
408         print(f"Existing multiprocessing.Queue s are:
409             {COLOR.BOLD(self.queues)}", file=sys._stdout_)
410
411     def builtin_fg(self, pipe="", args=[]):
412         # 将因为尝试获取用户输入而挂起的工作提到前台执行
413
414         # 数量检查
415         if len(args) != 1:
416             raise JobException("Argument number is not correct, only one
417 expected.", {"type": "len"})
418         elif args[0] not in self.jobs:
419             raise JobException(f"Cannot find job is jobs number \'{args[0]}\'",
420 {"type": "key"})
421
422         # 通过multiprocessing.Queue进行沟通
423         # ! 注意 · 外部命令的读取请求不会被处理 · 因为在后台执行的subshell中PIPE会被关闭
424         log.debug(f"Foreground is called with {COLOR.BOLD(args)}")
425         print(self.job_status_fmt(args[0], "continued", self.jobs[args[0]]),
426 file=sys._stdout_)
427
428         if self.status_dict[args[0]] == "suspended":
429             # ! only put into queue if already suspended
430             # else the main process will get what it just put into the queue
431             self.queues[args[0]].put("dummy")
432
433
434         log.info("Waiting for foreground task to finish ... ")
435
436         # 挂起主线程 · 等待后台程序执行完毕
437         self.queues[args[0]].get()
438         log.debug("Continuing main process")
439
440         # 继续执行后台job
441         def builtin_bg(self, pipe="", args=[]):
442             # 处理相关参数调用
443             if not len(args):
444                 raise JobException("Argument number is not correct, one or more
445 expected.", {"type": "len"})
446
447             not_in_list = [arg for arg in args if arg not in self.jobs]
448             args = [arg for arg in args if arg in self.jobs]
449
450             log.debug(f"Background is called with {COLOR.BOLD(args)}")
451             for job_number in args:
452                 if self.status_dict[job_number] == "suspended":
453                     print(self.job_status_fmt(job_number, "continued",
454 self.jobs[job_number]), file=sys._stdout_)
455                     print(self.job_status_fmt(job_number, "suspended",
456 self.jobs[job_number]), file=sys._stdout_)
457                 elif self.status_dict[job_number] == "running":
458                     log.warning(f"Job [{job_number}] is already running")
459
```

```
451     # 找不到的后台工作会被反馈给用户
452     if len(not_in_list):
453         raise JobException(f"Cannot find job with number \""
454             f"{not_in_list}\", {"type": "key"})
```

```
455     def builtin_term(self, pipe="", args=[]):
456         # 处理相关参数调用
457         if not len(args):
458             raise JobException("Argument number is not correct, one or more
459             expected.", {"type": "len"})
```

```
460             not_in_list = [arg for arg in args if arg not in self.jobs]
461             args = [arg for arg in args if arg in self.jobs]
```

```
462
463             for job_number in args:
464                 log.debug("Terminating ... ")
465                 # 对multiprocessing的进程包发出signal.SIGTERM信号，给其机会处理相关内容
466                 # (关闭subshell等)
467                 # ! 在*nix上与subshell·run_command_wrap配合可避免zombie process
468                 # ! Windows中由于接口不匹配的原因，无法完全清除zombie
469                 os.kill(self.process[job_number].pid, signal.SIGTERM)
```

```
470
471             print(self.job_status_fmt(job_number, "terminated",
472                 self.jobs[job_number]), file=sys.__stdout__)
```

```
473
474             # 我们以jobs数组为蓝本，判断jobs是否已完成或者被强行结束等
475             # 配合clean_up函数，status_dict和queues等其他数组会被正常清理
476             del self.jobs[job_number]
```

```
477
478             # 找不到的后台工作会被反馈给用户
479             if len(not_in_list):
480                 raise JobException(f"Cannot find jobs with number \""
481                     f"{not_in_list}\", {"type": "key"})
```

```
482
483             def builtin_check_zombie(self, pipe="", args=[]):
484                 # 开发者调试用函数
485                 # 用于检查当前进程下的zombie process
486                 any_process = -1
487                 while True:
488                     # This will raise an exception on Windows. That's ok.
489                     pid, status = os.waitpid(any_process, os.WNOHANG)
490                     if pid == 0:
491                         break
492                     if os.WIFEXITED(status):
493                         print(f"The process of pid \'{COLOR.BOLD(pid)}\' is exited.",
494                             file=sys.__stdout__)
495                     elif os.WIFSTOPPED(status):
496                         print(f"The process of pid \'{COLOR.BOLD(pid)}\' is stopped.",
497                             file=sys.__stdout__)
498                     elif os.WIFCONTINUED(status):
499                         print(f"The process of pid \'{COLOR.BOLD(pid)}\' is continued.",
500                             file=sys.__stdout__)
501                     else:
502                         print(f"The process of pid \'{COLOR.BOLD(pid)}\' is of status
503 {COLOR.BOLD(status)}", file=sys.__stdout__)
```

```
504
505             def cleanup_jobs(self):
506                 # 对jobs命令的拓展
```

```
500     # 我们的任务管理系统以self.jobs字典为准，每次命令调用后都会刷新当前的数据内容以删除  
不必要的元素  
501     # 管理：self.queues, self.status_dict, self.process  
502     # 避免了内存泄漏  
503     keys = list(self.queues.keys())  
504  
505     log.debug(f"Queue is being cleaned, previous keys are  
{COLOR.BOLD(keys)}")  
506     for i in keys:  
507         if i not in self.jobs.keys():  
508             del self.queues[i]  
509     log.debug(f"Queue is cleaned, keys are  
{COLOR.BOLD(list(self.queues.keys()))}")  
510  
511     keys = list(self.process.keys())  
512  
513     log.debug(f"Process is being cleaned, previous keys are  
{COLOR.BOLD(keys)}")  
514     for i in keys:  
515         if i not in self.jobs.keys():  
516             del self.process[i]  
517     log.debug(f"Process is cleaned, keys are  
{COLOR.BOLD(list(self.process.keys()))}")  
518  
519     keys = list(self.status_dict.keys())  
520  
521     log.debug(f"Status Dict is being cleaned, previous keys are  
{COLOR.BOLD(keys)}")  
522     for i in keys:  
523         if i not in self.jobs.keys():  
524             del self.status_dict[i]  
525     log.debug(f"Status Dict is cleaned, keys are  
{COLOR.BOLD(list(self.status_dict.keys()))}")  
526  
527     def builtin_environ(self, pipe="", args=[]):  
528         # 将MyShell的环境变量返回给用户  
529         result = [f"{key}={COLOR.BOLD(self.vars[key])}" for key in self.vars] +  
[""] # for dummy line break  
530         return "\n".join(result)  
531  
532     def builtin_set(self, pipe="", args=[]):  
533         # 修改脚本变量，用户需要保证自己的输入内容被解释为一个完整的参数  
534         # 用等于号分割变量名和变量的具体值  
535         # 变量都以字符串的形式储存  
536         # we'd like the user to use the equal sign  
537         # and we'd like to treat variables only as string  
538         for arg in args:  
539             split = arg.split("=")  
540             if len(split) != 2:  
541                 raise SetPairUnmatchedException(f"Cannot match argument {arg}.",  
{"type": "set"})  
542             key, value = split  
543             log.debug(f"Setting \'{key}\' in environment variables to \'  
{value}\'")  
544  
545         try:  
546             # 修改pwd时，可能会有找不到目录的错误  
547             # might be error since self.vars is not a simple dict
```

