题目：命令行成绩优化工具

1 系统设计

1.1 设计要求

1.1.1 设计任务

以VMware虚拟机软件和Ubuntu16操作系统为平台，使用shell脚本编写程序，实现对学生成绩的统计计算等功能。使用Linux shell编写程序脚本（Bash风格），并提供友好的交互界面。

1.1.2 性能指标要求

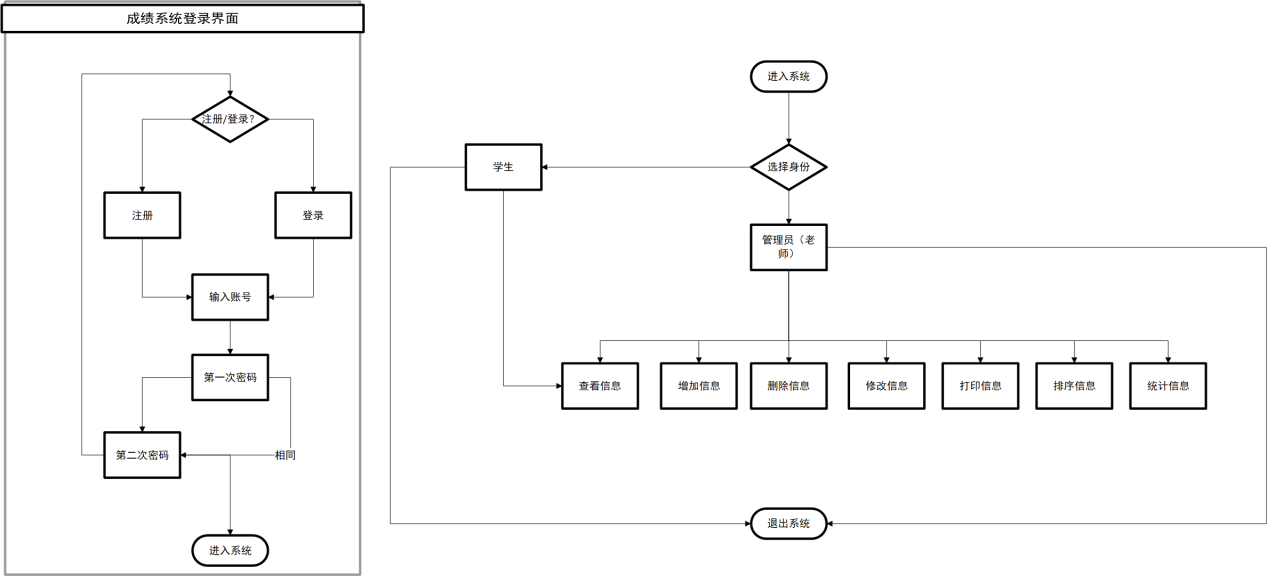
1. 基本功能要求
2. 按照题目要求完成程序源码编写、调试及关键代码注释
3. 以文本的形式存储各门功课成绩，生成成绩统计表
4. 实现成绩的计算、统计、排序、查询等功能
5. 能够以管理员身份对成绩进行管理
6. 扩展功能要求
7. 可以统计不同成绩段的人数，相同成绩的人数
8. 扩展其他功能，如界面的改进、增加提示信息等

1.2 设计思路及设计框图

1.2.1设计思路

该项目是对学生成绩信息进行管理，从用户角度考虑，可粗略分为管理员（老师）和学生两个类别。管理员可对学生信息进行增删查改等操作，而学生只能查看自己的成绩，不能查看其他学生信息。基于此，设计思路框图如下。

1.2.2总体设计框图



2 各个模块程序的设计

本项目共有三个模块，分别是登录模块、主要模块和拓展模块。

登录模块：该模块放在main.sh中，用于实现用户进入学生系统后对身份进行选择。main.sh存放“选择身份菜单”、“管理员菜单”和“学生菜单”，用户选择身份后，将会跳转至对应身份菜单选择“注册”或“登录”模式。注册账号时，系统会判断所输入的信息是否合法，例如两次输入的密码是否一致，以及账号是否已经存在。若在登录前未进行注册，则无法进入操作系统。由于管理员和学生的账户信息存储在不同文本中，因此即便选择管理员和学生的账号密码相同，也不会进入到同一个操作系统中。管理员的权限是可执行所有操作，而学生的权限仅能查看自己的成绩信息。

主要模块：这是整个项目中最关键的部分，我分了两个bash来实现该模块，一是用于存放全局变量和创建文本的global.sh，二是实现基本功能的add\_del\_find\_modify\_sort.sh。global.sh里创建的文本分别用于保存基本功能存储的学生成绩（stu\_messgae.txt）、管理员（account.txt）和学生（s\_account.txt）的账号信息。 add\_del\_find\_modify\_sort.sh中实现用户增删查改和排序功能。增加信息时，利用grep指令，在stu\_messgae.txt搜索是否存在输入学号相同的行以判断能否将信息添加进stu\_messgae.txt中。删除、修改查询都是基于增加信息的代码基础上进行修改。查询信息可通过学号查询和通过姓名查询，查询学号可用grep实现，而查询姓名使用awk进行字符分割实现。排序信息可分为顺序排序和逆序排序，选择排序模式后可选择不同学科，因此在排序函数中会多次调用选择学科的菜单（course\_menu）。学科的排序可用head、tail和sort实现。

拓展模块：该模块放在statistics.sh中，用于实现对学生成绩进行不同分数段和相同成绩的人数统计。学生分数段可分为五等，不及格（0-59）、及格（60-69）、良好（70-79）、优秀（80-89）、非常优秀（90-100），实现代码方面从高分往下进行筛选，即先判断是否大于90，再判断是否大于80，反之会导致分数和等级不匹配的现象。创建联合数组，用“学号–成绩”作为数组的键，在循环里遍历整个stu\_messgae.txt并存入不同学科的联合数组中，可实现相同成绩的人数统计。

