基于Bash的现代课程管理系统设计文档

基本信息

- 课程名称：Linux程序设计
- 实验项目名称：Shell命令
- 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：2020.07.15

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`

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：

```
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

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

可以通过如下操作创建新用户

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

.mysql.cnf就将有类似如下的内容

```
[client]
user=ShellDesigner
password=ShellDesigner
host=localhost
database=ShellDesign
```

下列是我们默认的一些设置

```
mysql_u_default="ShellDesigner"
mysql_p_default="ShellDesigner"
mysql_h_default="localhost"
mysql_d_default="ShellDesign"
mysql_f=".mysql.cnf"

# 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$mysql_prefix来开头
mysql_prefix="mysql --defaults-extra-file=$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）。

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

```
# 重定向标准输出的到文件并打印文件
 xuzh@ubuntu  ~/Projects/ShellDesign    master ●  mysql -uShellDesigner -
pShellDesigner ShellDesign -e "select * from admin;" > temp.txt; cat temp.txt
mysql: [Warning] Using a password on the command line interface can be insecure.
name    id      password_hash
root    1       53175bcc0524f37b47062fafdda28e3f8eb91d519ca0a184ca71bbebe72f969a
```

```
admin   2       fc8252c8dc55839967c58b9ad755a59b61b67c13227ddae4bd3f78a38bf394f7

# 直接执行语句，打印到标准输出
 xuzh@ubuntu  ~/Projects/ShellDesign    master ●  mysql -uShellDesigner -pShellDesigner ShellDesign -e "select * from admin;"
mysql: [Warning] Using a password on the command line interface can be insecure.
+-------+----+------------------------------------------------------------------+
| name  | id | password_hash                                                    |
+-------+----+------------------------------------------------------------------+
| root  |  1 | 53175bcc0524f37b47062fafdda28e3f8eb91d519ca0a184ca71bbebe72f969a |
| admin |  2 | fc8252c8dc55839967c58b9ad755a59b61b67c13227ddae4bd3f78a38bf394f7 |
+-------+----+------------------------------------------------------------------+

# 将标准输出重定向到Terminal标准输出
 xuzh@ubuntu  ~/Projects/ShellDesign    master ●  mysql -uShellDesigner -pShellDesigner ShellDesign -e "select * from admin;" > /dev/tty
mysql: [Warning] Using a password on the command line interface can be insecure.
+-------+----+------------------------------------------------------------------+
| name  | id | password_hash                                                    |
+-------+----+------------------------------------------------------------------+
| root  |  1 | 53175bcc0524f37b47062fafdda28e3f8eb91d519ca0a184ca71bbebe72f969a |
| admin |  2 | fc8252c8dc55839967c58b9ad755a59b61b67c13227ddae4bd3f78a38bf394f7 |
+-------+----+------------------------------------------------------------------+

# 重定向到变量并打印标准输出
 xuzh@ubuntu  ~/Projects/ShellDesign    master ●  temp=$(mysql -uShellDesigner -pShellDesigner ShellDesign -e "select * from admin;");echo "$temp"
mysql: [Warning] Using a password on the command line interface can be insecure.
name    id      password_hash
root    1       53175bcc0524f37b47062fafdda28e3f8eb91d519ca0a184ca71bbebe72f969a
admin   2       fc8252c8dc55839967c58b9ad755a59b61b67c13227ddae4bd3f78a38bf394f7
```

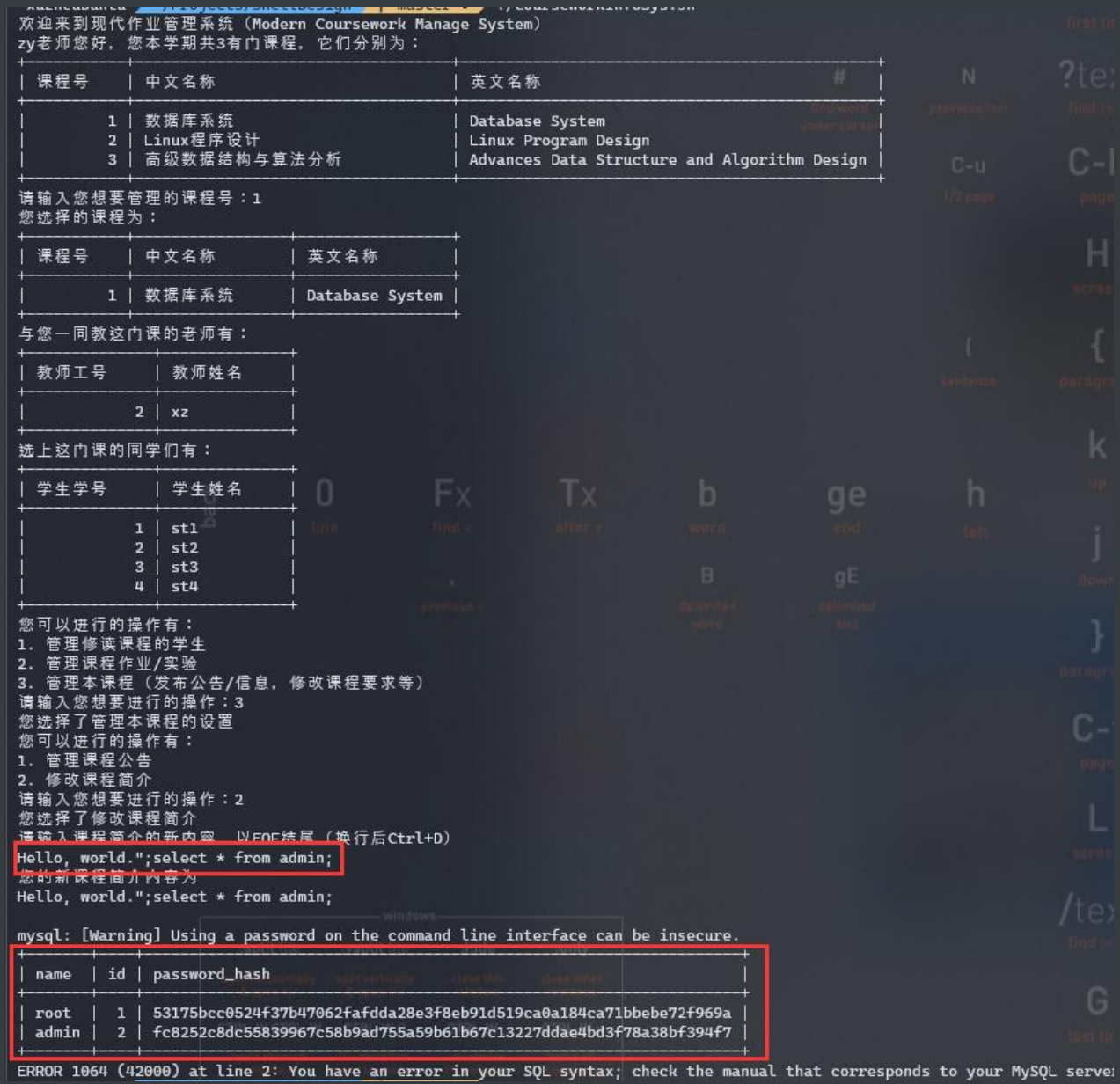
因此当我们想要让此变量获取打印的信息时，我们应直接将返回信息赋值到变量中，当我们想直接使用MySQL的格式化功能时，我们应直接使用命令将输出导入到 `/dev/tty` 。

调用举例：

```
# 通过变量$mysql_prefix中定义的格式执行MySQL查询：mysql --defaults-extra-file=$mysql_f
# 查询的内容是$query_id变量中存储的查询：select cid from take where sid=$sid
# 通过-s参数去除列名
# 并且将查询结果以数组形式存储到$cids变量中
cids=($($mysql_prefix -se "$query_id;"))

# 直接调用$mysql_prefix变量中定义的内容：mysql --defaults-extra-file=$mysql_f
# 查询的内容是$query_course变量中存储的查询：select id 课程号, name_zh 中文名称, name_en 英文名称 from course where id in ($query_id)
# 不进行输入输出的引导，直接打印到屏幕
# 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

```
function RemoveDanger() {
    danger_set=${2:-"[\"'\.\*;%]"}
    danger=$1
    safe=""
    for i in $(seq ${#danger}); do
        thechar="${danger:$i-1:1}"
        if [[ "$thechar" =~ $danger_set ]]; then
            # echo "$thechar"
            safe="$safe""\\""$thechar"
        else
            safe="$safe$thechar"
        fi
    done
    echo "$safe"
}
```

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

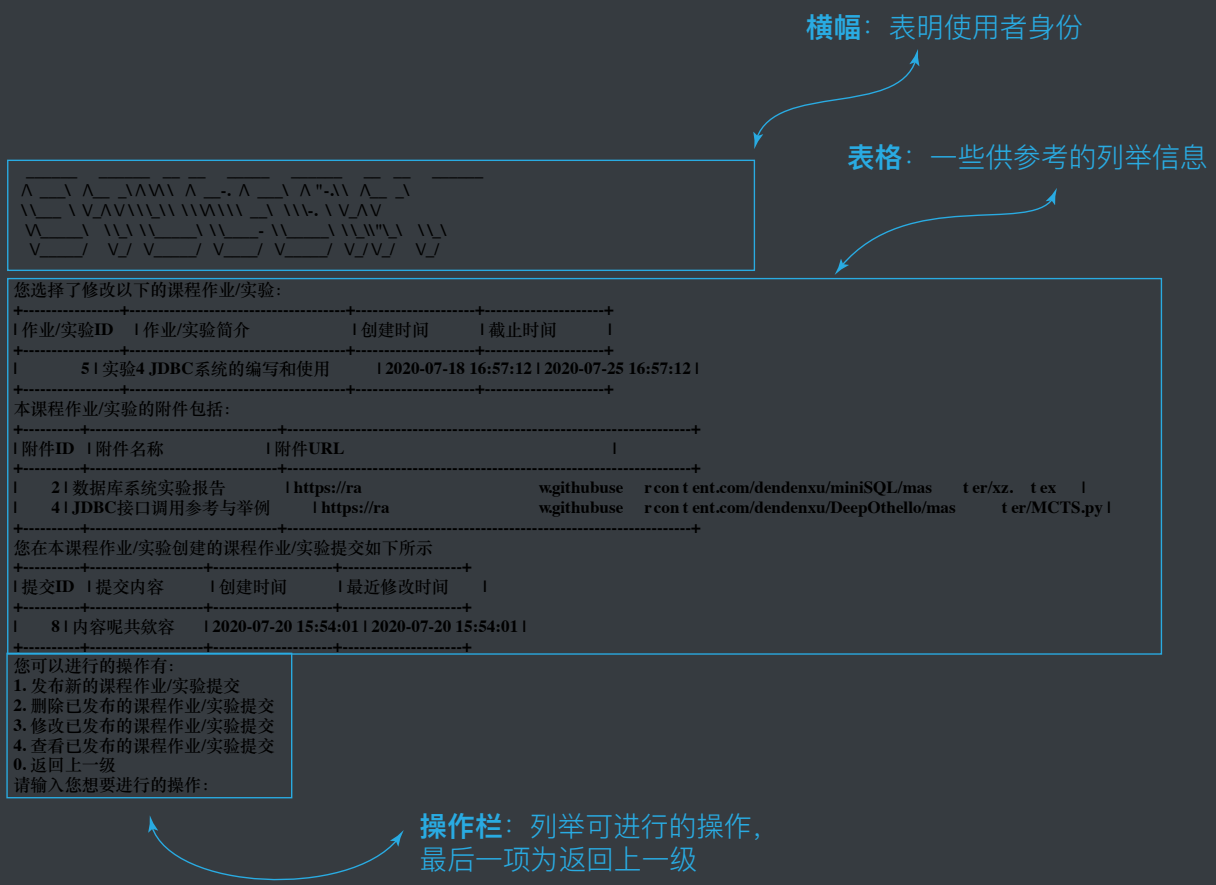
  即使数据库内容被泄露，sha256的加密也会让数据偷盗者很难猜出正确的密码

> 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



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：

```
function TeacherOPCourse() {
    while :; do # 课程操作UI主循环
        ################################################
        #                                              #
        #                   打印Banner                 #
        #                                              #
        ################################################
        PrintTeacher # 打印Banner

        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


        ################################################
        #                                              #
        #                   打印查询信息               #
        #                                              #
        ################################################
        query_tid="select tid from teach where cid=$cid"
        query_teacher="select id 教师工号, name 教师姓名, if(gender='F', \"女\", \"男\") 性别, registration_time 注册时间, title 职称, brief 简介 from teacher where id in ($query_tid)"

        echo "您选择的${target}为："

```

```
        # 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
        $mysql_prefix -e "select id 课程号, name_zh 中文名称, name_en 英文名称,
brief 课程简介 from course where id=$cid;"

        # 打印除了当前老师外一同教这门课的老师一共用户参考
        tids=($($mysql_prefix -e "$query_tid and tid <> $tid;"))
        if [ ${#tids[@]} -gt 0 ]; then
            echo "与您一同教这门课的老师有："
            $mysql_prefix -e "$query_teacher and id <> $tid;"
        else
            echo "这门${target}只有您自己在教"
        fi

        ##############################################
        #                                            #
        #                  操作栏循环                 #
        #                                            #
        ##############################################
        echo "您可以进行的操作有："
        echo "1. 管理修读${target}的学生"
        echo "2. 管理${target}作业/实验"
        echo "3. 管理本${target}信息（管理公告/简介等）"
        echo "0. ${ReturnPrev}"
        while :; do
            # 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
            # 且这里无论选择那种操作都没有直接清屏返回的必要
            read -rp "请输入您想要进行的操作：" op
            case $op in
            1)
                echo "您选择了管理修读该${target}的学生"
                TeacherManageStudent
                break
                ;;
            2)
                echo "您选择了管理本${target}的实验和作业"
                TeacherManageHomework
                break
                ;;
            3)
                echo "您选择了管理本${target}的公告/信息"
                TeacherManageCourse
                break
                ;;
            0)
                echo "您选择了${ReturnPrev}"
                return 0
                ;;
            *)
                echo "您输入的操作$op有误，请输入上面列出的操作"
```