```
548             self.vars[key] = value
549         except (FileNotFoundException, NotADirectoryError, PermissionError) as e:
550             raise FileNotFoundException(e, {"type": "set", "arg_pair": [key, value]})  

551
552     def builtin_unset(self, pipe="", args=[]):
553         # 取消MyShell中一些已经设置好的变量
554         cannot_unset = []
555         # 类似builtin_environ中的处理方式 · 为了跳过可能出错的变量 · 我们手动循环
556         for key in args:
557             try:
558                 del self.vars[key]
559             except (KeyError, ReservedKeyException) as e:
560                 cannot_unset.append(key)
561         if len(cannot_unset):
562             raise UnsetKeyException(f"Cannot find/unset these keys:  

563             {cannot_unset}", {"keys": cannot_unset})  

564
565     def builtin_umask(self, pipe="", args=[]):
566         # 设置系统umask的值
567         # subshell中的命令也会继续采用这一umask
568         # 若没有任何参数 · 则直接打印umask
569         # ! 在Windows上无法正常修改umask
570         if len(args) > 1:
571             # 参数数量为0或者1
572             raise UmaskException("Argument number is not correct, only one or  

573             zero expected.", {"type": "len"})
574         elif len(args) == 1:
575             try:
576                 # 用户输入的umask值不一定有效
577                 log.debug(f"Value of ARG: {COLOR.BOLD(args[0])}")
578                 umask = int(args[0], 8)
579                 log.debug(f"Value of ARG in int of 8: {COLOR.BOLD(umask)}")
580                 os.umask(umask)
581             except ValueError as e:
582                 raise UmaskException(e, {"type": "value"})
583         else:
584             # dummy parameter
585             old = os.umask(0)
586             log.debug(f"Value of OLD: {COLOR.BOLD(old)}")
587             os.umask(old)
588             # 返回umask的时候采用补零3位8进制格式
589             return "0o{:03o}".format(old)  

590
591     def builtin_printio(self, pipe="", args=[]):
592         # 开发者调试用命令 · 用于检查当前的exec文件重定向
593         # note: 由于我们要检查重定向 · 这里会直接打印到stdout而非输入输出文件
594         result = []
595         result.append(f"FILE NUMBER OF INPUT FILE:  

596             {COLOR.BOLD(self.input_file.fileno() if self.input_file is not None else  

597             sys.__stdin__.fileno())}, redirecting MyShell input to  

598             {COLOR.BOLD(self.input_file)})")
599         result.append(f"FILE NUMBER OF OUTPUT FILE:  

600             {COLOR.BOLD(self.output_file.fileno() if self.output_file is not None else  

601             sys.__stdout__.fileno())}, redirecting MyShell output to  

602             {COLOR.BOLD(self.output_file)})")
603         # ! debugging command, no redirection
```

```
596         print("\n".join(result), file=sys.__stdout__)
597
598     def builtin_exec(self, pipe="", args=[]):
599         # exec命令会修改MyShell命令的输入输出源
600         # note: 由于此命令的特殊性，我们会在该函数得到执行的上一层加入一些逻辑
601         if len(args) != 3:
602             log.critical("Internal error, builtin_exec should be executed with
603                           exactly three arguments")
604             raise ArgException("Internal error, builtin_exec should be executed
605                           with exactly three arguments")
606
607             log.debug(f"FILE NUMBER OF SYS.__STDIN__:
608 {COLOR.BOLD(sys.__stdin__.fileno())}")
609             log.debug(f"FILE NUMBER OF SYS.__STDOUT__:
610 {COLOR.BOLD(sys.__stdout__.fileno())}")
611             log.debug(f"FILE NUMBER OF SYS.__ STDERR__:
612 {COLOR.BOLD(sys.__stderr__.fileno())}")
613
614             # 我们刚刚已经保证走到这一步的参数数量为2
615             if args[0]:  # 若非空，尝试设置输入流
616                 try:
617                     # the function open will automatically raise FileNotFoundError
618                     new_file = open(args[0], "r", encoding="utf-8")
619
620                     # 新文件打开没有出错时我们才会关闭/修改旧文件
621                     # 无论如何，关闭原来已经打开的输入输出文件
622                     if self.input_file is not None:
623                         self.input_file.close()
624                     self.input_file = new_file
625                     log.debug(f"FILE NUMBER OF INPUT FILE:
626 {COLOR.BOLD(self.input_file.fileno())}")
627                     except FileNotFoundError as e:
628                         raise FileNotFoundException(e, {"type": "redi_in"})
629                     elif args[0] is None:
630                         log.debug("Doing nothing ... ")
631                     else:
632                         log.debug(f"Setting input file to None: stdin")
633                         self.input_file = None
634
635             if args[1]:  # 若非空，尝试设置输入流
636                 try:
637                     # the function open will automatically raise FileNotFoundError
638                     new_file = open(args[1], "a" if args[2] else "w",
639 encoding="utf-8")
640
641                     # 新文件打开没有出错时我们才会关闭/修改旧文件
642                     if self.output_file is not None:
643                         self.output_file.close()
644                     self.output_file = new_file
645                     log.debug(f"FILE NUMBER OF OUTPUT FILE:
646 {COLOR.BOLD(self.output_file.fileno())}")
647                     except FileNotFoundError as e:
648                         raise FileNotFoundException(e, {"type": "redi_out",
649 "redi_append": args[2]})
650                     elif args[1] is None:
651                         log.debug("Doing nothing ... ")
652                     else:
653                         log.debug(f"Setting output file to None: stdout")
```

```
645         self.output_file = None
646
647     self.builtin_printio()
648
649     def builtin_shift(self, pipe="", args=[]):
650         # 内置shift功能，在脚本调用时尤其有用
651         log.debug(f"Shift args are {COLOR.BOLD(args)}")
652         log.debug(f"Previously cmd_args are {COLOR.BOLD(self.cmd_args)}")
653
654         if len(args) > 1:
655             raise ArgException(f"Expecting zero or one argument(s), got"
656             f"{len(args)}", {"args": args})
657         elif not args:
658             # 空参数情况移动1位
659             args.append(1)
660
661             # $0会被保留
662             dollar_zero = self.cmd_args[0]
663
664             try:
665                 # 此时的shift时包含dollar_zero的
666                 int_val = int(args[0])
667                 if int_val < 0:
668                     raise ValueError("Non-negative int value expected")
669                 self.cmd_args = self.cmd_args[int_val::]
670             except IndexError:
671                 pass
672             except ValueError as e:
673                 raise ArgException("Non-negative int convertible argument expected."
674                 f"{e}", {"args": args})
675
676             # 替换掉dollar_zero位置的元素
677             if not len(self.cmd_args):
678                 self.cmd_args = [dollar_zero]
679             else:
680                 self.cmd_args[0] = dollar_zero
681
682             log.debug(f"Now cmd_args are {COLOR.BOLD(self.cmd_args)}")
683
684     def builtin_test(self, pipe="", args=[]):
685         # ! test函数的结合方向是从右向左
686         # todo: 实现带运算符顺序处理的逆波兰表达式（现在对-a和-o是一视同仁的）
687         # 结合递归调用
688         # builtin_test功能中使用到的等级函数
689         # 主要用于对操作符进行分类，方便调用，例如1，3为单目运算符
690         level = {
691             "(": 0, ")": 0,
692             "-z": 1, "-n": 1,
693             "=": 2, "!=": 2, "-eq": 2, "-ge": 2, "-gt": 2, "-le": 2, "-lt": 2,
694             "-ne": 2,
695             "!!": 3,
696             "-a": 4, "-o": 4,
697         }
698
699         # 单目运算符操作
700         def test_unary(operator, operand):
701             if operator == "-z":
702                 return not len(str(operand))
703
704
705         # 双目运算符操作
706         def test_binary(operator, left, right):
707             if operator == "-eq":
708                 return left == right
709             if operator == "-ne":
710                 return left != right
711             if operator == "-gt":
712                 return left > right
713             if operator == "-lt":
714                 return left < right
715             if operator == "-ge":
716                 return left >= right
717             if operator == "-le":
718                 return left <= right
719
720             if operator == "-z" or operator == "-n":
721                 return not len(str(left))
```