```
                ;;
            esac
        done
    done
}
```

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

```
function PrintTeacher() {
    # TEACHER分隔符，会在老师登陆后的管理界面打印
    clear
    cat <<"EOF"
    ______   ______     ______     ______     __  __     ______     ______
   /\__  _\ /\  ___\   /\  __ \   /\  ___\   /\ \_\ \   /\  ___\   /\  == \
   \/_/\ \/ \ \  __\   \ \  __ \  \ \ \____  \ \  __ \  \ \  __\   \ \  __<
      \ \_\  \ \_____\  \ \_\ \_\  \ \_____\  \ \_\ \_\  \ \_____\  \ \_\ \_\
       \/_/   \/_____/   \/_/\/_/   \/_____/   \/_/\/_/   \/_____/   \/_/ /_/

EOF
}
```

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

```
while :; do # 屏幕主循环

# 打印表格内容
$mysql_prefix ...

# 打印可用操作内容
echo "..."
    while :; do # 操作循环
    case $choice in
        1)
        # 用户输入了正确的选项
        # 在此执行下一步的功能
        ...
        # 运行结束后重新开始主循环，刷新数据
        break
        ;;

        0)
```

```
            # 通过return命令直接返回上一级函数调用/或退出运行
            return 0
            ;;

            *)
            # 用户输入有误
            # 不调用break命令直接进行操作循环
            ;;
        done
    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

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

  使用样例：

  ```
  # 这些变量打印出来都是有颜色或重量的
  # 每次刷新页面时都要清空目标变量
  target="${Green}${Bold}课程实验/作业${NoColor}"
  # 内容未发布提示信息
  no_publication="${Red}本课程还没有已发布的${NoColor}${target}"

  echo "Target is $target"
  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${target}"
xuzh@ubuntu:~/Projects/ShellDesign$
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

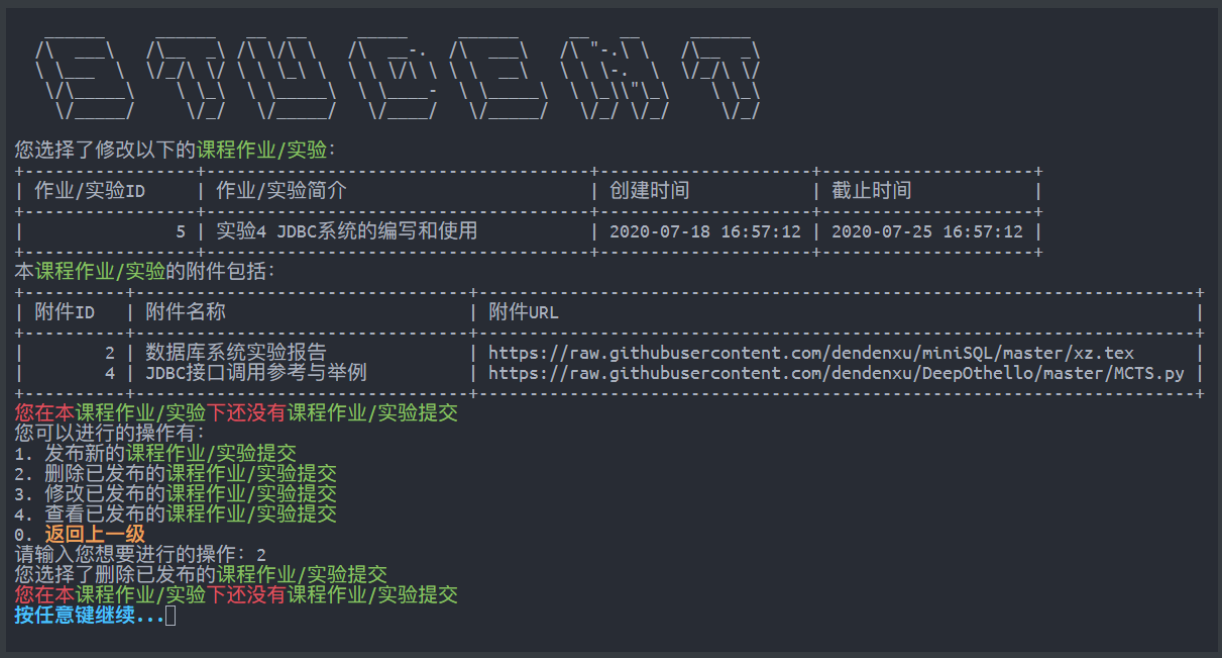
```
while :; do
    read -rp "请输入您想要管理的课程号: " cid

    # 注意到我们使用正则表达式展开数组来进行元素检查
    # 因此表达式右侧的值应用引号括起以保证完全匹配
    # 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
    # shellcheck disable=SC2076
    # 来关闭这一报错
    [[ "${cids[*]}" =~ "${cid}" ]] && break
    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"
done
```

上衣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

```
function ContinueWithKey() {
    # 按任意键继续...
    # 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
    read -n 1 -rp "${Blue}${Bold}按任意键继续...${NoColor}" -s
}
```

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

```
while :; do
    # 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
    # 所以暂时使用了变量来存储密码
    read -rp "请输入您的密码：" -s password
    echo ""
    password_hash=$(echo "$password" | sha256sum - | tr -d " -")
    echo "验证中……"
    [ "$password_hash" = "$right_hash" ] && break
    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
    echo "验证失败，请重新输入"
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

```
# 这里我们通过Bash内部计数来减少一次MySQL链接
attachment_count=0
while :; do
    # 我们根据用户回答来修改程序运行流程
    # 用户无需提前知道需要添加的附件数量
    # 他/她只需要不断输入Y并添加内容
    read -rp "请输入您是否需要为${target}添加附件（Y/n）：" need_attach
    if [[ $need_attach =~ ^[1Yy] ]]; then # 正则表达式匹配
        attachment_count+=1

        echo "您选择了添加附件"
        read -rp "请输入您想要添加的附件名称：" attach_name
        attach_name=$(RemoveDanger "$attach_name") # 可能包含危险字符
        echo "您的附件名称为：$attach_name"

        read -rp "请输入您想要添加的附件URL：" attach_url
        # 对于URL，我们使用不同的转义策略
        attach_url=$(RemoveDanger "$attach_url" "[\"'\.\*;]")
        echo "您的附件URL为：$attach_url"

        # 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
        # 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
        # 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
        query_insert_attach="insert into attachment(name, url) value (\"$attach_name\", \"$attach_url\")"
        query_insert_attach_to="insert into attach_to(aid, uid) value (last_insert_id(), $subid)"

        # 同样的，我们利用了Transaction功能
        attach_id=$($mysql_prefix -se "set autocommit=0;$query_insert_attach;select last_insert_id();$query_insert_attach_to;commit;set autocommit=1;")
```

```
29
30          echo "您刚刚添加的附件ID为: $attach_id"
31      else
32          break
33      fi
34  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      \     \___(  <_> )  |  /|  | \/\___ \\  ___/\     (  <_> )  | \/    <  /
    Y    \/ __ \|   |  \/ __ \_/ /_/  >  ___/|  | \/
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）：

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

- 用户/数据库定义部分

```
# note: we should define the default charset of the database before creating the tables without explicitly
# defining charset

drop database if exists ShellDesign;
drop user if exists ShellDesigner;
create user ShellDesigner identified by 'ShellDesigner';
create database ShellDesign;
grant all on ShellDesign.* to ShellDesigner;
alter database ShellDesign character set utf8mb4 collate utf8mb4_unicode_ci;
use ShellDesign;

drop table if exists `take`;
drop table if exists `info`;
drop table if exists `teach`;
drop table if exists `attach_to`;
drop table if exists `attachment`;
drop table if exists `submission`;
drop table if exists `homework`;
drop table if exists `content`;
drop table if exists `teacher`;
drop table if exists `student`;
drop table if exists `admin`;
drop table if exists `course`;
```

- 表建立部分

```
create table `teacher`
(
    name              varchar(100),
    id                bigint primary key auto_increment,
    brief             varchar(2000),
    gender            enum ('F', 'M') default 'F', # F for female and M for male
    registration_time datetime,
    title             varchar(500)    default 'Professor',
    password_hash     varchar(64)
);

create table `student`
(
    name          varchar(100),
    id            bigint primary key auto_increment,
    brief         varchar(2000),
```

```
    gender        enum ('F', 'M') default 'F', # F for female and M for male
    enroll_time   datetime,
    password_hash char(64)
);

create table `admin`
(
    name          varchar(100),
    id            bigint primary key auto_increment,
    password_hash char(64)
);

create table `course`
(
    name_zh  varchar(100),
    name_en  varchar(100),
    brief    varchar(2000),
    syllabus varchar(4000),
    id       bigint primary key auto_increment
);

create table `teach`
(
    tid bigint,
    cid bigint,
    foreign key (`tid`) references teacher (`id`) on delete cascade on update cascade,
    foreign key (`cid`) references course (`id`) on delete cascade on update cascade
);

create table `take`
(
    cid bigint,
    sid bigint,
    foreign key (`sid`) references student (`id`) on delete cascade on update cascade,
    foreign key (`cid`) references course (`id`) on delete cascade on update cascade
);

# this is a dummy class so that we can ensure foreign key references from attachments to both submissions and homework
create table `content`
(
    id bigint primary key auto_increment
);

create table `info`
```

```
(
    id           bigint primary key,
    content      varchar(2000),
    cid          bigint,
    release_time datetime,
    foreign key (`cid`) references course (`id`) on delete cascade on update cascade,
    foreign key (`id`) references content (`id`) on delete cascade on update cascade
);

create table `homework`
(
    id            bigint primary key auto_increment,
    cid           bigint,
    tid           bigint,
    intro         varchar(2000),
    creation_time datetime,
    end_time      datetime,
    type          enum ('H', 'E') default 'H', # H for homework and e for experiment
    foreign key (`id`) references content (`id`) on delete cascade on update cascade,
    foreign key (`tid`) references teacher (`id`) on delete cascade on update cascade,
    foreign key (`cid`) references course (`id`) on delete cascade on update cascade
);

create table `submission`
(
    id                      bigint primary key auto_increment,
    sid                     bigint,
    hid                     bigint,
    submission_text         varchar(2000),
    creation_time           datetime,
    latest_modification_time datetime,
    foreign key (`id`) references content (`id`) on delete cascade on update cascade,
    foreign key (`sid`) references student (`id`) on delete cascade on update cascade,
    foreign key (`hid`) references homework (`id`) on delete cascade on update cascade
);

create table `attachment`
(
    id    bigint primary key auto_increment,
    name  varchar(100),
```

```
    url   varchar(800),
    brief varchar(2000)
);

create table `attach_to`
(
    aid bigint,
    uid bigint,
    foreign key (`aid`) references attachment (`id`) on delete cascade on update cascade,
    foreign key (`uid`) references content (`id`) on delete cascade on update cascade
);
```

- Dummy内容插入部分

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

insert into `teacher`(id, name, password_hash, registration_time)
values (1, 'zy', '49aabdaa1b0f6c3506f54521ef81fe5b5fe835d268f1f86e1021a342b59d43bc', now()), # password is zy
       (2, 'xz', 'b44f7d6b5283a44ee5f2bd98f84087a04810092122d75e8fbf8ad85f8f2981f1', now()); # password is xz

insert into `admin`(id, name, password_hash)
values (1, 'root', '53175bcc0524f37b47062fafdda28e3f8eb91d519ca0a184ca71bbebe72f969a'), # password is root
       (2, 'admin', 'fc8252c8dc55839967c58b9ad755a59b61b67c13227ddae4bd3f78a38bf394f7'); # password is admin

insert into `student`(id, name, password_hash, enroll_time)
values (1, 'st1', '2238ead9c048f351712c34d22b41f6eec218ea9a9e03e48fad829986b0dafc11', now()), # password is same as name
       (2, 'st2', '5e61d026a7889d9fc72e17f1b25f4d6d48bfe17046fea845aa8c5651ec89c333', now()),
```

```
        (3, 'st3',
'bbb977f8e93feb5dbd79e0688b822115b5acf774dd8a1fe6964e03d6b9579384', now()),
        (4, 'st4',
'6133396ebcd382b137088d2ea91d60637744e404b4376e4635b45784b718db72', now()),
        (5, 'st5',
'd691a62aa63f1be970582902d0ff78df29899f09c5dd540b1447cdd051dcfc8d', now()),
        (6, 'st6',
'a7a287ffc9cb27131b9dc54199ba96cef87e753968bc620d714af212ef0f7a8c', now()),
        (7, 'st7',
'73d0daf13c6159a1fbdeb37b6972325b6e29c312371a0f3d427bd35c0c87b928', now()),
        (8, 'st8',
'4ce70fc1eef7303879a2ef33996db2f85058ae06e8590521267ae8d46ec59793', now());

insert into `teach`(cid, tid)
values (1, 1),
       (1, 2),
       (2, 1),
       (3, 1),
       (4, 2),
       (5, 2);

insert into `take`(cid, sid)
values (1, 1),
       (1, 2),
       (1, 3),
       (1, 4),
       (2, 3),
       (2, 4),
       (2, 5),
       (2, 6),
       (3, 7),
       (3, 8),
       (4, 1),
       (4, 3),
       (4, 5),
       (5, 2),
       (5, 4),
       (5, 6),
       (5, 8),
       (6, 1),
       (6, 7),
       (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);
```

初始化模块

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：

- 颜色变量，用以打印有色UI

```
function DefineColor() {
    # 我们使用tput命令来定义颜色信息
```

```
    # 各类颜色常数，通过echo调用可以改变Shell的输出样式
    # 例如echo "${Red}Hello${NoColor}, world."会打印红色的Hello和原色的World
    # 上述例子会展开成echo "$(tput setaf 1)Hello$(tput sgr0), world."
    # ! consider more about this colorization
    Red=$(tput setaf 1)
    Green=$(tput setaf 2)
    Yellow=$(tput setaf 3)
    Blue=$(tput setaf 4)
    Magenta=$(tput setaf 5)
    Cyan=$(tput setaf 6)
    Bold=$(tput bold)
    NoColor=$(tput sgr0)
    ReturnPrev="${Yellow}${Bold}返回上一级${NoColor}"
}
```