```

700         if operator == "-n":
701             return len(str(operand))
702         if operator == "!" :
703             return not bool(operand)
704
705         log.critical("Unrecognized operator in a place it shouldn't be")
706         raise TestException(f"Unrecognized unary operator \'{operator}\'")
707
708     # 双目运算符操作
709     def test_binary(op, lhs, rhs):
710         if op == "=":
711             return str(lhs) == str(rhs)
712         if op == "!=":
713             return str(lhs) != str(rhs)
714         if op == "-eq":
715             # todo: exception
716             return float(lhs) == float(rhs)
717         if op == "-ge":
718             return float(lhs) >= float(rhs)
719         if op == "-gt":
720             return float(lhs) > float(rhs)
721         if op == "-le":
722             return float(lhs) <= float(rhs)
723         if op == "-lt":
724             return float(lhs) < float(rhs)
725         if op == "-ne":
726             return float(lhs) != float(rhs)
727         if op == "-a":
728             return bool(lhs) and bool(rhs)
729         elif op == "-o":
730             return bool(lhs) or bool(rhs)
731
732         log.critical("Unrecognized operator in a place it shouldn't be")
733         raise TestException(f"Unrecognized binary operator \'{operator}\'")
734
735     def expand_expr(args):
736         # ! we combine from the right, that is the right most value are
737         # evaluated first
738         try:
739             ind = 0
740             if len(args) == 1:
741                 return args[0]
742
743             # 非运算符 · 视为双目运算
744             if args[ind] not in level:
745                 lhs = args[ind]
746                 op = args[ind+1]
747                 # 这句话保证了从右向左结合
748                 rhs = expand_expr(args[ind+2::])
749                 return test_binary(op, lhs, rhs)
750
751             # match parentheses
752             if args[ind] == "(":
753                 # 对于小括号 · 我们将其视为一个整体
754                 org = ind
755                 count = 1
756                 while count:
757                     ind += 1

```

```
757             if args[ind] == "(":
758                 count += 1
759             elif args[ind] == ")":
760                 count -= 1
761             lhs = expand_expr(args[org+1:ind])
762             if ind == len(args)-1:
763                 return lhs
764             op = args[ind+1]
765             # 这句话保证了从右向左结合
766             rhs = expand_expr(args[ind+2::])
767             return test_binary(op, lhs, rhs)
768
769     if level[args[ind]] in [1, 3]:
770         # 对于单目运算符，我们也将其视为一个整体
771         op = args[ind]
772         oa, ind = get_one(args[ind+1::])
773         lhs = test_unary(op, oa)
774         if ind == len(args)-1:
775             return lhs
776         op = args[ind+1]
777         # 这句话保证了从右向左结合
778         rhs = expand_expr(args[ind+2::])
779         return test_binary(op, lhs, rhs)
780
781     # note: 在此处理整个函数调用过程中可能的错误
782     # int·bool无法转换/解释等
783     except (ValueError, KeyError) as e:
784         # log.error(f"{e}")
785         raise TestException(e)
786
787     def get_one(args):
788         # 获取一位bool值，并返回下一个有效值的位置
789         ind = 0
790         if len(args) == 1 or args[ind] not in level:
791             return args[0], 1
792
793         if args[ind] == "(":
794             org = ind
795             while args[ind] != ")":
796                 ind += 1
797             return expand_expr(args[org+1:ind]), ind+1
798
799         if level[args[ind]] in [1, 3]:
800             op = args[ind]
801             oa, ind = get_one(args[ind+1::])
802             return test_unary(op, oa), ind+1
803
804     # we can only use string to pass values
805     # 返回布尔值的字符串表示
806     try:
807         result = str(bool(expand_expr(args)))
808         return result
809     except IndexError as e:
810         raise TestException(f"Unrecognized test expression, check your
syntax. {e}")
811
812     def builtin_sleep(self, pipe="", args=[]):
813         if os.name == "nt":
```

```
814         # ! Windows系统没有相应的sleep命令，但Python存在相应的接口
815         try:
816             if len(args) != 1:
817                 raise ArgException("Exactly one argument expected")
818             if args[0].endswith("s"):
819                 value = float(args[0][0:-1])
820             else:
821                 value = float(args[0])
822             time.sleep(value)
823         except ValueError as e:
824             raise SleepException(f"Unrecognized sleep time format, are you
on NT? Use second as unit and put s at the back of the time string. {e}")
825         else:
826             self.subshell(target="sleep", args=args, pipe=pipe)
827
828     def builtin_time(self, pipe="", args=[]):
829         # 显示当前的时间
830         return str(datetime.datetime.now())
831
832     def builtin_dummy(self, pipe="", args=[]):
833         # 一个内置的dummy命令，用于测试是否可以正常触发suspension
834         print("builtin_dummy: before any input requirements")
835         print(input("dummy1> "))
836         print(input("dummy2> "))
837         print(input("dummy3> "))
838         print(input("dummyend> "))
839         result = input("dummy_content> ")
840         return result
841
842     def builtin_help(self, pipe="", args=[]):
843         # 在线帮助函数
844         # todo: 写好在线帮助
845         help_dict = {
846             # CONTENT OMITTED HERE
847             # CHECK THE MANUAL IN THE ABOVE TEXT
848             # OR CHECK THE SOURCE CODE FILE DIRECTLY
849             # IT'S TOOOOOOOO LONG TO BE PUT IN THE REPORT
850             .....
851             .....
852             .....
853         }
854         if not args:
855             # length is zero
856             result = help_dict["MyShell"]
857         elif len(args) == 1:
858             try:
859                 result = help_dict[args[0]]
860             except KeyError as e:
861                 # raise HelpException(f"Cannot find manual entry for {args[0]}.
{type(e).__name__}: {e}")
862                 log.warning("Cannot find help page in builtin dict, trying with
man command")
863                 try:
864                     self.subshell(target="man", args=[args[0]], piping_in=False,
piping_out=False)
865                 except CalledProcessException as e:
866                     raise HelpException(f"Cannot find help page for {args[0]}")
867             return
```

```
868         else:
869             raise ArgException("Help takes one or zero argument.", {"type":
870 "help"})
871
872             # self.subshell(target="more", pipe=result, piping_in=True,
873             # piping_out=False)
874             self.subshell(target="less", pipe=result, piping_in=True,
875             piping_out=False)
876
877             return result
878
879
880     def builtin_verbose(self, pipe="", args=[]):
881         # developer command: setting debug log printing level
882         supported = {
883             "-e": "ERROR",
884             "-w": "WARNING",
885             "-i": "INFO",
886             "-d": "DEBUG"
887         }
888
889         if len(args) > 1:
890             raise ArgException("This command takes zero or one argument.")
891         elif not args:
892             l = coloredlogs.get_level()
893             levels = [COLOR.BOLD(i) for i, k in
894             coloredlogs.find_defined_levels().items() if k == l]
895             return "Current logging level: " + ", ".join(levels)
896         elif args[0] not in supported:
897             raise ArgException(f"Unrecognized argument: {args[0]}. We're
898 supporting only {supported}.")
899
900
901     def subshell(self, pipe="", target="", args=[], piping_in=False,
902 piping_out=False, io_control=False):
903         # 运行外部程序
904         # 根据需要调整输入输出
905
906         # 是一个内部命令 · 通过调用subprocess的接口完成外部程序的调用
907         if not target:
908             raise EmptyException(f"Command \'{target}\' is empty", {"type":
909 "subshell"})
910             to_run = [target] + args
911             log.debug(f"Running in subprocess: {COLOR.BOLD(to_run)}")
912             try:
913                 log.debug(f"EXEC OI controller is: {COLOR.BOLD(str(self.input_file)
914 + ' ' + str(self.output_file)))}")
915                 result = None
916                 if io_control:
917                     piping_in = True
918                     # waits for the process to end
919                     # 若需要通过管道传递相关内容 · 则使用PIPE
920                     # ! windows中有些内置命令是无法通过subprocess调用的 · 例如type等cmd.exe内置
921                     命令
922                     p = subprocess.Popen(
923                         to_run,
924                         stdin=PIPE if piping_in else self.input_file,
925                         stdout=PIPE if piping_out else self.output_file,
926                         stderr=STDOUT, encoding="utf-8",
927                         shell=True)
```

```
917             env=dict(os.environ, PARENT=self.vars["SHELL"])
918         )
919
920         self.subp = p
921         result, error = p.communicate(pipe) # 如果我们选择不使用PIPE，这里传入
空字符串也不会有任何影响
922         if p.returncode != 0:
923             log.warning("The subprocess is not returning zero exit code")
924             raise CalledProcessException("None zero return code
encountered")
925     finally:
926         # 我们使用try block的原因在于无论如何都要清空一下self.subp
927         self.subp = None
928
929         # 若通过subshell调用，则直接将结果以字符串形式返回
930         return result
931
932     def path(self):
933         # callable PATH function，随着系统变量的改变而改变
934         return self.vars["PATH"]
935
936     def home(self):
937         # callable HOME function，随着系统变量的改变而改变
938         return self.vars['HOME']
939
940     def cwd(self):
941         # callable PWD function，随着系统变量的改变而改变
942         cwd = os.getcwd()
943         if cwd.startswith(self.home()):
944             cwd = f"~{cwd[len(self.home()):]}"
945         return cwd
946
947     def user(self):
948         # callable USER function，随着系统变量的改变而改变
949         return getpass.getuser()
950
951     def location(self):
952         # callable LOCATION function，随着系统变量的改变而改变
953         return platform.node()
954
955     def prompt(self):
956         # 返回将要打印到屏幕的命令提示符
957         # 包含：用户@地点 当前目录 当前时间 提示符
958         # ($CONDA_DEFAULT_ENV) $USER@location $PWD time("%H:%M:%S") $PS1
959         prompt = f"{COLOR.BEIGE(self.user())+'@'+self.location()}"
960         prompt += f"{COLOR.BOLD(COLOR.BLUE(self.cwd()))}"
961         prompt += f"{COLOR.BOLD(datetime.datetime.now().strftime('%H:%M:%S'))}"
962         prompt += f"{COLOR.BOLD(COLOR.YELLOW(self.vars['PS1']) if 'PS1' in self.vars else '$')}" "
963         # log.debug(repr(prompt))
964     try:
965         conda = os.environ["CONDA_DEFAULT_ENV"]
966         prompt = f"({conda}) {prompt}"
967     except KeyError:
968         pass # not a conda environment
969     return prompt
970
971     def execute(self, command, pipe=""):
972         # 执行一个已经被格式化的命令
```

```
970     # 命令以dict形式传入，其中exec代表命令本身，args代表命令参数
971     # 还有一些其他控制参数，例如重定向或者管道操作
972     # 本函数还包含了对于exec命令的特殊处理（我们会提前处理重定向操作，而非交给命令本身，因此需要特殊化参数才可正常传入）
973     log.info(f"Executing command {COLOR.BOLD(command['exec'])}")
974     log.info(f"Arguments are {COLOR.BOLD(command['args'])}")
975
976     # pipe或者重定向输入只能有一个
977     if command["pipe_in"] and command["redi_in"]:
978         raise MultipleInputException("Redirection and pipe are set as input at the same time.")
979
980     # piping_in/piping_out主要用于subshell的处理
981     command["piping_in"] = command["pipe_in"] or command["redi_in"]
982     command["piping_out"] = command["redi_out"] or command["pipe_out"]
983
984     # 处理输入重定向
985     # to the command itself, it doesn't matter whether the input comes from a pipe or file
986     if command["redi_in"]:
987         file_path = command["redi_in"]
988         try:
989             # the function open will automatically raise FileNotFoundError
990             f = open(file_path, "r")
991             pipe = f.read()
992             f.close()
993         except FileNotFoundError as e:
994             raise FileNotFoundError(e, {"type": "redi_in"})
995
996     # if we've specified input file in some thing
997     if self.input_file is not None:
998         sys.stdin = self.input_file
999     # if we've specified output file in some thing
1000    if self.output_file is not None:
1001        sys.stdout = self.output_file
1002
1003    # actual execution
1004    result = ""
1005    try:
1006        if command["exec"] == "exec":
1007            if len(command["args"]):
1008                raise ArgException("exec command takes no argument")
1009            # setting redi_files to arguments
1010            # something might change
1011            self.builtin_exec(args=[command["redi_in"], command["redi_out"],
1012                                command["redi_append"]])
1013            elif command["exec"] in self.builtins.keys():
1014                log.debug("This is a builtin command.")
1015                # executing as static method, calling with self variable
1016                result = self.builtins[command["exec"]](self, pipe=pipe,
1017                                                args=command['args'])
1018            else:
1019                log.debug("This is not a builtin command.")
1020                try:
1021                    result = self.subshell(pipe, command["exec"],
1022                                         command['args'], piping_in=command["piping_in"],
1023                                         piping_out=command["piping_out"], io_control=command["io_control"])
1024                except FileNotFoundError as e:
```

```
1021             raise FileNotFoundError(e, {"type": "subshell"})
1022     finally:
1023         # 我们使用try block的原因在于无论如何都要还原一下sys.stdin, sys.stdout
1024         sys.stdin = sys.__stdin__
1025         sys.stdout = sys.__stdout__
1026
1027     # 处理输出重定向
1028     if command['redi_out']:
1029         log.debug(f"User want to redirect the output to
{COLOR.BOLD(command['redi_out'])}")
1030         try:
1031             f = open(command["redi_out"], "a" if command["redi_append"] else
1032 "w")
1033             f.write(result)
1034             f.close()
1035         except FileNotFoundError as e:
1036             raise FileNotFoundError(e, {"type": "redi_out",
1037 "redi_append": command["redi_append"]})
1038         return result
1039
1040     # 若用户需要进行管道操作，就不打印相关内容，直接以字符串返回获得的内容
1041     if command['pipe_out']:
1042         log.debug(f"User want to pipe the IO")
1043         return result
1044
1045     if result is not None:
1046         print(result, end="" if result.endswith("\n") or not len(result)
1047 else "\n", file=self.output_file)
1048         # return result # won't be used anymore
1049
1050     def command_prompt(self):
1051         # 命令提示符打印
1052         # note: readline quirk
1053         # strange error if we use input
1054         # the input and readline prompt seems to be counting color char as one
1055         # of the line chars
1056         # well it turns out to be a quirk of readline
1057         # we've fixed it in COLOR.py
1058         try:
1059             command = input(self.prompt()).strip()
1060         except EOFError:
1061             # 在程序以交互模式运行，但是读入源非标准输入，就有可能在读完全部命令后得到
1062             # EOFError
1063             # 我们选择在这种情况下退出交互模式
1064             # note: 这也意味着在普通的交互模式下输入Ctrl+D也会使shell退出
1065             print()
1066             log.warning("Getting EOF from command prompt, exiting ... ")
1067             return self.run_command("exit")
1068
1069     @staticmethod
1070     def run_command_wrap(count, shell, args, job, queue, jobs, status_dict):
1071         # 用于后台程序管理
1072         # 我们不仅可以在后台运行外部命令，程序自身命令也可以后台执行
```