- 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

```
function DefineMySQL() {
    # 下列是我们默认的一些设置
    mysql_u_default="ShellDesigner"
    mysql_p_default="ShellDesigner"
    mysql_h_default="localhost"
    mysql_d_default="ShellDesign"
    mysql_f=".mysql.cnf"

    # 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
    if [ ! -f "$mysql_f" ]; then
        echo "Automatically generating configuration file..." >&2
        echo "[client]" >$mysql_f
        echo "user=$mysql_u_default" >>$mysql_f
        echo "password=$mysql_p_default" >>$mysql_f
        echo "host=$mysql_h_default" >>$mysql_f
        echo "database=$mysql_d_default" >>$mysql_f
    fi

    # 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$mysql_prefix来开头
    mysql_prefix="mysql --defaults-extra-file=$mysql_f"
}

```

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

```
# 初始界面登陆逻辑
function LoginInUI() {
    while :; do
        PrintBanner # 打印一个好看的小Banner: CourseworkManger

        # 获取用户的身份/因为我们使用了有可能会重复的ID
        # 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
        # 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
        while :; do
            read -rp "请输入您的身份（T/S/A）或输入0退出系统: " identity
            case $identity in
            [Tt])
                identity="teacher"
                break
                ;;
            [Ss])
                identity="student"
                break
                ;;
            [Aa])
                identity="admin"
                break
                ;;
            0)
                echo "Bye"
                return 0
                ;;
            *) echo "请输入T, S, A或0" ;;
            esac
        done

        # 我们会在密码判断前进行账号检测
        while :; do
            read -rp "请输入您的登陆账号: " user_id
            echo "检查中..."
            sleep 0.1s # 防止暴力登录攻击，100ms的惩罚时间

            # * 防止SQL注入攻击，转义危险字符，详见StudentManageSubmission逻辑
            user_id=$(RemoveDanger "$user_id")

            # * MySQL调用方式详见StudentUI逻辑
            query_all_hash="select id, name, password_hash from $identity"
            query_right_hash="select password_hash from ($query_all_hash) all_hash where id=\"$user_id\""
            right_hash=$($mysql_prefix -se "$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

        `输入管理员姓名，输入/确认密码`
    2. 删除管理员账户

        `输入要删除的管理员账号`
    3. 修改管理员账户

        `输入要修改的管理员账号`

        `输入新的管理员姓名，输入/确认新的密码`
    4. 返回上一级
2. 管理教师账户
    1. 添加教师账户

        `输入教师姓名，性别，简介，职称等，输入/确认密码`
    2. 删除教师账户

        `输入要删除的教师账号`
    3. 修改教师账户

        `输入要修改的教师账号`

        `输入新的教师姓名，性别，简介，职称等，输入/确认新的密码`
    4. 返回上一级

3. 管理学生账户
    1. 添加学生账户

        `输入学生姓名，性别，简介等，输入/确认密码`
    2. 删除学生账户

        `输入要删除的学生账号`
    3. 修改学生账户

        `输入要修改的学生账号`

        `输入新的学生姓名，性别，简介等，输入/确认新的密码`
    4. 返回上一级
4. 管理课程列表
    1. 添加课程

        `输入课程的中文、英文名称，添加课程简介等`
    2. 删除课程

        `输入要删除的课程号`
    3. 修改课程

        `输入要修改的课程号`

        `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

```
function RemoveDanger() {
    danger_set=${2:-"[\"'\.\*;%]"}
    danger=$1
    safe=""
    for i in $(seq ${#danger}); do
        thechar="${danger:$i-1:1}"
        if [[ "$thechar" =~ $danger_set ]]; then
            # echo "$thechar"
```

```
            safe="$safe""\\""$thechar"
        else
            safe="$safe$thechar"
        fi
    done
    echo "$safe"
}
```

- 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）

```
function PrintAttachment() {
    # 用于打印附件信息的小函数，可以提高代码可读性
    # 这个函数认为：
    # 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
    # 2. $target是目标内容的字符串描述，例如"课程作业/实验"
    # 3. $mysql_prefix可以正确执行MySQL命令，$query_attachment可以正确打印相关附件
    if [ "$attachment_count" -gt 0 ]; then
        echo "本${target}的附件包括："
        $mysql_prefix -e "$query_attachment;"
    else
        # 我们是用红色显示来让用户快速定位这一提示
        echo "${Red}本${target}${Red}还没有附件${NoColor}"
    fi
}
```

- 打印各类 ASCII ART

  1. Teacher

```
function PrintTeacher() {
    # TEACHER分隔符，会在老师登陆后的管理界面打印
    clear
    cat <<"EOF"
     ______   ______     ______     ______     __  __     ______     ______

    /\__  _\ /\  ___\   /\  __ \   /\  ___\   /\ \_\ \   /\  ___\   /\  == \

    \/_/\ \/ \ \  __\   \ \  __ \  \ \ \____  \ \  __ \  \ \  __\   \ \  __<

       \ \_\  \ \_____\  \ \_\ \_\  \ \_____\  \ \_\ \_\  \ \_____\  \ \_\ \_\

        \/_/   \/_____/   \/_/\/_/   \/_____/   \/_/\/_/   \/_____/   \/_/ /_/

EOF
}
```

2. `Student`

```
function PrintStudent() {
    # STUDENT分隔符，会在学生登陆后的管理界面打印
    clear
    cat <<"EOF"
    ______     ______   __  __     _____     ______     __   __     ______

   /\  ___\   /\__  _\ /\ \/\ \   /\  __-.  /\  ___\   /\ "-.\ \   /\__  _\
   \ \___  \  \/_/\ \/ \ \ \_\ \  \ \ \/\ \ \ \ \__ \  \ \ \-.  \  \/_/\ \/
    \/\_____\    \ \_\  \ \_____\  \ \____-  \ \_____\  \ \_\\"\_\    \ \_\
     \/_____/     \/_/   \/_____/   \/____/   \/_____/   \/_/ \/_/     \/_/

EOF
}
```

3. `Admin`

```
function PrintAdmin() {
    # ADMIN分隔符，会在管理员登陆后的管理界面打印
    clear
    cat <<"EOF"
    ______     _____     __    __     __     __   __
   /\  __ \   /\  __-.  /\ "-./  \   /\ \   /\ "-.\ \
   \ \  __ \  \ \ \/\ \ \ \ \-./\ \  \ \ \  \ \ \-.  \
    \ \_\ \_\  \ \____-  \ \_\ \ \_\  \ \_\  \ \_\\"\_\
     \/_/\/_/   \/____/   \/_/  \/_/   \/_/   \/_/ \/_/

EOF
}
```

- **继续运行**按键模块

```
function ContinueWithKey() {
    # 按任意键继续...
    # 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
    read -n 1 -rp "${Blue}${Bold}按任意键继续...${NoColor}" -s
}
```

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

```
DefineColor
DefineMySQL
LoginInUI
```

附录

完整源码

Bash

```bash
#!/bin/bash
# CourseworkInfoSys
# Author: Xu Zhen 徐震 3180105504
# shellcheck disable=SC2076
# 这是一个现代教务管理系统，主要面向作业管理
# 我们通过编写Shell程序，调用MySQL数据库来管理作业系统
# ！ 您的MySQL版本要至少为5.7
# ！ 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会较大
# 5.6.* 版本的MySQL会在执行tables.sql中的语句时出现问题
# * 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，我们会将注释主要写在第一次遇到相关逻辑的部分
# * 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, StudentManageSubmission)

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

function DefineMySQL() {
    # 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
    # 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:
```

```
    # 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

    # ！ 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
    # ALTER USER 'user'@'hostname' IDENTIFIED BY 'newPass';
    # 可以通过如下操作创建新用户
    # create user ShellDesigner identified by 'ShellDesigner';
    # create database ShellDesign;
    # grant all on ShellDesign.* to ShellDesigner;

    # .mysql.cnf就将有类似如下的内容
    # [client]
    # user=ShellDesigner
    # password=ShellDesigner
    # host=localhost
    # database=ShellDesign

    # 下列是我们默认的一些设置
    mysql_u_default="ShellDesigner"
    mysql_p_default="ShellDesigner"
    mysql_h_default="localhost"
    mysql_d_default="ShellDesign"
    mysql_f=".mysql.cnf"

    # 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
    if [ ! -f "$mysql_f" ]; then
        echo "Automatically generating configuration file..." >&2
        echo "[client]" >$mysql_f
        echo "user=$mysql_u_default" >>$mysql_f
        echo "password=$mysql_p_default" >>$mysql_f
        echo "host=$mysql_h_default" >>$mysql_f
        echo "database=$mysql_d_default" >>$mysql_f
    fi

    # 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$mysql_prefix来开头
    mysql_prefix="mysql --defaults-extra-file=$mysql_f"
}
```

```
79
80  # 初始界面登陆逻辑
81  function LoginInUI() {
82      while :; do
83          PrintBanner # 打印一个好看的小Banner: CourseworkManger
84
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}"
```

```
        if [[ "$thechar" =~ $danger_set ]]; then
            # echo "$thechar"
            safe="$safe""\\""$thechar"
        else
            safe="$safe$thechar"
        fi
    done
    echo "$safe"
}

# 以下几个Print函数都是用于打印ASCII Art的
# 同时，它们通过调用clear函数来进行假GUI的页面模拟功能
# 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
# 后来我们发现通过调用clear函数可以达到模拟GUI的功能
function PrintBanner() {
    # 程序的主横幅：CourseworkManger
    # 会在初始登陆界面打印
    clear
    cat <<"EOF"
   _________                                                  __
_____
   \_   ___ \  ____  __ _________  ______ ______  _  ______________|  | __   /
  \ _____    ____ _____     ____   ___________
   /    \  \/ /  _ \|  |  \_  __ \/  ___// __ \ \/ \/ /  _ \_  __ \  |/ /  /  \
/  \\__  \  /    \\__  \   / ___\_/ __ \_  __ \
   \     \___(  <_> )  |  /|  | \/\___ \\  ___/\    (  <_> )  | \/    <  /
Y    \/ __ \|   |  \/ __ \_/ /_/  >  ___/|  | \/
    \______  /\____/|____/ |__|  /____  >\___  >\/\_/ \____/|__|  |__|_ \
\____|__  (____  /___|  (____  /\___  / \___  >__|
           \/                \/     \/      \/                      \/
  \/     \/     \/     \//_____/      \/
EOF
}

function PrintDelimiter() {
    # SECTION分隔符，会在无法确定用户身份但需要分隔的位置打印
    clear
    cat <<"EOF"
   ______     ______     ______     ______   __     ______     __   __
  /\  ___\   /\  ___\   /\  ___\   /\__  _\ /\ \   /\  __ \   /\ "-.\ \
  \ \___  \  \ \  __\   \ \ \____  \/_/\ \/ \ \ \  \ \ \/\ \  \ \ \-.  \
   \/\_____\  \ \_____\  \ \_____\    \ \_\  \ \_\  \ \_____\  \ \_\\"\_\
    \/_____/   \/_____/   \/_____/     \/_/   \/_/   \/_____/   \/_/ \/_/

EOF
}

```

```
function PrintTeacher() {
    # TEACHER分隔符，会在老师登陆后的管理界面打印
    clear
    cat <<"EOF"
   ______   ______     ______     ______     __  __     ______     ______
  /\__  _\ /\  ___\   /\  __ \   /\  ___\   /\ \_\ \   /\  ___\   /\  == \
  \/_/\ \/ \ \ \__\   \ \  __ \  \ \ \____  \ \  __ \  \ \ \__\   \ \  __<
     \ \_\  \ \_____\  \ \_\ \_\  \ \_____\  \ \_\ \_\  \ \_____\  \ \_\ \_\
      \/_/   \/_____/   \/_/\/_/   \/_____/   \/_/\/_/   \/_____/   \/_/ /_/

EOF
}

function PrintStudent() {
    # STUDENT分隔符，会在学生登陆后的管理界面打印
    clear
    cat <<"EOF"
   ______     ______   __  __     _____     ______     __   __     ______
  /\  ___\   /\__  _\ /\ \/\ \   /\  __-.  /\  ___\   /\ "-.\ \   /\__  _\
  \ \___  \  \/_/\ \/ \ \ \_\ \  \ \ \/\ \ \ \  __\   \ \ \-.  \  \/_/\ \/
   \/\_____\    \ \_\  \ \_____\  \ \____-  \ \_____\  \ \_\\"\_\    \ \_\
    \/_____/     \/_/   \/_____/   \/____/   \/_____/   \/_/ \/_/     \/_/

EOF
}

function PrintAdmin() {
    # ADMIN分隔符，会在管理员登陆后的管理界面打印
    clear
    cat <<"EOF"
   ______     _____     __    __     __     __   __
  /\  __ \   /\  __-.  /\ "-./  \   /\ \   /\ "-.\ \
  \ \  __ \  \ \ \/\ \ \ \ \-./\ \  \ \ \  \ \ \-.  \
   \ \_\ \_\  \ \____-  \ \_\ \ \_\  \ \_\  \ \_\\"\_\
    \/_/\/_/   \/____/   \/_/  \/_/   \/_/   \/_/ \/_/