```
1073     # 且管道、重定向等操作都由MyShell执行，这与直接刷出一个新的subprocess完全不同
1074     # 我们没有在这种情况下借用已有shell的功能
1075     log.debug(f"Wrapper [{count}] called with {COLOR.BOLD(f'{shell} and
1076     {args}')})")
1077
1078     # ! DOESN'T WORK ON WINDOWS!
1079     # ! WILL CREATE ZOMBIE PROCESS!
1080
1081     # 为了在信号处理过程中正确杀死自身进程
1082     my_pid = os.getpid()
1083
1084     # signal handler · 遇到signal.SIGTERM后杀死可能正在运行的子进程
1085     def exit_subp(sig, frame):
1086         if shell.subp is not None:
1087             log.warning("Killing a still running subprocess ... ")
1088             os.kill(shell.subp.pid, signal.SIGKILL)
1089             log.debug(f"Getting signal: {COLOR.BOLD(sig)}")
1090             log.debug(f"Are you: {COLOR.BOLD(signal.SIGTERM)}")
1091         if sig == signal.SIGTERM:
1092             # 杀死自身进程
1093             log.warning(f"Terminating job [{count}] handler process by
1094             signal ... ")
1095             os.kill(my_pid, signal.SIGKILL)
1096
1097     # 注册信号处理器
1098     # note: multiprocessing.Process.daemon = True时 · 若父进程退出 · 该信号会被触发
1099     signal.signal(signal.SIGTERM, exit_subp)
1100
1101     # note: 正文
1102     # 通过Wrapper来控制内部命令的suspension
1103     if sys.stdin is not None:
1104         sys.stdin.close()
1105     # stdin的文件号
1106     stdin = open(0)
1107     buffer = stdin.detach()
1108     wrapper = MyShell.StdinWrapper(queue, count, job, status_dict, buffer)
1109     sys.stdin = wrapper
1110
1111     try:
1112         # 真正的执行过程
1113         # 详见run_command
1114         # 包括解析命令/执行命令/错误处理
1115         # 此时不会执行后台逻辑 · 跳过parse阶段
1116         shell.run_command(args, io_control=True)
1117     finally:
1118         # 正常情况下不会出现raise的情况
1119         # 无论如何 · 都要退出当前的job
1120         print(shell.job_status_fmt(count, "finished", job))
1121         queue.put("dummy")
1122         del jobs[count]
1123
1124     def term_all(self):
1125         # 杀死jobs数组中所有的剩余任务
1126         jobs = self.jobs
1127         for job in jobs:
1128             os.kill(self.process[job].pid, signal.SIGTERM)
1129             # 我们会将job相关信息直接打印到屏幕上
1130             print(self.job_status_fmt(job, "terminated", jobs[job]))
```

```
1129
1130     def run_command(self, command, io_control=False):
1131         # 解析命令 · 执行命令 · 错误处理 · 后台任务
1132         try:
1133             # commands是一个command数组 · 包含具体的方便读取的命令格式信息
1134             # 由于存在PIPE调用的可能性 · 我们对commands创建了数组
1135             # parse会将命令字符串解释为完整的命令
1136             commands, is_bg = self.parse(command)
1137             if is_bg:
1138                 # ! changes made in subprocess is totally within the subprocess
1139                 only
1140
1141                 # job_counter实际上是一个函数
1142                 # 会获取当前没有被使用掉的最小的job编号 · 从0开始
1143                 str_cnt = self.job_counter
1144
1145                 # 创建相关内容
1146                 self.jobs[str_cnt] = command
1147                 self.queues[str_cnt] = Queue()
1148                 self.status_dict[str_cnt] = "running"
1149
1150                 # 备份相关内容以便deepcopy时删除
1151                 queues_bak = self.queues
1152                 process_bak = self.process
1153                 jobs_bak = self.jobs
1154                 status_dict_bak = self.status_dict
1155                 input_bak = self.input_file
1156                 output_bak = self.output_file
1157
1158                 # note: 为了规避一些pickle error · 我们会删除这里的：
1159                 # thread_lock, multiprocessing.Manager, multiprocessing.Queue,
1160                 Manager.dict, io.TextWrapper
1161                 del self.queues
1162                 del self.process
1163                 del self.jobs
1164                 del self.status_dict
1165                 del self.input_file
1166                 del self.output_file
1167
1168                 # 获取一个deepcopy的shell以供后台程序执行
1169                 clean_self = copy.deepcopy(self)
1170
1171                 # 填充默认内容
1172                 # ! 我们默认后台程序不会尝试再次构建后台的后台指令
1173                 clean_self.queues = {}
1174                 clean_self.jobs = {}
1175                 clean_self.process = {}
1176                 clean_self.status_dict = {}
1177                 clean_self.input_file = None
1178                 clean_self.output_file = None
1179
1180                 # 还原到deepcopy以前的状态
1181                 self.queues = queues_bak
1182                 self.process = process_bak
1183                 self.jobs = jobs_bak
1184                 self.status_dict = status_dict_bak
1185                 self.input_file = input_bak
1186                 self.output_file = output_bak
```

```
1185
1186          # 使用deepcopy初来的clean_self来构建新的进程
1187          p = Process(target=self.run_command_wrap, args=(str_cnt,
1188          clean_self, command[0:-1], self.jobs[str_cnt], self.queues[str_cnt], self.jobs,
1189          self.status_dict), name=command)
1190
1191          # 添加到进程管理中，方便kill命令等的执行
1192          self.process[str_cnt] = p
1193
1194          # ! so the multiprocessing process won't actually be totally
1195          # gone if the main process is still around
1196          # but it will be terminated if demanded so (ps command can see
1197          # it as <defunc>, but not a zombie)
1198          # 保证主进程退出后，下面的小进程也会退出
1199          p.daemon = True
1200
1201          # 开始运行
1202          p.start()
1203          log.debug(f"We've spawned the job in a Process for command:
1204          {COLOR.BOLD(p.name)}")
1205
1206      else:
1207          # 一般命令的执行
1208          result = None # so that the first piping is directly from stdin
1209          if io_control:
1210              for command in commands:
1211                  command["io_control"] = True
1212          else:
1213              for command in commands:
1214                  command["io_control"] = False
1215          for cidx, command in enumerate(commands):
1216              result = self.execute(command, pipe=result)
1217              # log.debug(f"Getting result: {COLOR.BOLD(result)}")
1218
1219      except ExitException as e:
1220          # 我们通过异常来处理程序的退出命令
1221          # 包括quit和exit
1222          # 某种意义上Ctrl+D代表的EOF也是一种推出指令
1223          log.debug("User is exiting ... ")
1224          log.debug(f"Exception says: {e}")
1225          log.info("Bye")
1226          self.term_all()
1227
1228      return True
1229
1230  except EmptyException as e:
1231      # EmptyException较为特殊，需要单独处理
1232      # 因为错误等级会根据type不同而变换
1233      log.debug("The command is empty ... ")
1234      log.info(f"Exception says: {e}")
1235      if e.errors["type"] == "pipe":
1236          log.error(f"Your pipe is incomplete. {e}")
1237      elif e.errors["type"] == "subshell":
1238          log.warning(f"Your command is empty. Did you use an empty var?
1239          {e}")
1240      elif e.errors["type"] == "empty":
1241          log.info(f"Your command is empty. {e}")
1242
1243  except MyShellException as e:
1244      # this means that we've got a none zero return code / execution
1245      # Failure
1246      error_cmd = command['exec'] if isinstance(command, dict) else
1247      command
```