EOF
}

function ContinueWithKey() {
    # 按任意键继续...
    # 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
    read -n 1 -rp "${Blue}${Bold}按任意键继续...${NoColor}" -s
}

function StudentUI() {
    # 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
    sid=${1:-"1"}
    name=${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
    Injection攻击，后面会提到如何防御）
272         # 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）。
273         query_id="select cid from take where sid=$sid"
274         query_course="select id 课程号, name_zh 中文名称, name_en 英文名称 from
    course where id in ($query_id)"
275 
276         # 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
277         # ! 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
278         # ! 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
279 
280         # * 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）
281 
282         # 重定向标准输出的到文件并打印文件
283         #  xuzh@ubuntu  ~/Projects/ShellDesign    master ●  mysql -
    uShellDesigner -pShellDesigner ShellDesign -e "select * from admin;" >
    temp.txt; cat temp.txt
284         # mysql: [Warning] Using a password on the command line interface can
    be insecure.
285         # name    id      password_hash
286         # root    1
    53175bcc0524f37b47062fafdda28e3f8eb91d519ca0a184ca71bbebe72f969a
287         # admin   2
    fc8252c8dc55839967c58b9ad755a59b61b67c13227ddae4bd3f78a38bf394f7
288 
289         # 直接执行语句，打印到标准输出
290         #  xuzh@ubuntu  ~/Projects/ShellDesign    master ●  mysql -
    uShellDesigner -pShellDesigner ShellDesign -e "select * from admin;"
291         # mysql: [Warning] Using a password on the command line interface can
    be insecure.
292         # +-------+----+------------------------------------------------------
    -----------+
293         # | name  | id | password_hash
            |
294         # +-------+----+------------------------------------------------------
    -----------+
295         # | root  |  1 |
    53175bcc0524f37b47062fafdda28e3f8eb91d519ca0a184ca71bbebe72f969a |
```

```
296         # | admin |  2 |
    fc8252c8dc55839967c58b9ad755a59b61b67c13227ddae4bd3f78a38bf394f7 |
297         # +-------+----+-------------------------------------------------------------
    -----------+
298
299         # 将标准输出重定向到Terminal标准输出
300         #  xuzh@ubuntu  ~/Projects/ShellDesign    master ●  mysql -
    uShellDesigner -pShellDesigner ShellDesign -e "select * from admin;" > /dev/tty
301         # mysql: [Warning] Using a password on the command line interface can
    be insecure.
302         # +-------+----+-------------------------------------------------------------
    -----------+
303         # | name  | id | password_hash
            |
304         # +-------+----+-------------------------------------------------------------
    -----------+
305         # | root  |  1 |
    53175bcc0524f37b47062fafdda28e3f8eb91d519ca0a184ca71bbebe72f969a |
306         # | admin |  2 |
    fc8252c8dc55839967c58b9ad755a59b61b67c13227ddae4bd3f78a38bf394f7 |
307         # +-------+----+-------------------------------------------------------------
    -----------+
308
309         # 重定向到变量并打印标准输出
310         #  xuzh@ubuntu  ~/Projects/ShellDesign    master ●  temp=$(mysql -
    uShellDesigner -pShellDesigner ShellDesign -e "select * from admin;");echo
    "$temp"
311         # mysql: [Warning] Using a password on the command line interface can
    be insecure.
312         # name    id      password_hash
313         # root    1
    53175bcc0524f37b47062fafdda28e3f8eb91d519ca0a184ca71bbebe72f969a
314         # admin   2
    fc8252c8dc55839967c58b9ad755a59b61b67c13227ddae4bd3f78a38bf394f7
315
316         # * 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
317         # * 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
318         cids=($($mysql_prefix -se "$query_id;"))
319
320         echo "$name同学您好，欢迎来到现代作业管理系统（Modern Coursework Manage
    System）"
321         if [ ${#cids[@]} -eq 0 ]; then
322             echo "$no_publication"
323         else
324             echo "您本学期共${#cids[@]}有门课程，它们分别为："
325         fi
326         echo "您可以进行的操作有："
327         echo "1. 管理课程（提交/修改/删除作业）"
328         echo "2. 查看所有的作业/实验"
```

```
329         echo "0. ${ReturnPrev}"
330         while :; do # 操作循环UI，直到获得正确的输入
331             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，因此我们使用了
348                     # shellcheck disable=SC2076
349                     # 来关闭这一报错
350                     [[ "${cids[*]}" =~ "${cid}" ]] && break
351                     echo "您输入的课程号$cid有误，请输入上表中列举出的某个课程号"
352                 done
353 
354                 # 每次调用新的函数代表我们将要进入一个新的页面，我们不想让用户在下一页面刷新时每次都重复选择某一门课程的过程
355                 # 因此我们将选择好的课程号存储到cid变量中，隐式传递到函数StudentOPCourse中
356                 StudentOPCourse
357                 break
358                 ;;
359             2)
360                 # 查看所有作业及其完成情况
361                 # 这波，这波是个SQL题，这种长长的还不能格式化的SQL Query也是让人头大
362                 # 我们调用了许多MySQL内置功能，例如UNIX_TIMESTAMP还有IF语句等，还嵌套了Linux的命令以及变量
363                 # 值得注意的是，对于双引号需要加上转移符号，防止Bash解释它们
364                 echo "您选择了查看所有的作业和实验"
365                 query_all_hw="select sub.hid 作业ID, sub.intro 作业简介, sub.creation_time 发布时间, sub.end_time 截止时间,if(unix_timestamp(sub.end_time)<$(date +%s),\"是\",\"否\") 是否截止, if(count(sub.id)>0,\"是\",\"否\") 是否完成, count(sub.id) 创建的提交数目 from (select S.sid, S.id, H.id hid, H.intro, H.creation_time, H.end_time from (select * from submission where sid=$sid) S right join homework H on S.hid=H.id where H.cid in (select cid from take where sid=$sid)) sub group by sub.hid"
366 
367                 $mysql_prefix -e "$query_all_hw;"
```

```
368
369                 # 我们打印了一些信息，让用户确认一下
370                 ContinueWithKey
371                 break
372                 ;;
373             0)
374                 echo "您选择了${ReturnPrev}"
375                 return 0
376                 ;;
377             *)
378                 echo "您输入的操作$op有误，请输入上面列出的操作"
379                 # 此时不进行Break而是继续请求用户的操作选择
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_prefix -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$hid有误，请输入上表中列举出的某
    个${target}ID"
443                 done
444                 # 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面刷新
    时每次都重复选择某一项课程作业/实验
445                 # 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
446                 StudentManageSubmission
447
448                 break
449                 ;;
450             0)
451                 echo "您选择了${ReturnPrev}"
452                 return 0
453                 ;;
454             *)
455                 echo "您输入的操作$op有误，请输入上面列出的操作"
456                 ;;
457             esac
458         done
459     done
```

```
}

function PrintAttachment() {
    # 用于打印附件信息的小函数，可以提高代码可读性
    # 这个函数认为：
    # 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
    # 2. $target是目标内容的字符串描述，例如"课程作业/实验"
    # 3. $mysql_prefix可以正确执行MySQL命令，$query_attachment可以正确打印相关附件
    if [ "$attachment_count" -gt 0 ]; then
        echo "本${target}的附件包括："
        $mysql_prefix -e "$query_attachment;"
    else
        # 我们是用红色显示来让用户快速定位这一提示
        echo "${Red}本${target}${Red}还没有附件${NoColor}"
    fi
}

function StudentManageSubmission() {
    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，退出循环后会回到上一级
        PrintStudent

        # "提交"的上一级为："课程作业/实验"
        upper="${Green}课程作业/实验${NoColor}"
        target="$upper${Green}提交${NoColor}"

        # 用红色显示的没有提交的信息，方便用户定位
        no_publication="${Red}您在本${NoColor}$upper${Red}下还没有${NoColor}${target}"

        echo "您选择了修改以下的$upper："
        query_course_homework="select id \`作业/实验ID\`, intro \`作业/实验简介\`, creation_time 创建时间, end_time 截止时间 from homework where id=$hid"
        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=$hid"
        query_count_attachment="select count(1) from attachment join attach_to on id=aid where uid=$hid"
        $mysql_prefix -e "$query_course_homework;"

        # 我们通过MySQL Query直接确定相关附件数量的值
        attachment_count=$($mysql_prefix -se "$query_count_attachment")

        # 暂时替换$target和$upper
        temp=${target}
        target=$upper
        PrintAttachment # 这里我们打印的是upper的附件，但PrintAttachment会通过$target打印名称
        target=$temp

```

```
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，并对他们进行手动转义
553                 # 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解释后很可能不再得到转义，也就给了不法分子可乘之机。
554                 full_string=$(RemoveDanger "$full_string")
555
556                 echo -e "您的${target}的简介内容为\n$full_string"
557
558                 # 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Transaction功能
559                 # 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ID值，以保证数据完整性和对应性
560                 query_insert_content="insert into content value ()"
561                 query_insert_submission="insert into submission value (last_insert_id(), $sid, $hid, \"$full_string\", now(), now())"
562
563                 # 我们可以通过;串联SQL语句来让它们在同一个MySQL Connection中执行
564                 # 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唯一有打印内容的一个（返回上次插入的信息）
565                 subid=$($mysql_prefix -se "set autocommit=0;$query_insert_content;select last_insert_id();$query_insert_submission;commit;set autocommit=1;")
566
567                 echo "您刚刚添加的${target}ID为：$subid"
568
569                 # 这里我们通过Bash内部计数来减少一次MySQL链接
570                 attachment_count=0
571                 while :; do
572                     # 我们根据用户回答来修改程序运行流程
573                     # 用户无需提前知道需要添加的附件数量
574                     # 他/她只需要不断输入Y并添加内容
575                     read -rp "请输入您是否需要为${target}添加附件（Y/n）：" need_attach
576                     if [[ $need_attach =~ ^[1Yy] ]]; then # 正则表达式匹配
577                         attachment_count+=1
578
579                         echo "您选择了添加附件"
580                         read -rp "请输入您想要添加的附件名称：" attach_name
581                         attach_name=$(RemoveDanger "$attach_name") # 可能包含危险字符
582                         echo "您的附件名称为：$attach_name"
583
584                         read -rp "请输入您想要添加的附件URL：" attach_url
585                         # 对于URL，我们使用不同的转义策略
586                         attach_url=$(RemoveDanger "$attach_url" "[\"'\.\*;]")
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的关系
590                         # 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
591                         # 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
592                         query_insert_attach="insert into attachment(name, url) value (\"$attach_name\", \"$attach_url\")"
593                         query_insert_attach_to="insert into attach_to(aid, uid) value (last_insert_id(), $subid)"
594
595                         # 同样的，我们利用了Transaction功能
596                         attach_id=$($mysql_prefix -se "set autocommit=0;$query_insert_attach;select last_insert_id();$query_insert_attach_to;commit;set autocommit=1;")
597
598                         echo "您刚刚添加的附件ID为：$attach_id"
599                     else
600                         break
601                     fi
602                 done
603
604                 # 打印一些信息，让用户得到应有的反馈
605                 echo "您刚刚对课程号为$cid的课程的ID为$hid的$upper发布了如下的${target}："
606                 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=$subid"
607                 query_attachment="select A.id 附件ID, A.name 附件名称, A.url 附件URL from attachment A join attach_to T on A.id=T.aid where T.uid=$subid"
608                 $mysql_prefix -e "$query_course_submission;"
609                 PrintAttachment
610
611                 # 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
612                 ContinueWithKey
613                 # 清空屏幕
614                 break
615                 ;;
616             2)
617                 echo "您选择了删除已发布的${target}"
618                 if [ "$end_time" -lt "$(date +%s)" ]; then
619                     echo "${Red}本作业已经截止提交${NoColor}"
620                     ContinueWithKey
621                     break
622                 fi
623                 # 若根本没有发布内容，删除就是完全无用的
624                 if [ ${#subids[@]} -eq 0 ]; then
625                     echo "$no_publication"
```

```
626                     ContinueWithKey
627                     break
628                 fi
629
630                 # 逻辑同上述的while read 循环
631                 while :; do
632                     read -rp "请输入您想要删除的${target}ID: " subid
633                     [[ "${subids[*]}" =~ "${subid}" ]] && break
634                     echo "您输入的${target}ID$subid有误，请输入上表中列举出的某个${target}ID"
635                 done
636
637                 # 我们对各类Foreign Key使用了on update cascade on delete cascade功能，就无需显式的删除其他有可能引用到相关内容的东西
638                 query_delete_content="delete from content where id=$subid"
639                 $mysql_prefix -e "$query_delete_content;"
640
641                 break
642                 ;;
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$subid有误，请输入上表中列举出的某个${target}ID"
662                 done
663
664                 echo "您选择修改的${target}为: "
665
666                 # 让用户观察自己选择修改的内容
667                 query_course_submission="select id 提交ID, submission_text 提交内容, creation_time 创建时间, latest_modification_time 最近修改时间 from submission where id=$subid"
668                 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"
```

```
669                 $mysql_prefix -e "$query_course_submission;"
670                 query_count_attachment="select count(1) from attachment join
    attach_to on id=aid where uid=$subid"
671                 attachment_count=$($mysql_prefix -se "$query_count_attachment")
672                 PrintAttachment
673 
674                 # 对于full_string的处理同上
675                 echo "请输入${target}的简介内容，以EOF结尾（换行后Ctrl+D）"
676                 full_string=""
677                 while read -r temp; do
678                     full_string+="$temp"$'\n'
679                 done
680                 full_string=$(RemoveDanger "$full_string")
681                 echo -e "您的${target}的简介内容为\n$full_string"
682 
683                 # 同上
684                 query_modify_submission="update submission set
    submission_text=\"$full_string\", latest_modification_time=now() where
    id=$subid"
685                 $mysql_prefix -e "$query_modify_submission;"
686                 echo "您刚刚修改的${target}ID为：$subid"
687                 while :; do
688                     read -rp "请输入您是否需要为${target}添加附件（Y/n）："
    need_attach
689                     if [[ $need_attach =~ ^[1Yy] ]]; then
690                         echo "您选择了添加附件"
691                         read -rp "请输入您想要添加的附件名称：" attach_name
692                         attach_name=$(RemoveDanger "$attach_name")
693                         echo "您的附件名称为：$attach_name"
694                         read -rp "请输入您想要添加的附件URL：" attach_url
695                         # 对于URL，我们使用不同的转义策略
696                         attach_url=$(RemoveDanger "$attach_url" "[\"'\.\*;]")
697                         echo "您的附件URL为：$attach_url"
698                         query_insert_attach="insert into attachment(name, url)
    value (\"$attach_name\", \"$attach_url\")"
699                         query_insert_attach_to="insert into attach_to(aid, uid)
    value (last_insert_id(), $subid)"
700                         attach_id=$($mysql_prefix -se "set
    autocommit=0;$query_insert_attach;select
    last_insert_id();$query_insert_attach_to;commit;set autocommit=1;")
701                         echo "您刚刚添加的附件ID为：$attach_id"
702                     else
703                         break
704                     fi
705                 done
706 
707                 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
    ${target}："
708                 $mysql_prefix -e "$query_course_submission;"
```