```
1235         line_end = "\n"
1236         log.error(f"Cannot successfully execute command \'{error_cmd}\'."
1237         Exception is: {line_end}{COLOR.BOLD(type(e).__name__ + ': ' + str(e) + line_end
1238         + 'Extra info: ' + str(e.errors))}"))
1239     except Exception as e:
1240         log.error(f"Unhandled error. {traceback.format_exc()}")
1241     finally:
1242         # 同步status_dict, queues, jobs
1243         self.cleanup_jobs() # always clean up
1244
1245     # 不返回退出指令
1246     return False
1247
1248
1249     def parse(self, command):
1250         # 命令解释器
1251
1252         # quote函数会处理引号/变量替换/特殊变量转义/~替换
1253         inputs = self.quote(command) # splitting by whitespace and processing
1254         variables, quotes
1255
1256         # 将命令分解 · 进行piping调用
1257         # splitting command of pipling
1258         commands = []
1259         command = []
1260
1261         for word in inputs:
1262             if word != "|":
1263                 command.append(word)
1264             else:
1265                 commands.append(command)
1266                 command = []
1267
1268         commands.append(command)
1269
1270         # 对每一个个别的命令进行解析
1271         is_bg = False # 是否后台执行是整个命令的设置 · 只需要一个
1272         parsed_commands = []
1273
1274
1275         for cidx, command in enumerate(commands):
1276             parsed_command = self.parsed_clean()
1277             # 可能用户只是敲了一下回车
1278             # 也可能用户的某一个管道环节是空的 : 这是不允许的
1279             # 我们通过type传递相关信息
1280             if not len(command):
1281                 raise EmptyException(f"Command at position {cidx} is empty",
1282 {"type": "pipe" if len(commands) > 1 else "empty"})
1283             parsed_command["pipe_in"] = (cidx > 0)
1284             parsed_command["pipe_out"] = (cidx != len(commands)-1)
1285             index = 0
1286
1287             while index < len(command):
1288                 if not index: # this should have a larger priority
1289                     # 整个命令的第一个词汇是执行目标
1290                     parsed_command["exec"] = command[index]
1291                 elif command[index] == "<":
1292                     # 输入重定向
1293                     if index == len(command) - 1:
1294                         raise SetPairUnmatchedException("Cannot match
1295 redirection input file with < sign.", {"type": "redi"})
1296                     parsed_command["redi_in"] = command[index+1]
1297                     index += 1
```

```

1288             elif command[index] == ">":
1289                 # 输出重定向
1290                 if index == len(command) - 1:
1291                     raise SetPairUnmatchedException("Cannot match
 redirection output file with > sign.", {"type": "redi"})
1292                     parsed_command["redi_out"] = command[index+1]
1293                     index += 1
1294             elif command[index] == ">>":
1295                 # 输出重定向：添加型
1296                 if index == len(command) - 1:
1297                     raise SetPairUnmatchedException("Cannot match
 redirection output file with >> sign.", {"type": "redi"})
1298                     parsed_command["redi_out"] = command[index+1]
1299                     parsed_command["redi_append"] = True
1300                     index += 1
1301             elif command[index] == "&":
1302                 if index != len(command)-1 or cidx != len(commands) - 1:
1303                     # & not at the end
1304                     raise UnexpectedAndException("Syntax error, unexpected &
 found")
1305
1306                 # todo: communication
1307                 is_bg = True
1308                 log.info("User want this to run in background.")
1309             else:
1310                 parsed_command["args"].append(command[index])
1311                 index += 1
1312             parsed_commands.append(parsed_command)
1313
1314     return parsed_commands, is_bg
1315
1316     def quote(self, command, quote=True):
1317         # 可以被递归调用的引号处理功能
1318         # command should already been splitted by word
1319         # mark: this structure makes it possible that the arg is keep in one
 place if using quote
1320         # by not brutally expanding on every possible environment variables
1321         command = command.split()
1322         # 清除所有的注释
1323         # note: 就像一般的shell，注释前面要有一个空格
1324         comment = [i for i, v in enumerate(command) if v.startswith("#")]
1325         if len(comment):
1326             command = command[0:comment[0]]
1327
1328         quote_stack = []
1329         index = 0
1330         while index < len(command):
1331
1332             # 这一个if-else代码块保证所有内容被解析过一次，且仅解析一次，调用self.expand
 函数
1333             # 并且若解析后的内容有引号，也不会影响程序的正常执行，因为我们是在处理引号后解
析变量的
1334             # note: 也就是说如果你的变量中存在着可以被解析为变量的内容，它们不会被解析，以
防止无限递归解析
1335             if command[index].count("\\"") >= 1:
1336                 # should remove all quotes here
1337                 quote_count = command[index].count("\\"")
1338                 log.debug("Trying to remove quote ... ")

```

```
1339             splitted = command[index].split("``")
1340             # there were not space at the beginning
1341             # expand函数会进行变量和~的替换
1342             # 我们会将读入的内容按照引号数量拆分，然后进行解析
1343             # 最后合并成一个长字符串
1344             # 并通过quote_count来和下面的代码沟通
1345             command[index] = "``".join([self.expand(split) for split in
1346                                         splitted])
1347         else:
1348             command[index] = self.expand(command[index])
1349             quote_count = 0
1350
1351             if quote_count % 2: # previous quote_count
1352                 # 引号分词中出现了空格
1353                 if quote_stack:
1354                     # except the last char
1355                     quote_stack.append(command[index])
1356                     # recursion to process $ and "" in the already processed ""
1357                     command[index] = " ``".join(quote_stack)
1358                     quote_stack = ""
1359                     # index should continue to be added in the end
1360             else:
1361                 log.debug("Trying to match quote ... ")
1362                 quote_stack.append(command[index])
1363                 if index == len(command)-1:
1364                     raise QuoteUnmatchedException("Cannot match the quote
for the last argument")
1365                     command = command[0:index] + command[index+1::]
1366                     index -= 1
1367             elif quote_stack:
1368                 # 引号分词没有空格，加入到上一个quote组中
1369                 # no quote but
1370                 quote_stack.append(command[index])
1371                 if index == len(command)-1:
1372                     raise QuoteUnmatchedException("Cannot match the quote for a
series of arguments")
1373                     command = command[0:index] + command[index+1::]
1374                     index -= 1 # index should stay the same
1375                     index += 1
1376
1377             return [i.replace("\\~", "~").replace("\\$", "$").replace("\\#", "#")
for i in command]
1378
1379     def expand(self, string):
1380         def get_key(key):
1381             # 第一个字符$不能被用于寻找变量
1382             key = key[1::]
1383             log.debug(f"Trying to get variable {COLOR.BOLD(key)}")
1384             var = self.vars[key]
1385             # splitting the expanded command since it might contain some
information
1386             return var
1387
1388     def get_home(key):
1389         log.debug(f"GETTING HOME SIGN: {COLOR.BOLD(key)}")
1390         if key == "~":
1391             return self.home()
1392         else:
```