```
709
710                 attachment_count=$($mysql_prefix -se "$query_count_attachment")
711                 PrintAttachment
712                 ContinueWithKey
713                 break
714                 ;;
715             4)
716                 echo "您选择了查询已发布的${target}"
717
718                 # 几乎相同的逻辑
719                 if [ ${#subids[@]} -eq 0 ]; then
720                     echo "$no_publication"
721                     ContinueWithKey
722                     break
723                 fi
724
725                 while :; do
726                     read -rp "请输入您想要查询的作业/实验提交ID: " subid
727                     [[ "${subids[*]}" =~ "${subid}" ]] && break
728                     echo "您输入的提交ID$subid有误，请输入上表中列举出的某个提交ID"
729                 done
730
731                 echo "您选择查询的提交为: "
732                 query_course_submission="select id 提交ID, submission_text 提交内
    容, creation_time 创建时间, latest_modification_time 最近修改时间 from submission
    where id=$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                 ;;
```

```
            esac
        done
    done
}

function TeacherUI() {
    # 同样的，我们使用默认值以方便调试
    tid=${1:-"1"}
    name=${2:-"zy"}

    while :; do      # 页面主循环
        PrintTeacher # 打印TEACHER BANNER提示用户

        no_publication="${Red}您本学期没有课程${NoColor}"
        query_id="select cid from teach where tid=$tid"
        query_course="select id 课程号, name_zh 中文名称, name_en 英文名称 from course where id in ($query_id)"

        # 所有课程数目
        cids=($($mysql_prefix -se "$query_id;"))

        echo "$name老师您好，欢迎来到现代作业管理系统（Modern Coursework Manage System）"
        if [ ${#cids[@]} -eq 0 ]; then
            echo "您本学期没有课程"
        else
            echo "您本学期共${#cids[@]}有门课程，它们分别为："
            $mysql_prefix -e "$query_course;"
        fi

        # 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
        echo "您可以进行的操作有："
        echo "1. 管理课程"
        echo "0. ${ReturnPrev}"
        while :; do # 错误输入的处理循环，这里只能输入0或者1
            read -rp "请输入您想要进行的操作：" op
            case $op in
            1)
                echo "您选择了管理课程"
                if [ ${#cids[@]} -eq 0 ]; then
                    echo "您本学期没有课程"
                    ContinueWithKey
                    break
                fi
                while :; do
                    read -rp "请输入您想要管理的课程号：" cid
                    [[ "${cids[*]}" =~ "${cid}" ]] && break
                    echo "您输入的课程号$cid有误，请输入上表中列举出的某个课程号"
                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认
823         query_tid="select tid from teach where cid=$cid"
824         query_teacher="select id 教师工号, name 教师姓名, if(gender='F', \"女\",
    \"男\") 性别, registration_time 注册时间, title 职称, brief 简介 from teacher where
    id in ($query_tid)"
825
826         echo "您选择的${target}为："
827
828         # 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
829         $mysql_prefix -e "select id 课程号, name_zh 中文名称, name_en 英文名称,
    brief 课程简介 from course where id=$cid;"
830
831         # 打印除了当前老师外一同教这门课的老师一共用户参考
832         tids=($($mysql_prefix -e "$query_tid and tid <> $tid;"))
833         if [ ${#tids[@]} -gt 0 ]; then
834             echo "与您一同教这门课的老师有："
835             $mysql_prefix -e "$query_teacher and id <> $tid;"
836         else
837             echo "这门${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
```

```
            # 且这里无论选择那种操作都没有直接清屏返回的必要
            read -rp "请输入您想要进行的操作: " op
            case $op in
            1)
                echo "您选择了管理修读该${target}的学生"
                TeacherManageStudent
                break
                ;;
            2)
                echo "您选择了管理本${target}的实验和作业"
                TeacherManageHomework
                break
                ;;
            3)
                echo "您选择了管理本${target}的公告/信息"
                TeacherManageCourse
                break
                ;;
            0)
                echo "您选择了${ReturnPrev}"
                return 0
                ;;
            *)
                echo "您输入的操作$op有误，请输入上面列出的操作"
                ;;
            esac
        done
    done
}

function TeacherManageCourse() {
    # 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
    while :; do
        PrintTeacher

        target1="${Green}课程公告${NoColor}"
        target2="${Green}课程简介${NoColor}"
        echo "您可以进行的操作有: "
        echo "1. 管理课程$target1"
        echo "2. 修改课程$target2"
        echo "0. ${ReturnPrev}"
        while :; do
            read -rp "请输入您想要进行的操作: " op
            case $op in
            1)
                echo "您选择了管理$target1"
                TeacherManageCourseInfo
                break
                ;;
```

```
            2)
                echo "您选择了修改$target2"
                TeacherManageCourseBrief
                break
                ;;
            0)
                echo "您选择了${ReturnPrev}"
                return 0
                ;;
            *)
                echo "您输入的操作$op有误，请输入上面列出的操作"
                ;;
            esac
        done
    done
}

function TeacherManageCourseBrief() {
    # 管理课程简介内容
    # 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提示信息
    target="${Green}课程简介${NoColor}"
    echo "${target}的原内容为"
    $mysql_prefix -e "select brief 课程简介 from course where id=$cid"

    # 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
    echo "请输入${target}的新内容，以EOF结尾（换行后Ctrl+D）"
    # 这种读取方式在前面已经介绍过
    full_string=""
    while read -r temp; do
        full_string+="$temp"$'\n'
    done
    full_string=$(RemoveDanger "$full_string")

    echo -e "您的新${target}内容为\n$full_string"
    query_brief_update="update course set brief = \"$full_string\" where id=$cid"
    # 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。
    # we can easily perfomr SQL injection if the string is not carefully treated
    # update course set brief = "Hello, world.";select * from admin;\" where id=$cid
    $mysql_prefix -e "$query_brief_update;"

    # 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看一眼
    ContinueWithKey
}

```

```
function TeacherManageCourseInfo() {
    # 管理公告的逻辑和学生管理作业提交的逻辑十分类似
    # 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
    while :; do
        PrintTeacher

        target="${Green}课程公告${NoColor}"
        no_publication="${Red}本课程没有已发布的${NoColor}${target}"

        query_iid="select id from info where cid=$cid"
        query_info="select id 公告ID, release_time 公告发布时间, content 公告内容 from info where cid=$cid"

        iids=($($mysql_prefix -e "$query_iid;"))

        # 惯例：打印一下已有的公告来供用户参考
        if [ ${#iids[@]} -gt 0 ]; then
            echo "本课程已有的${target}如下图所示"
            $mysql_prefix -e "$query_info;"
        else
            echo "$no_publication"
        fi

        echo "您可以进行的操作有："
        echo "1. 发布新的${target}"
        echo "2. 删除已发布的${target}"
        echo "3. 修改已发布的${target}"
        echo "4. 查询已发布的${target}"
        echo "0. ${ReturnPrev}"

        while :; do
            read -rp "请输入您想要进行的操作：" op
            case $op in
            1)
                echo "您选择了发布新的${target}"

                # todo：这一段操作可以考虑封装成函数
                echo "请输入${target}的新内容，以EOF结尾（换行后Ctrl+D）"
                full_string=""
                while read -r temp; do
                    full_string+="$temp"$'\n'
                done
                full_string=$(RemoveDanger "$full_string")
                echo -e "您的新${target}内容为\n$full_string"

                # 这里的逻辑在上面也有体现
                # 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
                query_insert_content="insert into content value ()"
```

```
                query_insert_info="insert into info(id, content, cid,
release_time) value (last_insert_id(), \"$full_string\", $cid, now())"

                iid=$($mysql_prefix -se "set
autocommit=0;$query_insert_content;select
last_insert_id();$query_insert_info;commit;set autocommit=1;")

                echo "您刚刚发布的${target}ID为：$iid"
                attachment_count=0
                while :; do
                    read -rp "请输入您是否需要为${target}添加附件（Y/n）："
need_attach
                    if [[ $need_attach =~ ^[1Yy] ]]; then
                        attachment_count+=1
                        echo "您选择了添加附件"
                        read -rp "请输入您想要添加的附件名称：" attach_name
                        attach_name=$(RemoveDanger "$attach_name")
                        echo "您的附件名称为：$attach_name"
                        read -rp "请输入您想要添加的附件URL：" attach_url
                        # 对于URL，我们使用不同的转义策略
                        attach_url=$(RemoveDanger "$attach_url" "[\"'\.\*;]")
                        echo "您的附件URL为：$attach_url"
                        query_insert_attach="insert into attachment(name, url)
value (\"$attach_name\", \"$attach_url\")"
                        query_insert_attach_to="insert into attach_to(aid, uid)
value (last_insert_id(), $iid)"
                        attach_id=$($mysql_prefix -se "set
autocommit=0;$query_insert_attach;select
last_insert_id();$query_insert_attach_to;commit;set autocommit=1;")
                        echo "您刚刚添加的附件ID为：$attach_id"
                    else
                        break
                    fi
                done

                echo "您刚刚对课程号为$cid的课程发布了如下的${target}："
                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=$iid;"
                query_attachment="select A.id 附件ID, A.name 附件名称, A.url 附件
URL from attachment A join attach_to T on A.id=T.aid where T.uid=$iid"
                $mysql_prefix -e "$query_course_info;"

                PrintAttachment
                ContinueWithKey

                break
                ;;
            2)
```

```
1025                # 完全类似的逻辑
1026                echo "您选择了删除已发布的${target}"
1027                if [ ${#iids[@]} -eq 0 ]; then
1028                    echo "$no_publication"
1029                    ContinueWithKey
1030                    break
1031                fi
1032                while :; do
1033                    read -rp "请输入您想要删除的${target}ID: " iid
1034                    [[ "${iids[*]}" =~ "${iid}" ]] && break
1035                    echo "您输入的${target}ID$iid有误，请输入上表中列举出的某
    个${target}ID"
1036                done
1037                query_delete_content="delete from content where id=$iid"
1038                $mysql_prefix -e "$query_delete_content;"
1039                break
1040                ;;
1041            3)
1042                # 同上
1043                echo "您选择了修改已发布的${target}"
1044                if [ ${#iids[@]} -eq 0 ]; then
1045                    echo "$no_publication"
1046                    ContinueWithKey
1047                    break
1048                fi
1049                while :; do
1050                    read -rp "请输入您想要修改的${target}ID: " iid
1051                    [[ "${iids[*]}" =~ "${iid}" ]] && break
1052                    echo "您输入的${target}ID$iid有误，请输入上表中列举出的某
    个${target}ID"
1053                done
1054
1055                # 修改内容前让用户有确认的机会
1056                echo "您选择了修改以下的${target}: "
1057                query_course_info="select I.id 公告ID, I.content 公告内容,
    I.release_time 公告发布时间 from (info I join course C on I.cid=C.id) where
    I.id=$iid;"
1058                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=$iid"
1059                query_count_attachment="select count(1) from attachment join
    attach_to on id=aid where uid=$iid"
1060                $mysql_prefix -e "$query_course_info;"
1061                attachment_count=$($mysql_prefix -se "$query_count_attachment")
1062                PrintAttachment
1063
1064                # 同上
1065                echo "请输入${target}的新内容，以EOF结尾（换行后Ctrl+D）"
1066                full_string=""
1067                while read -r temp; do
```

```
1068                    full_string+="$temp"$'\n'
1069                done
1070                full_string=$(RemoveDanger "$full_string")
1071                echo -e "您的新${target}内容为\n$full_string"
1072
1073                query_insert_info="update info set content=\"$full_string\"
    where id=$iid"
1074
1075                $mysql_prefix -se "$query_insert_info;"
1076
1077                echo "您刚刚修改的${target}ID为：$iid"
1078
1079                # 同上
1080                while :; do
1081                    read -rp "请输入您是否需要为${target}添加新的附件（Y/n）："
    need_attach
1082                    if [[ $need_attach =~ ^[1Yy] ]]; then
1083                        echo "您选择了添加附件"
1084                        read -rp "请输入您想要添加的附件名称：" attach_name
1085                        attach_name=$(RemoveDanger "$attach_name")
1086                        echo "您的附件名称为：$attach_name"
1087                        read -rp "请输入您想要添加的附件URL：" attach_url
1088                        # 对于URL，我们使用不同的转义策略
1089                        attach_url=$(RemoveDanger "$attach_url" "[\"'\.\*;]")
1090                        echo "您的附件URL为：$attach_url"
1091                        query_insert_attach="insert into attachment(name, url)
    value (\"$attach_name\", \"$attach_url\")"
1092                        query_insert_attach_to="insert into attach_to(aid, uid)
    value (last_insert_id(), $iid)"
1093                        attach_id=$($mysql_prefix -se "set
    autocommit=0;$query_insert_attach;select
    last_insert_id();$query_insert_attach_to;commit;set autocommit=1;")
1094                        echo "您刚刚添加的附件ID为：$attach_id"
1095                    else
1096                        break
1097                    fi
1098                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}"
```

```
                if [ ${#iids[@]} -eq 0 ]; then
                    echo "$no_publication"
                    ContinueWithKey
                    break
                fi
                while :; do
                    read -rp "请输入您想要查询的${target}ID: " iid
                    [[ "${iids[*]}" =~ "${iid}" ]] && break
                    echo "您输入的${target}ID$iid有误，请输入上表中列举出的某个${target}ID"
                done
                echo "您选择了查询以下的${target}: "
                query_course_info="select I.id 公告ID, I.content 公告内容, I.release_time 公告发布时间 from (info I join course C on I.cid=C.id) where I.id=$iid;"
                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=$iid"
                query_count_attachment="select count(1) from attachment join attach_to on id=aid where uid=$iid"
                $mysql_prefix -e "$query_course_info;"
                attachment_count=$($mysql_prefix -se "$query_count_attachment")
                PrintAttachment
                ContinueWithKey
                break
                ;;
            0)
                echo "您选择了${ReturnPrev}"
                return 0
                ;;
            *)
                echo "您输入的操作$op有误，请输入上面列出的操作"
                ;;
            esac
        done
    done
}

function TeacherManageStudent() {
    # 老师管理学生账户
    # 添加/删除到课程等
    while :; do
        PrintTeacher # 打印Banner
        target="${Green}学生${NoColor}"
        no_publication="${Red}没有${NoColor}$target${Red}选上这门课${NoColor}"

        # 查询已经选上课的同学们
        query_sid="select sid from take where cid=$cid"
        query_student="select id 学生学号, name 学生姓名 from student where id in ($query_sid)"
```

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

```
1199                echo "您选择了从课程名单中移除$target"
1200                if [ ${#sids[@]} -eq 0 ]; then
1201                    echo "$no_publication"
1202                    ContinueWithKey
1203                    break
1204                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 id=$sid"
1212                $mysql_prefix -e "$query_student_info;"
1213
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 [[ $need_delete_student_course =~ ^[1Yy] ]]; then
1218
1219                    # * 值得注意的是，虽然我们已经使用了on delete cascade功能来方便MySQL中的外键管理，但此时的删除并不是删除整个学生账户
1220                    # * 而是调整账户使其不再在课程内
1221                    # 这里如果处理不当会出现数据不一致的错误
1222                    # todo: 想出一种可以从设计上避免数据不一致的数据库定义范式
1223                    # ! 但这里有一个Paradox: 若一个学生被移出课程名单，是否需要清除其已有的提交呢?
1224                    # * 我们现在选择的是移除，也就是说若学生曾经提交过作业，但老师将其从名单中移除了，后又添加回来了，他的所有提交都会消失
1225                    query_delete_student_course="delete from take where sid=$sid and cid=$cid"
1226                    query_delete_student_attach_to="delete from attach_to where uid in (select id from submission where sid=$sid and hid in (select id from homework where cid=$cid))"
1227                    query_delete_student_submission="delete from submission where sid=$sid and hid in (select id from homework where cid=$cid)"
1228
1229                    # 我们使用了commit来尽量保证操作的完整性
1230                    $mysql_prefix -e "set autocommit=0;$query_delete_student_course;$query_delete_student_attach_to;$query_delete_student_submission;commit;set autocommit=1;"
1231                fi
1232                break
1233                ;;
1234            0)
1235                echo "您选择了${ReturnPrev}"
1236                return 0
1237                ;;
```

```
1238                *)
1239                    echo "您输入的操作$op有误，请输入上面列出的操作"
1240                    ;;
1241                esac
1242            done
1243        done
1244    }
1245
1246    function TeacherManageHomework() {
1247        # * 老师管理作业的逻辑和学生管理作业提交的逻辑十分相似
1248        # 详细注释内容请参考：StudentManageSubmission函数
1249        while :; do
1250            PrintTeacher
1251
1252            target="${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=$cid"
1256            query_hw="select id 作业ID, intro 作业简介, creation_time 作业发布时间,
    end_time 作业截止时间 from homework where cid=$cid"
1257
1258            hids=($($mysql_prefix -e "$query_hid;"))
1259            if [ ${#hids[@]} -gt 0 ]; then
1260                echo "本课程已有的${target}如下图所示"
1261                $mysql_prefix -e "$query_hw;"
1262            else
1263                echo "$no_publication"
1264            fi
1265
1266            echo "您可以进行的操作有："
1267            echo "1. 发布新的${target}"
1268            echo "2. 删除已发布的${target}"
1269            echo "3. 修改已发布的${target}"
1270            echo "4. 查看已发布的${target}"
1271            echo "0. ${ReturnPrev}"
1272
1273            while :; do
1274                read -rp "请输入您想要进行的操作：" op
1275                case $op in
1276                1)
1277                    echo "您选择了发布新的${target}"
1278                    echo "请输入课程实验的简介内容，以EOF结尾（换行后Ctrl+D）"
1279                    full_string=""
1280                    while read -r temp; do
1281                        full_string+="$temp"$'\n'
1282                    done
1283
1284                    full_string=$(RemoveDanger "$full_string")
1285
```

```
1286                 echo -e "您的${target}的简介内容为\n$full_string"
1287
1288                 read -rp "请输入您想要创建的是作业还是实验（H/E）：" h_or_e
1289                 [[ $h_or_e =~ ^[Hh] ]] && h_or_e="H" || h_or_e="E"
1290
1291                 while :; do
1292                     read -rp "请输入作业的持续时间（天）：" days
1293                     [[ $days =~ ^[0-9]+$ ]] && break || echo "请输入整数"
1294                 done
1295
1296                 # 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
1297                 # 一天有86400秒
1298                 # 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
    要转义
1299                 # 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expr
    需要转义和$
1300                 query_insert_content="insert into content value ()"
1301                 query_insert_hw="insert into homework(id,
    cid,tid,intro,creation_time,end_time) value
    (last_insert_id(),$cid,$tid,\"$full_string\",now(),from_unixtime($(($(date +%s)
    + days * 86400))))"
1302
1303                 hid=$($mysql_prefix -se "set
    autocommit=0;$query_insert_content;select
    last_insert_id();$query_insert_hw;commit;set autocommit=1;")
1304
1305                 echo "您刚刚添加的${target}ID为：$hid"
1306                 attachment_count=0
1307                 while :; do
1308                     read -rp "请输入您是否需要为${target}添加附件（Y/n）："
    need_attach
1309                     if [[ $need_attach =~ ^[1Yy] ]]; then
1310                         attachment_count+=1
1311                         echo "您选择了添加附件"
1312                         read -rp "请输入您想要添加的附件名称：" attach_name
1313                         attach_name=$(RemoveDanger "$attach_name")
1314                         echo "您的附件名称为：$attach_name"
1315                         read -rp "请输入您想要添加的附件URL：" attach_url
1316                         # 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）
1317                         attach_url=$(RemoveDanger "$attach_url" "[\"'\.\*;]")
1318                         echo "您的附件URL为：$attach_url"
1319                         query_insert_attach="insert into attachment(name, url)
    value (\"$attach_name\", \"$attach_url\")"
1320                         query_insert_attach_to="insert into attach_to(aid, uid)
    value (last_insert_id(), $hid)"
1321                         attach_id=$($mysql_prefix -se "set
    autocommit=0;$query_insert_attach;select
    last_insert_id();$query_insert_attach_to;commit;set autocommit=1;")
1322                         echo "您刚刚添加的附件ID为：$attach_id"
```

```
                    else
                        break
                    fi
                done

                # 打印全部信息
                echo "您刚刚对课程号为$cid的课程发布了如下的${target}: "
                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 H.id=$hid"
                query_attachment="select A.id 附件ID, A.name 附件名称, A.url 附件URL from attachment A join attach_to T on A.id=T.aid where T.uid=$hid"
                $mysql_prefix -e "$query_course_homework;"
                PrintAttachment

                ContinueWithKey

                break
                ;;
            2)
                # 同上
                echo "您选择了删除已发布的${target}"
                if [ ${#hids[@]} -eq 0 ]; then
                    echo "$no_publication"
                    ContinueWithKey
                    break
                fi
                while :; do
                    read -rp "请输入您想要删除的${target}ID: " hid
                    [[ "${hids[*]}" =~ "${hid}" ]] && break
                    echo "您输入的${target}ID$hid有误，请输入上表中列举出的某个${target}ID"
                done
                query_delete_content="delete from content where id=$hid"
                $mysql_prefix -e "$query_delete_content;"

                break
                ;;
            3)
                # 同上
                echo "您选择了修改已发布的${target}"
                if [ ${#hids[@]} -eq 0 ]; then
                    echo "$no_publication"
                    ContinueWithKey
                    break
                fi
                while :; do
                    read -rp "请输入您想要修改的${target}ID: " hid
                    [[ "${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${target}ID"
1369                done
1370
1371                echo "您选择了修改以下的${target}："
1372                query_course_homework="select id \`作业/实验ID\`, intro \`作业/实验简介\`, creation_time 创建时间, end_time 截止时间 from homework where id=$hid"
1373                query_attachment="select A.id 附件ID, A.name 附件名称, A.url 附件URL from attachment A join attach_to T on A.id=T.aid where T.uid=$hid"
1374                query_count_attachment="select count(1) from attachment join attach_to on id=aid where uid=$hid"
1375                $mysql_prefix -e "$query_course_homework;"
1376                attachment_count=$($mysql_prefix -se "$query_count_attachment")
1377                PrintAttachment
1378
1379                echo "请输入${target}简介的新内容，以EOF结尾（换行后Ctrl+D）"
1380                full_string=""
1381                while read -r temp; do
1382                    full_string+="$temp"$'\n'
1383                done
1384
1385                full_string=$(RemoveDanger "$full_string")
1386
1387                echo -e "您的新${target}简介内容为\n$full_string"
1388
1389                query_insert_homework="update homework set intro=\"$full_string\" where id=$hid"
1390
1391                $mysql_prefix -se "$query_insert_homework;"
1392
1393                while :; do
1394                    read -rp "请输入作业的持续时间（天）：" days
1395                    [[ $days =~ ^[0-9]+$ ]] && break || echo "请输入整数"
1396                done
1397
1398                query_get_start_time="select unix_timestamp(creation_time) from homework where id=$hid"
1399                creation_time=$($mysql_prefix -se "$query_get_start_time;")
1400                query_update_end_time="update homework set end_time=from_unixtime($((creation_time + days * 86400))) where id=$hid"
1401                $mysql_prefix -e "$query_update_end_time;"
1402
1403                echo "您刚刚修改的课程${target}ID为：$hid"
1404                attachment_count=$($mysql_prefix -se "$query_count_attachment")
1405                while :; do
1406                    read -rp "请输入您是否需要为课程${target}添加新的附件（Y/n）：" need_attach
1407                    if [[ $need_attach =~ ^[1Yy] ]]; then
1408                        echo "您选择了添加附件"
```

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

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

```
1493                # subid: submission_id: 提交ID
1494                query_subids="select id from submission where sid=$sid and
    hid=$hid"
1495                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 ($query_subids)"
1496
1497                subids=($($mysql_prefix -se "$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（一个模块宕机其他模块也能暂时正常使用）
1593        # todo: 重构重复性高的内容到一个大函数中
1594        # 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语言的目的本身就不是如此吧
1595        while :; do
1596            PrintAdmin
1597            target="${Green}课程${NoColor}"
1598            no_course="${Red}系统中没有课程${NoColor}"
1599            query_cid="select id from course"
1600            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"
1601            cids=($($mysql_prefix -se "$query_cid;"))
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现
1624                    # 获取并确认中文内容
1625                    read -rp "请输入您想要的新${target}中文名称：" c_name_zh
1626                    c_name_zh=$(RemoveDanger "$c_name_zh")
```

```
1627                 echo "您的${target}名称为：$c_name_zh"
1628
1629                 # 获取并确认英文内容
1630                 read -rp "请输入您想要的新${target}英文名称：" c_name_en
1631                 c_name_en=$(RemoveDanger "$c_name_en")
1632                 echo "您的${target}名称为：$c_name_en"
1633
1634                 # 获取并确认简介内容
1635                 echo "请输入${target}的简介内容，以EOF结尾（换行后Ctrl+D）"
1636                 full_string=""
1637                 while read -r temp; do
1638                     full_string+="$temp"$'\n'
1639                 done
1640                 full_string=$(RemoveDanger "$full_string")
1641                 echo -e "您的${target}的简介内容为\n$full_string"
1642
1643                 query_insert_course="insert into course(name_zh, name_en,
    brief) value (\"$c_name_zh\",\"$c_name_en\",\"$full_string\")"
1644                 query_last_insert_id="select last_insert_id()"
1645
1646                 cid=$($mysql_prefix -se
    "$query_insert_course;$query_last_insert_id;")
1647                 echo "添加成功……"
1648                 query_course_new="$query_course where id=$cid"
1649                 echo "您新添加的$target如下所示"
1650                 $mysql_prefix -e "$query_course_new;"
1651                 ContinueWithKey
1652                 break
1653                 ;;
1654             2)
1655                 echo "您选择了删除${target}"
1656                 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$cid有误，请输入上表中列举出的某
    个${target}ID"
1665                 done
1666                 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
    都会丢失：${NoColor}："
1667                 query_course_info="$query_course where id=$cid"
1668                 $mysql_prefix -e "$query_course_info;"
1669
1670                 # 类似的，给老师一个确认的机会
1671                 read -rp "是否要移除（Y/n）：" need_delete
```