```
1392             return key
1393
1394     # 通过调用regex的替换命令 ( 其实是我们自己实现的 )
1395     # ! re.sub在处理utf-8字符串时有难以预料的错误
1396     # ! if you use re.sub on windows, strange things can happen
1397     # *nix可以正常运行 · 但Windows下报错
1398     string = self.sub_re(r"(?<!\\)\$\w+", string, get_key)
1399     string = self.sub_re(r"(?<!\\)~", string, get_home)
1400
1401     return string
1402
1403 @staticmethod
1404 def sub_re(pattern, string, method):
1405     # 根据regex寻找匹配子串 · 然后通过调用method函数进行替换的静态方法
1406
1407     # 所有的匹配结果
1408     var_list = [(m.start(0), m.end(0)) for m in re.finditer(pattern,
1409     string)]
1410
1411     # 拆分为替换后的内容
1412     str_list = []
1413     prev = [0, 0]
1414     for start, end in var_list:
1415         str_list.append(string[prev[1]:start])
1416         str_list.append(string[start:end])
1417         prev = [start, end]
1418     str_list.append(string[prev[1]:])
1419
1420     # log.debug(f"The splitted vars are {COLOR.BOLD(str_list)}")
1421
1422     # 对于每一个匹配成功的字串 · 进行method(key)的调用
1423     for i in range(1, len(str_list), 2):
1424         str_list[i] = method(str_list[i])
1425         # to result in index staying the same
1426
1427     string = "".join(str_list)
1428     return string
1429
1430 def parsed_clean(self):
1431     # 返回一个干净的指令
1432     parsed_command = {
1433         "exec": "",
1434         "args": [],
1435         "pipe_in": False,
1436         "pipe_out": False,
1437         "redi_in": None,
1438         "redi_out": None,
1439         "redi_append": False,
1440     }
1441     return parsed_command
1442
1443 if __name__ == "__main__":
1444     # cat < dummy.mysh | wc > /dev/tty | echo "zy" > result.out | sha256sum | tr
1445     -d " " >> result.out | wc | cat result.out | wc | cat result.out
1446     # test ! -z "" -a ( -n "1" -o 1 -ge 1 ) -o 2 -ne 1 # False, -a -o from right
1447     to left
1448     # test ( ! -z "" -a ( -n "1" -o 1 -ge 1 ) ) -o 2 -ne 1 # True
```

```
1447      # ./MyShell.py -w dummy.mysh -a foo bar foobar hello world linux linus
1448      PyTorch CS231n
1449
1450      # 调用此脚本时候传入-h以查看帮助
1451      parser = argparse.ArgumentParser(description='MyShell by
1452          xudenden@gmail.com')
1453      parser.add_argument('-f', metavar='F', type=str, nargs='?', help='the batch
1454          file to be executed')
1455      parser.add_argument('-a', metavar='A', type=str, nargs='*', help='command
1456          line arguments to batch file')
1457      parser.add_argument('-e', help='enable error level debugging info log',
1458          action='store_true')
1459      parser.add_argument('-w', help='enable warning level debugging info log',
1460          action='store_true')
1461      parser.add_argument('-i', help='enable info level debugging info log',
1462          action='store_true')
1463      parser.add_argument('-d', help='enable debug(verbose) level debugging info
1464          log', action='store_true')
1465
1466      args = parser.parse_args()
1467      # args为MyShell的命令行参数
1468      # 处理MyShell的命令行参数
1469      if args.e:
1470          coloredlogs.set_level("ERROR")
1471      if args.w:
1472          coloredlogs.set_level("WARNING")
1473      if args.i:
1474          coloredlogs.set_level("INFO")
1475      if args.d:
1476          coloredlogs.set_level("DEBUG")
1477
1478      # 命令行参数的第一个为当前脚本的路径(脚本模式) 或MyShell的路径(交互模式)
1479      cmd_args = [os.path.abspath(args.f) if args.f else
1480          os.path.abspath(__file__)]
1481      if args.a is not None:
1482          cmd_args += args.a
1483      myshell = MyShell(cmd_args=cmd_args)
1484      log.debug(f"Getting user command line argument(s): {COLOR.BOLD(args)}")
1485
1486      # ! 一次只准执行一个脚本
1487      if args.f:
1488          try:
1489              # ! using utf-8
1490              f = open(args.f, encoding="utf-8") # opening the file specified
1491              for line in f:
1492                  line = line.strip()
1493                  result = myshell.run_command(line) # execute line by line
1494                  if result: # the execution of the file should be terminated
1495                      break
1496              myshell.run_command("exit") # call exit at the end of the shell
1497              execution
1498          except FileNotFoundError as e:
1499              log.error(f"Cannot find the file specified for batch processing: \
1500 {args.f}\\". {e}")
1501          else:
1502              # 直接使用交互模式
1503              myshell()
```

讨论/心得

心得实在是太多了.....竟然一时不知道从何说起

关于Python语言的一些感想

在尝试实现MyShell的过程中，我们一开始选定的语言是C/C++，原因也很明显，他们最深入系统底层且速度客观。

开发一段时间后我们决定更换开发语言。因为纵使在系统调用上使用C语言实现会方便许多，在太多太多其他小部分的功能却需要重复制造太多/太多的轮子。单单在读入用户输入这一点上，从零开始实现或许就需要巨大的人力物力投入，才能达到

1. 有历史记录
2. 方向键会被正确解释的程度

否则在输入过程中键入方向键时，Linux会将其解释为一个单独的特殊字符。

```
(aliconda) ✨ root@aliciecs ➤ ~ ipython
Python 3.8.2 (default, Mar 26 2020, 15:53:00)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.13.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: input("")
hello^[[D^[[C^[[C^[[A^[[A^[[B^[[B]
```

采用Python的第二个原因是：跨平台性。

纵使MyShell在设计时没有以此为根本目的，但谁不想要一个可以跨平台运行的Shell呢？

我们说，只要涉及到系统调用，Win32与*nix的很多接口不尽相同，但Python在某种程度上缓解了这一问题

例如，我们可以通过 subprocess 或者 multiprocessing 包对很多内容进行统一管理（虽然在不同系统中还是会有微小的使用差别，在实际实现MyShell过程中，需要在各种奇怪的位置用一些奇怪的手段来规避这些错误）

```
1 # 备份相关内容以便deepcopy时删除
2 queues_bak = self.queues
3 process_bak = self.process
4 jobs_bak = self.jobs
5 status_dict_bak = self.status_dict
6 input_bak = self.input_file
7 output_bak = self.output_file
8
9 # note: 为了规避一些pickle error，我们会删除这里的：
10 # thread_lock, multiprocessing.Manager, multiprocessing.Queue, Manager.dict,
11 # io.TextWrapper
12 del self.queues
13 del self.process
14 del self.jobs
15 del self.status_dict
16 del self.input_file
17 del self.output_file
```