```
1672                 if [[ $need_delete =~ ^[1Yy] ]]; then
1673                     query_delete_course="delete from course where id=$cid"
1674                     $mysql_prefix -e "$query_delete_course"
1675                 fi
1676                 break
1677                 ;;
1678             3)
1679                 echo "您选择了修改${target}"
1680                 if [ ${#cids[@]} -eq 0 ]; then
1681                     echo "$no_course"
1682                     ContinueWithKey
1683                     break
1684                 fi
1685                 while :; do
1686                     read -rp "请输入您想要修改的${target}ID: " cid
1687                     [[ "${cids[*]}" =~ "${cid}" ]] && break
1688                     echo "您输入的${target}ID$cid有误，请输入上表中列举出的某
      个${target}ID"
1689                 done
1690
1691                 read -rp "请输入您想要的新${target}中文名称: " c_name_zh
1692                 c_name_zh=$(RemoveDanger "$c_name_zh")
1693                 echo "您的${target}名称为: $c_name_zh"
1694
1695                 read -rp "请输入您想要的新${target}英文名称: " c_name_en
1696                 c_name_en=$(RemoveDanger "$c_name_en")
1697                 echo "您的${target}名称为: $c_name_en"
1698
1699                 echo "请输入${target}的简介内容，以EOF结尾（换行后Ctrl+D）"
1700                 full_string=""
1701                 while read -r temp; do
1702                     full_string+="$temp"$'\n'
1703                 done
1704                 full_string=$(RemoveDanger "$full_string")
1705                 echo -e "您的${target}的简介内容为\n$full_string"
1706
1707                 query_change_course="update course set name_zh=\"$c_name_zh\",
      name_en=\"$c_name_en\", brief=\"$full_string\" where id=$cid"
1708
1709                 $mysql_prefix -e "$query_change_course;"
1710
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 "0. ${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
    ($query_tid)"
1778                    query_all_teachers="$query_teacher_basic where id not in
    ($query_tid)"
1779                    all_tids=($($mysql_prefix -se "$query_all_tids;"))
1780                    echo "没有被导入该课程但是已经注册的$target有："
1781                    $mysql_prefix -e "$query_all_teachers;"
1782                    while :; do
1783                        read -rp "请输入您想要添加的${target}ID：" tid
1784                        [[ "${all_tids[*]}" =~ "${tid}" ]] && break
1785                        echo "您输入的ID$tid有误，请输入上表中列举出的某个$target的ID"
1786                    done
1787
1788                    # 打印下老师选择的同学
1789                    echo "您选择了将下列$target添加进课程名单："
1790                    query_teacher_info="$query_teacher_basic where id=$tid"
1791                    $mysql_prefix -e "$query_teacher_info;"
1792
1793                    # 给老师一个确认是否添加的机会
1794                    read -rp "是否要添加（Y/n）：" need_insert_teacher_course
1795                    if [[ $need_insert_teacher_course =~ ^[1Yy] ]]; then
1796                        query_insert_teacher_course="insert into teach(tid, cid)
    value ($tid, $cid)"
1797                        $mysql_prefix -e "$query_insert_teacher_course;"
1798                    fi
1799                    break
1800                    ;;
1801                2)
1802                    echo "您选择了从课程名单中移除$target"
1803                    if [ ${#tids[@]} -eq 0 ]; then
1804                        echo "$no_teacher"
1805                        ContinueWithKey
1806                        break
1807                    fi
1808                    while :; do
1809                        read -rp "请输入您想要移除的${target}ID：" tid
1810                        [[ "${all_tids[*]}" =~ "${tid}" ]] && break
```

```
1811                    echo "您输入的ID$tid有误，请输入上表中列举出的某个$target的ID"
1812                done
1813                echo "您选择了将下列$target从课程名单中移除"
1814                query_teacher_info="$query_teacher_basic where id=$tid"
1815                $mysql_prefix -e "$query_teacher_info;"
1816
1817                # 类似的，给老师一个确认的机会
1818                read -rp "是否要移除（Y/n）：" need_delete_teacher_course
1819
1820                if [[ $need_delete_teacher_course =~ ^[1Yy] ]]; then
1821                    # * 我们现在选择的是移除，也就是说若老师曾经发布过作业，但管理员将其从名单中移除了，后又添加回来了，他的所有作业都会消失
1822                    # * 类似的，对应作业的提交也会消失，对应提交的附件关系也会消失
1823                    # todo: 实现附件attachment和带附件内容content的多对多管理能力
1824                    query_delete_teacher_course="delete from teach where tid=$tid and cid=$cid"
1825                    query_delete_teacher_hw="delete from homework where tid=$tid"
1826
1827                    # 我们使用了commit来尽量保证操作的完整性
1828                    $mysql_prefix -e "set autocommit=0;$query_delete_teacher_course;$query_delete_teacher_hw;commit;set autocommit=1;"
1829                fi
1830                break
1831                ;;
1832            0)
1833                echo "您选择了${ReturnPrev}"
1834                return 0
1835                ;;
1836            *)
1837                echo "您输入的操作$op有误，请输入上面列出的操作"
1838                ;;
1839            esac
1840        done
1841    done
1842}
1843
1844function AdminManageStudent() {
1845    # 和课程管理逻辑十分相似
1846    while :; do
1847        PrintAdmin
1848        target="${Green}学生账户${NoColor}"
1849        no_student="${Red}系统中没有学生${NoColor}"
1850        query_sid="select id from student"
1851        query_student="select id 学生学号, name 学生姓名, if(gender='F', \"女\", \"男\") 性别, enroll_time 录取时间, brief 简介 from student"
1852        sids=($($mysql_prefix -se "$query_sid;"))
1853
```

```
1854            if [ ${#sids[@]} -gt 0 ]; then
1855                echo "系统中已有的${target}如下所示："
1856                $mysql_prefix -e "$query_student;"
1857            else
1858                echo "$no_student"
1859            fi
1860            echo "您可以进行的操作有："
1861            echo "1. 添加${target}"
1862            echo "2. 删除${target}"
1863            echo "3. 修改${target}"
1864            echo "0. ${ReturnPrev}"
1865            while :; do
1866                read -rp "请输入您想要进行的操作：" op
1867                case $op in
1868                1)
1869                    echo "您选择了添加${target}"
1870
1871                    read -rp "请输入您想要的新${target}名称：" s_name
1872                    s_name=$(RemoveDanger "$s_name")
1873                    echo "您的${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 "$password" | sha256sum - | tr -d
    " -")
1894                        read -rp "请确认您的密码：" -s password
1895                        echo ""
1896                        password_hash_fi=$(echo "$password" | sha256sum - | tr -d "
    -")
1897                        if [ "$password_hash_ori" = "$password_hash_fi" ]; then
1898                            echo "密码设置成功...."
1899                            break
1900                        fi
```

```
1901                     echo "您两次输入的密码不一致，请重新输入"
1902                 done
1903                 query_insert_student="insert into student(name, brief, gender,
     password_hash, enroll_time) value (\"$s_name\",\"$full_string\",\"$s_gender\",
     \"$password_hash\", now())"
1904                 query_last_insert_id="select last_insert_id()"
1905
1906                 sid=$($mysql_prefix -se
     "$query_insert_student;$query_last_insert_id;")
1907
1908                 query_student_new="$query_student where id=$sid"
1909                 echo "添加成功……"
1910
1911                 echo "您新添加的$target如下所示"
1912                 $mysql_prefix -e "$query_student_new;"
1913                 ContinueWithKey
1914                 break
1915                 ;;
1916             2)
1917                 echo "您选择了删除${target}"
1918                 if [ ${#sids[@]} -eq 0 ]; then
1919                     echo "$no_student"
1920                     ContinueWithKey
1921                     break
1922                 fi
1923                 while :; do
1924                     read -rp "请输入您想要删除的${target}ID: " sid
1925                     [[ "${sids[*]}" =~ "${sid}" ]] && break
1926                     echo "您输入的${target}ID$sid有误，请输入上表中列举出的某
     个${target}ID"
1927                 done
1928                 echo "您选择了将下列$target从系统中中移除，${Red}注意：其所有相关信息都
     会丢失：${NoColor}"
1929                 query_student_info="$query_student where id=$sid"
1930                 $mysql_prefix -e "$query_student_info;"
1931
1932                 # 类似的，给老师一个确认的机会
1933                 read -rp "是否要移除（Y/n）：" need_delete
1934                 if [[ $need_delete =~ ^[1Yy] ]]; then
1935                     query_delete_student="delete from student where id=$sid"
1936                     $mysql_prefix -e "$query_delete_student"
1937                 fi
1938                 break
1939                 ;;
1940             3)
1941                 echo "您选择了修改${target}"
1942                 if [ ${#sids[@]} -eq 0 ]; then
1943                     echo "$no_student"
1944                     ContinueWithKey
```

```
1945                    break
1946                fi
1947                while :; do
1948                    read -rp "请输入您想要修改的${target}ID: " sid
1949                    [[ "${sids[*]}" =~ "${sid}" ]] && break
1950                    echo "您输入的${target}ID$sid有误，请输入上表中列举出的某
    个${target}ID"
1951                done
1952
1953                read -rp "请输入您想要的新${target}名称: " s_name
1954                s_name=$(RemoveDanger "$s_name")
1955                echo "您的${target}名称为: $s_name"
1956
1957                read -rp "请输入新的${target}对应的性别 (M/F) : " s_gender
1958                [[ $s_gender =~ ^[Mm] ]] && s_gender="M" || s_gender="F"
1959                echo "您选择的性别为: $s_gender"
1960
1961                echo "请输入${target}的简介内容，以EOF结尾 (换行后Ctrl+D) "
1962                full_string=""
1963                while read -r temp; do
1964                    full_string+="$temp"$'\n'
1965                done
1966
1967                full_string=$(RemoveDanger "$full_string")
1968
1969                echo -e "您的${target}的简介内容为\n$full_string"
1970
1971                # 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
1972                # 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
1973                # 即使数据库遭到泄露，用户也无法直接获得密码
1974                read -rp "是否要修改${target}密码 (Y/n) : " need_change_pw
1975                if [[ $need_change_pw =~ ^[1Yy] ]]; then
1976                    while :; do
1977                        read -rp "请输入您的密码: " -s password
1978                        echo ""
1979                        password_hash_ori=$(echo "$password" | sha256sum - | tr
    -d " -")
1980                        read -rp "请确认您的密码: " -s password
1981                        echo ""
1982                        password_hash_fi=$(echo "$password" | sha256sum - | tr
    -d " -")
1983                        if [ "$password_hash_ori" = "$password_hash_fi" ]; then
1984                            query_change_pw="update student set
    password_hash=\"$password_hash_ori\" where id=$sid"
1985                            $mysql_prefix -e "$query_change_pw;"
1986                            echo "密码修改成功..."
1987                            break
1988                        fi
1989                        echo "您两次输入的密码不一致，请重新输入"
```

```
                    done
                fi

                query_change_student="update student set name=\"$s_name\", brief=\"$full_string\", gender=\"$s_gender\" where id=$sid"

                $mysql_prefix -e "$query_change_student;"

                echo "修改成功..."
                query_student_new="$query_student where id=$sid"
                echo "您新添加的$target如下所示"
                $mysql_prefix -e "$query_student_new;"
                ContinueWithKey

                break
                ;;
            0)
                echo "您选择了${ReturnPrev}"
                return 0
                ;;
            *)
                echo "您输入的操作$op有误，请输入上面列出的操作"
                ;;
            esac
        done
    done
}

function AdminManageTeacher() {
    # 与管理同学的逻辑十分相似
    while :; do
        PrintAdmin
        target="${Green}教师账户${NoColor}"
        no_teacher="${Red}系统中没有教师${NoColor}"

        query_tid="select id from teacher"
        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"
        tids=($($mysql_prefix -se "$query_tid;"))

        if [ ${#tids[@]} -gt 0 ]; then
            echo "系统中已有的${target}如下所示："
            $mysql_prefix -e "$query_teacher;"
        else
            echo "$no_teacher"
        fi
        echo "您可以进行的操作有："
        echo "1. 添加${target}"
        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                        read -rp "请确认您的密码: " -s password
2072                        echo ""
2073                        password_hash_fi=$(echo "$password" | sha256sum - | tr -d "
    -")
2074                        if [ "$password_hash_ori" = "$password_hash_fi" ]; then
2075                            echo "密码设置成功...."
2076                            break
2077                        fi
2078                        echo "您两次输入的密码不一致, 请重新输入"
2079                    done
2080                    query_insert_teacher="insert into teacher(name, brief, title,
    gender, password_hash, registration_time) value
    (\"$t_name\",\"$full_string\",\"$t_title\",\"$t_gender\", \"$password_hash\",
    now())"
```

```
2081                query_last_insert_id="select last_insert_id()"
2082
2083                tid=$($mysql_prefix -se
    "$query_insert_teacher;$query_last_insert_id;")
2084                echo "添加成功……"
2085
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$tid有误，请输入上表中列举出的某
    个${target}ID"
2103                done
2104                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
    都会丢失：${NoColor}："
2105                query_teacher_info="$query_teacher where id=$tid"
2106                $mysql_prefix -e "$query_teacher_info;"
2107
2108                # 类似的，给老师一个确认的机会
2109                read -rp "是否要移除（Y/n）：" need_delete
2110                if [[ $need_delete =~ ^[1Yy] ]]; then
2111                    query_delete_teacher="delete from teacher where id=$tid"
2112                    $mysql_prefix -e "$query_delete_teacher"
2113                fi
2114                break
2115                ;;
2116            3)
2117                echo "您选择了修改${target}"
2118                if [ ${#tids[@]} -eq 0 ]; then
2119                    echo "$no_teacher"
2120                    ContinueWithKey
2121                    break
2122                fi
2123                while :; do
2124                    read -rp "请输入您想要修改的${target}ID: " tid
2125                    [[ "${tids[*]}" =~ "${tid}" ]] && break
```

```
2126                     echo "您输入的${target}ID$tid有误，请输入上表中列举出的某
    个${target}ID"
2127                 done
2128
2129                 read -rp "请输入您想要的新${target}名称： " t_name
2130                 t_name=$(RemoveDanger "$t_name")
2131                 echo "您的${target}名称为： $t_name"
2132
2133                 # 为了表示对女性的尊重，我们将无法判断为男性的任何情况都设定为女性教师/学生
2134                 read -rp "请输入新的${target}对应的性别（M/F）： " t_gender
2135                 [[ $t_gender =~ ^[Mm] ]] && t_gender="M" || t_gender="F"
2136                 echo "您选择的性别为： $t_gender"
2137
2138                 read -rp "请输入您的${target}的职称： " t_title
2139                 t_title=$(RemoveDanger "$t_title")
2140                 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 "$password" | sha256sum - | tr
    -d " -")
2158                         read -rp "请确认您的密码： " -s password
2159                         echo ""
2160                         password_hash_fi=$(echo "$password" | sha256sum - | tr
    -d " -")
2161                         if [ "$password_hash_ori" = "$password_hash_fi" ]; then
2162                             query_change_pw="update teacher set
    password_hash=\"$password_hash_ori\" where id=$tid"
2163                             $mysql_prefix -e "$query_change_pw;"
2164                             echo "密码修改成功..."
2165                             break
2166                         fi
2167                         echo "您两次输入的密码不一致，请重新输入"
2168                     done
2169                 fi
2170
```