```
18 # 获取一个deepcopy的shell以供后台程序执行
19 clean_self = copy.deepcopy(self)
20
21 # 填充默认内容
22 # ! 我们默认后台程序不会尝试再次构建后台的后台指令
23 clean_self.queues = {}
24 clean_self.jobs = {}
25 clean_self.process = {}
26 clean_self.status_dict = {}
27 clean_self.input_file = None
28 clean_self.output_file = None
29
30 # 还原到deepcopy以前的状态
31 self.queues = queues_bak
32 self.process = process_bak
33 self.jobs = jobs_bak
34 self.status_dict = status_dict_bak
35 self.input_file = input_bak
36 self.output_file = output_bak
```

其中一个极为奇怪的问题便是：在Win32系统下调用multiprocessing.Manager会导致Pickle无法正常复制Object

我们不得不承认，相对于C/C++，Python是一种更加高级的语言，它可以让我们的代码变得精简易读。但或许用脚本语言实现的Shell永远无法很好的进入真正的生产环境（速度/与系统底层的接口等问题）。

但我们实现MyShell的根本目的在于学习，因此很多情况下，相对于直接使用系统提供的接口，我们使用 模拟器 的方式来模拟功能。而一个抽象程度更高的语言显然可以让人们在 模拟 Shell功能时有更方便的接口/逻辑。

关于工具.....

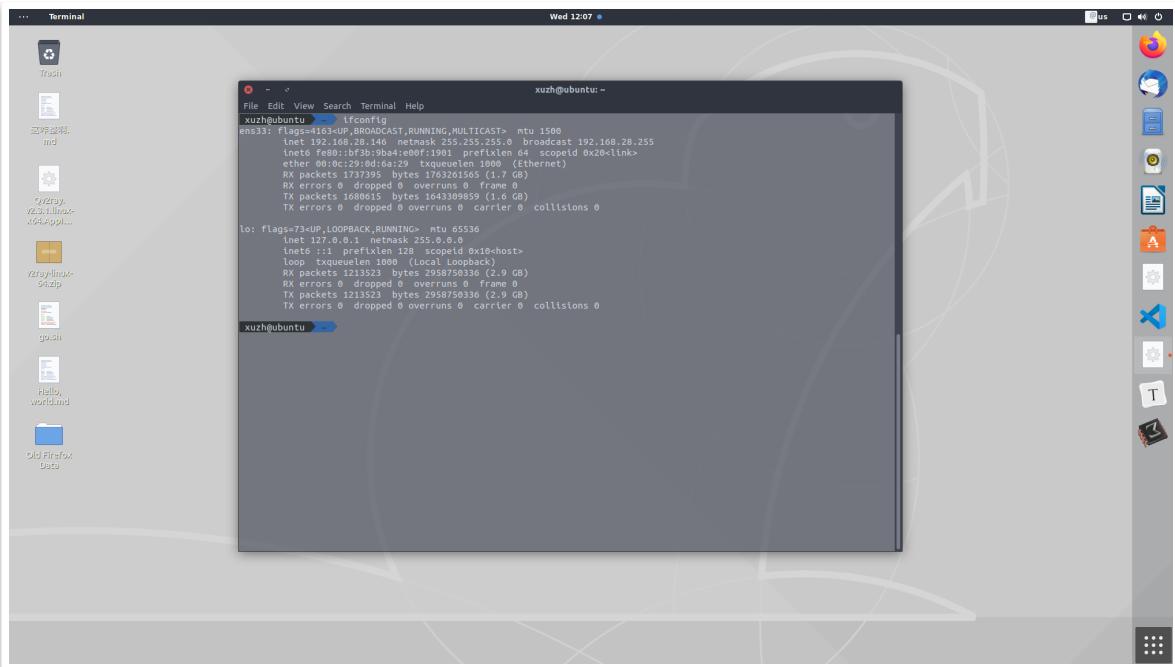
正如系统环境/实验环境介绍的环节中我们提到的，在实现第二个实验的过程中，我们详细介绍本次实验中用到的各色环境。

实际使用过程中，我们主要在 Ubuntu 虚拟机，阿里云 CentOS 7，搬瓦工 CentOS 7，MacOS Catalina 上进行试验。

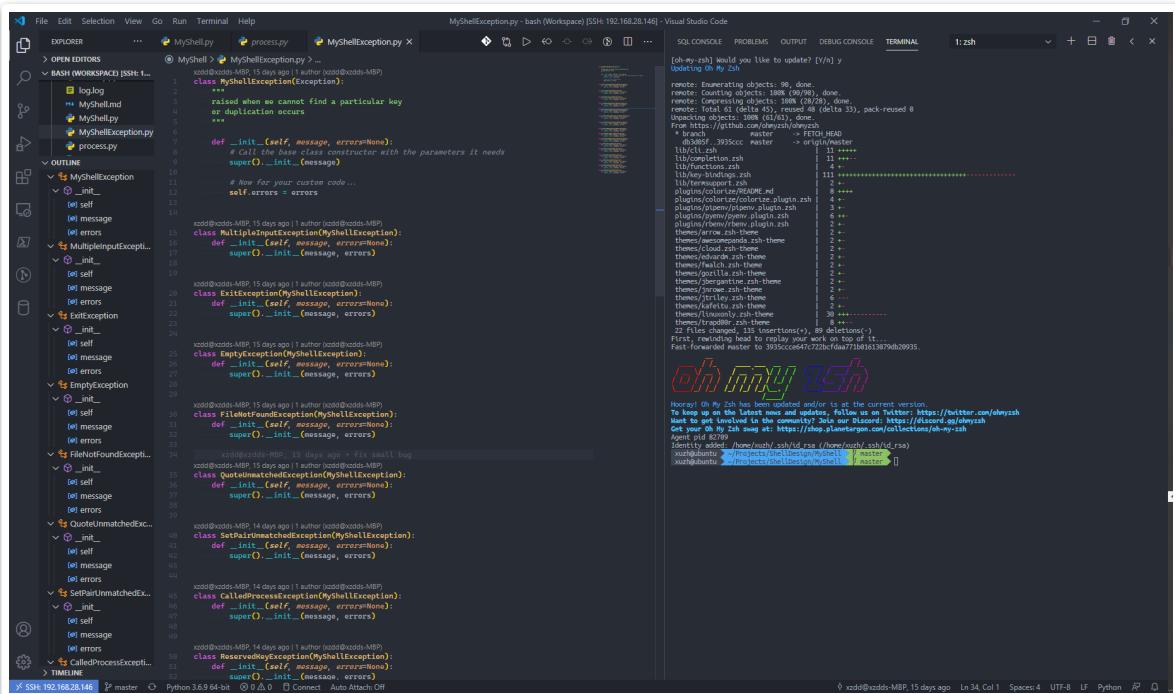
其中我们使用最多的工具是 Microsoft Visual Studio Code，让我们抛开其他内容不说，其中最吸引我们的是远程连接功能。

只要能够建立SSH到远程服务器/虚拟机，我们就可以在几乎类似本机的环境/速度上进行开发。

Ubuntu虚拟机：



宿主机器VSCode界面：



我们甚至可以进行调试：

The screenshot shows a Visual Studio Code interface with a Python debugger attached. The terminal window displays a log of events from a process named 'MyShell'. A specific entry in the log is highlighted with a red box:

```
2028-07-30 12:49:57 ubuntu _main_[84560] DEBUG This is a builtin command.
```

This indicates that the 'cd' command was executed successfully.