```
2171                query_change_teacher="update teacher set name=\"$t_name\",
    brief=\"$full_string\", gender=\"$t_gender\", title=\"$t_title\" where id=$tid"
2172
2173                $mysql_prefix -e "$query_change_teacher;"
2174
2175                echo "教师账户修改成功..."
2176                query_teacher_new="$query_teacher where id=$tid"
2177                echo "您新添加的$target如下所示"
2178                $mysql_prefix -e "$query_teacher_new;"
2179                ContinueWithKey
2180
2181                break
2182                ;;
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
2195 function AdminManageAdmin() {
2196     while :; do
2197         PrintAdmin
2198         target="${Green}管理员账户${NoColor}"
2199         no_admin="${Red}系统中没有管理员${NoColor}"
2200
2201         query_admid="select id from admin"
2202         query_admin="select id 管理员账号, name 管理员姓名 from admin"
2203         admids=($($mysql_prefix -se "$query_admid;"))
2204
2205         if [ ${#admids[@]} -gt 0 ]; then
2206             echo "系统中已有的${target}如下所示："
2207             $mysql_prefix -e "$query_admin;"
2208         else
2209             echo "$no_admin"
2210         fi
2211         echo "您可以进行的操作有："
2212         echo "1. 添加${target}"
2213         echo "2. 删除${target}"
2214         echo "3. 修改${target}"
2215         echo "0. ${ReturnPrev}"
2216         while :; do
2217             read -rp "请输入您想要进行的操作：" op
2218             case $op in
```

```
2219            1)
2220                echo "您选择了添加${target}"
2221
2222                read -rp "请输入您想要的新${target}名称: " admin_name
2223                admin_name=$(RemoveDanger "$admin_name")
2224                echo "您的${target}名称为: $admin_name"
2225
2226                while :; do
2227                    read -rp "请输入您的密码: " -s password
2228                    echo ""
2229                    password_hash_ori=$(echo "$password" | sha256sum - | tr -d
    " -")
2230                    read -rp "请确认您的密码: " -s password
2231                    echo ""
2232                    password_hash_fi=$(echo "$password" | sha256sum - | tr -d "
    -")
2233                    if [ "$password_hash_ori" = "$password_hash_fi" ]; then
2234                        echo "密码设置成功...."
2235                        break
2236                    fi
2237                    echo "您两次输入的密码不一致, 请重新输入"
2238                done
2239                query_insert_admin="insert into admin(name, password_hash)
    value (\"$admin_name\", \"$password_hash\")"
2240                query_last_insert_id="select last_insert_id()"
2241                admid=$($mysql_prefix -se
    "$query_insert_admin;$query_last_insert_id;")
2242
2243                echo "添加成功……"
2244
2245                query_admin_new="select id 管理员账号, name 管理员姓名 from admin
    where id=$admid"
2246                echo "您新添加的$target如下所示"
2247                $mysql_prefix -e "$query_admin_new;"
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 [ ${#admids[@]} -eq 0 ]; then
2255                    echo "$no_admin"
2256                    ContinueWithKey
2257                    break
2258                fi
2259                while :; do
2260                    read -rp "请输入您想要删除的${target}ID: " admid
2261                    [[ "${admids[*]}" =~ "${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${target}ID"
2263                 done
2264                 echo "您选择了将下列$target从课程名单中移除："
2265                 query_admin_info="select id 管理员账号, name 管理员姓名 from admin where id=$admid"
2266                 $mysql_prefix -e "$query_admin_info;"
2267 
2268                 # 类似的，给老师一个确认的机会
2269                 read -rp "是否要移除（Y/n）：" need_delete
2270                 if [[ $need_delete =~ ^[1Yy] ]]; then
2271                     query_delete_admin="delete from admin where id=$admid"
2272                     $mysql_prefix -e "$query_delete_admin"
2273                 fi
2274                 break
2275                 ;;
2276             3)
2277                 echo "您选择了修改${target}"
2278                 if [ ${#admids[@]} -eq 0 ]; then
2279                     echo "$no_admin"
2280                     ContinueWithKey
2281                     break
2282                 fi
2283                 while :; do
2284                     read -rp "请输入您想要修改的${target}ID：" admid
2285                     [[ "${admids[*]}" =~ "${admid}" ]] && break
2286                     echo "您输入的${target}ID$admid有误，请输入上表中列举出的某个${target}ID"
2287                 done
2288 
2289                 read -rp "请输入您想要的新${target}名称：" admin_name
2290                 admin_name=$(RemoveDanger "$admin_name")
2291                 echo "您的${target}名称为：$admin_name"
2292 
2293                 query_change_admin_name="update admin set name=\"$admin_name\" where id=$admid"
2294                 $mysql_prefix -e "$query_change_admin_name;"
2295 
2296                 read -rp "是否要修改${target}密码（Y/n）：" need_change_pw
2297                 if [[ $need_change_pw =~ ^[1Yy] ]]; then
2298                     while :; do
2299                         read -rp "请输入您的密码：" -s password
2300                         echo ""
2301                         password_hash_ori=$(echo "$password" | sha256sum - | tr -d " -")
2302                         read -rp "请确认您的密码：" -s password
2303                         echo ""
2304                         password_hash_fi=$(echo "$password" | sha256sum - | tr -d " -")
```

```
2305                        if [ "$password_hash_ori" = "$password_hash_fi" ]; then
2306                            query_change_pw="update admin set
    password_hash=\"$password_hash_ori\" where id=$admid"
2307                            $mysql_prefix -e "$query_change_pw;"
2308                            echo "密码修改成功...."
2309                            break
2310                        fi
2311                        echo "您两次输入的密码不一致，请重新输入"
2312                    done
2313                fi
2314                echo "管理员账户修改成功...."
2315                query_admin_new="select id 管理员账号, name 管理员姓名 from admin
    where id=$admid"
2316                echo "您新修改的$target如下所示"
2317                $mysql_prefix -e "$query_admin_new;"
2318                ContinueWithKey
2319
2320                break
2321                ;;
2322            0)
2323                echo "您选择了${ReturnPrev}"
2324                return 0
2325                ;;
2326            *)
2327                echo "您输入的操作$op有误，请输入上面列出的操作"
2328                ;;
2329            esac
2330        done
2331    done
2332 }
2333
2334 # ！ 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）
2335 # https://www.shellscript.sh/tips/change-running-script/
2336 # 主程序从这里开始，上面定义的都是可供调用的函数
2337 # 请查看对程序的注释来理解本软件的工作原理
2338 DefineColor
2339 DefineMySQL
2340 LoginInUI
```

MySQL

```
1  # note: we should define the default charset of the database before creating the
   tables without explicitly
2  # defining charset
3
4  drop database if exists ShellDesign;
5  drop user if exists ShellDesigner;
```

```
create user ShellDesigner identified by 'ShellDesigner';
create database ShellDesign;
grant all on ShellDesign.* to ShellDesigner;
alter database ShellDesign character set utf8mb4 collate utf8mb4_unicode_ci;
use ShellDesign;

drop table if exists `take`;
drop table if exists `info`;
drop table if exists `teach`;
drop table if exists `attach_to`;
drop table if exists `attachment`;
drop table if exists `submission`;
drop table if exists `homework`;
drop table if exists `content`;
drop table if exists `teacher`;
drop table if exists `student`;
drop table if exists `admin`;
drop table if exists `course`;

create table `teacher`
(
    name              varchar(100),
    id                bigint primary key auto_increment,
    brief             varchar(2000),
    gender            enum ('F', 'M') default 'F', # F for female and M for male
    registration_time datetime,
    title             varchar(500)    default 'Professor',
    password_hash     varchar(64)
);

create table `student`
(
    name          varchar(100),
    id            bigint primary key auto_increment,
    brief         varchar(2000),
    gender        enum ('F', 'M') default 'F', # F for female and M for male
    enroll_time   datetime,
    password_hash char(64)
);

create table `admin`
(
    name          varchar(100),
    id            bigint primary key auto_increment,
    password_hash char(64)
);

create table `course`
(
```

```
    name_zh  varchar(100),
    name_en  varchar(100),
    brief    varchar(2000),
    syllabus varchar(4000),
    id       bigint primary key auto_increment
);

create table `teach`
(
    tid bigint,
    cid bigint,
    foreign key (`tid`) references teacher (`id`) on delete cascade on update cascade,
    foreign key (`cid`) references course (`id`) on delete cascade on update cascade
);

create table `take`
(
    cid bigint,
    sid bigint,
    foreign key (`sid`) references student (`id`) on delete cascade on update cascade,
    foreign key (`cid`) references course (`id`) on delete cascade on update cascade
);

# this is a dummy class so that we can ensure foreign key references from attachments to both submissions and homework
create table `content`
(
    id bigint primary key auto_increment
);

create table `info`
(
    id           bigint primary key,
    content      varchar(2000),
    cid          bigint,
    release_time datetime,
    foreign key (`cid`) references course (`id`) on delete cascade on update cascade,
    foreign key (`id`) references content (`id`) on delete cascade on update cascade
);

create table `homework`
(
    id            bigint primary key auto_increment,
```

```
    cid           bigint,
    tid           bigint,
    intro         varchar(2000),
    creation_time datetime,
    end_time      datetime,
    type          enum ('H', 'E') default 'H', # H for homework and e for experiment
    foreign key (`id`) references content (`id`) on delete cascade on update cascade,
    foreign key (`tid`) references teacher (`id`) on delete cascade on update cascade,
    foreign key (`cid`) references course (`id`) on delete cascade on update cascade
);

create table `submission`
(
    id                      bigint primary key auto_increment,
    sid                     bigint,
    hid                     bigint,
    submission_text         varchar(2000),
    creation_time           datetime,
    latest_modification_time datetime,
    foreign key (`id`) references content (`id`) on delete cascade on update cascade,
    foreign key (`sid`) references student (`id`) on delete cascade on update cascade,
    foreign key (`hid`) references homework (`id`) on delete cascade on update cascade
);

create table `attachment`
(
    id    bigint primary key auto_increment,
    name  varchar(100),
    url   varchar(800),
    brief varchar(2000)
);

create table `attach_to`
(
    aid bigint,
    uid bigint,
    foreign key (`aid`) references attachment (`id`) on delete cascade on update cascade,
    foreign key (`uid`) references content (`id`) on delete cascade on update cascade
);

```

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

insert into `teacher`(id, name, password_hash, registration_time)
values (1, 'zy', '49aabdaa1b0f6c3506f54521ef81fe5b5fe835d268f1f86e1021a342b59d43bc', now()), # password is zy
       (2, 'xz', 'b44f7d6b5283a44ee5f2bd98f84087a04810092122d75e8fbf8ad85f8f2981f1', now()); # password is xz

insert into `admin`(id, name, password_hash)
values (1, 'root', '53175bcc0524f37b47062fafdda28e3f8eb91d519ca0a184ca71bbebe72f969a'), # password is root
       (2, 'admin', 'fc8252c8dc55839967c58b9ad755a59b61b67c13227ddae4bd3f78a38bf394f7'); # password is admin

insert into `student`(id, name, password_hash, enroll_time)
values (1, 'st1', '2238ead9c048f351712c34d22b41f6eec218ea9a9e03e48fad829986b0dafc11', now()), # password is same as name
       (2, 'st2', '5e61d026a7889d9fc72e17f1b25f4d6d48bfe17046fea845aa8c5651ec89c333', now()),
       (3, 'st3', 'bbb977f8e93feb5dbd79e0688b822115b5acf774dd8a1fe6964e03d6b9579384', now()),
       (4, 'st4', '6133396ebcd382b137088d2ea91d60637744e404b4376e4635b45784b718db72', now()),
       (5, 'st5', 'd691a62aa63f1be970582902d0ff78df29899f09c5dd540b1447cdd051dcfc8d', now()),
       (6, 'st6', 'a7a287ffc9cb27131b9dc54199ba96cef87e753968bc620d714af212ef0f7a8c', now()),
       (7, 'st7', '73d0daf13c6159a1fbdeb37b6972325b6e29c312371a0f3d427bd35c0c87b928', now()),
       (8, 'st8', '4ce70fc1eef7303879a2ef33996db2f85058ae06e8590521267ae8d46ec59793', now());

insert into `teach`(cid, tid)
values (1, 1),
       (1, 2),
       (2, 1),
```

```
        (3, 1),
        (4, 2),
        (5, 2);

insert into `take`(cid, sid)
values (1, 1),
        (1, 2),
        (1, 3),
        (1, 4),
        (2, 3),
        (2, 4),
        (2, 5),
        (2, 6),
        (3, 7),
        (3, 8),
        (4, 1),
        (4, 3),
        (4, 5),
        (5, 2),
        (5, 4),
        (5, 6),
        (5, 8),
        (6, 1),
        (6, 7),
        (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'),
```

```
212         (3, '蒙特卡洛树搜索实现',
    'https://raw.githubusercontent.com/dendenxu/DeepOthello/master/MCTS.py'),
213         (4, 'JDBC接口调用参考与举例',
    'https://raw.githubusercontent.com/dendenxu/DeepOthello/master/MCTS.py');
214
215 insert into info(id, content, cid, release_time)
216 values (1, '作业1的提交就要截止啦！请大家及时关注。', 1, NOW()),
217         (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()),
218         (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()),
219         (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());
220
221 insert into attach_to(aid, uid)
222 values (1, 1),
223         (1, 2),
224         (1, 3),
225         (2, 1),
226         (2, 5),
227         (2, 6),
228         (4, 5),
229         (3, 1);
```