3 调试过程

登录模块：这个模块的调试过程主要是检查用户输入的合法性，对用户的输入进行合法性检查，避免出现空白输入、重复输入、非法字符等情况，以及确保账号信息的正确性。需要测试不同的情况，例如：用户选择不同的身份，是否能正确跳转到相应的菜单；用户输入的账号和密码是否与文本中存储的一致，是否能正确登录或注册；用户输入的账号和密码是否符合格式要求，例如长度、字符类型等；用户输入的账号是否已经存在，是否能避免重复注册；用户在登录前是否已经注册，是否能防止未注册用户进入系统。

主要模块：这是项目中最关键的部分，需要对全局变量和文本创建进行测试，以确保它们能正确地存储学生成绩和账号信息。然后，需要测试用例的增删查改和排序功能。在增加信息时，需要验证系统是否能正确地判断输入的学号是否已存在。删除、修改和查询功能都基于增加信息的代码，因此也需要进行测试。查询信息可以通过学号或姓名进行，需要验证这两种方式都能正确工作。排序信息可以顺序排序或逆序排序，并可以选择不同的学科，因此需要验证所有这些选项都能正确工作。

拓展模块：该模块需要验证分数段划分是否准确，并且相同成绩的人数统计是否正确。首先，检查statistics.sh文件是否有语法错误或逻辑错误，比如括号是否匹配，变量是否正确赋值，循环是否正常结束等。其次，运行statistics.sh文件，并输入学科名称和分数段，观察输出结果是否符合预期，比如分数段是否正确划分，相同成绩的人数是否准确统计。再次，对比statistics.sh文件的输出结果和stu\_messgae.txt文件的原始数据，检查是否有遗漏或错误的数据，比如学号或成绩是否正确读取，数组是否正确存储等。

4 功能测试

4.1 测试仪器与设备

VMware Workstation Pro

4.2 性能指标测试

1. 登录模块：
2. 测试用例：输入不同的账号、密码和身份，观察系统是否能正确响应
3. 测试目标：验证用户输入的合法性和账号信息的正确性
4. 测试指标：输入合法性检查的准确率，账号信息匹配的准确率，身份跳转的准确率
5. 测试方法：对不同的输入进行边界值分析和等价类划分，检查系统是否能按照预期输出
6. 主要模块：
7. 测试用例：输入不同的学生信息，观察系统是否能正确存储、修改、查询和排序
8. 测试目标：验证全局变量和文本创建的正确性，以及用户功能的完整性
9. 测试指标：全局变量、文本创建和用户功能的正确率，用户功能的执行时间
10. 测试方法：对全局变量和文本创建进行代码覆盖测试，检查是否有逻辑错误或语法错误
11. 拓展模块：
12. 测试用例：输入不同的学科名称和分数段，观察系统是否能正确输出分数段划分和相同成绩人数统计
13. 测试目标：验证分数段划分和相同成绩人数统计的准确性
14. 测试指标：分数段划分的准确率，相同成绩人数统计的准确率，输出结果的可读性
15. 测试方法：对不同的输入进行边界值分析和等价类划分，检查系统是否能按照预期输出。对statistics.sh文件进行代码覆盖测试，检查是否有语法错误或逻辑错误

5 心得体会

设计该项目的思路是完成“增删查改、打印和退出”的基本菜单，在主函数中使用case语句实现对各函数调用，以实现各个功能。先粗略实现“增加学生信息”模块，以确定“学号”“姓名”“性别”“英语成绩”等变量，将这些变量置成全局变量，并存入一个名为stu\_message的文本中，通过文本可知存入的信息有哪些，对此进行完善增加模块，再逐渐实现其他模块。

增加信息模块的实现思路是，遍历stu\_message中以学号为开头的行并赋给sno\_info，检查索引中是否存在与输入的学号相同的行，若存在则显示“学好已存在”，反之把输入的学生信息重定向进stu\_message中。捋清思路，但不意味着能顺利实现功能。用什么指令遍历以学号开头的行，如何读取stu\_message中的信息，学号唯一性无法实现等问题接踵而至。经过一番查找资料，才知道“^$sno[ \t]\*”能实现“以学号为开头的行”，而且正则表达式^不能写在$sno之后，而在$sno之前，且若是对sno没有进行取值，也无法进行学号唯一性验证。在该模块实现对总分和平均分的计算，则无须另外对这两项成绩进行计算。其中，平均分的计算用到bc这个数学操作指令，考虑到真实情况，我采用“scale=2”这个指令，将平均分取到小数点后两位。

完成增加信息模块后，对剩下模块的思路更为清晰。其次是打印信息模块，直接在主函数中使用cat语句则可实现。删除模块中仅需对增加模块进行适当删改，此外，使用sed流编辑器对相应学生的行进行操作，易错点在于“/^$sno/d”，正则表达式的位置颠倒，且 不查相关资料无法得知需要使用“//”来表示操作的数。查询信息模块中，我分为查询学号和查询名字两方面，查询学号可用sed实现，而查询名字则需要用awk切割空格和分隔符来才能实现功能。sed和awk都属于流编辑器，对输入流或者文件进行操作。两者区别在于，sed适合大规模修改，以行动单位处理字符流；而awk更适合多个模式或行操作复杂的任务，以定界符（用于分隔数据的字段）分隔的字段进行操作。修改模块直接在增加模块上修改即可，问题在于修改信息之后，不是在原本信息上进行修改，二是新增学生信息，且此时学号不唯一。反复检验和调整之后，才想到修改信息可以理解为：找出该学生信息并进行删除，再重新添加。此时，增删查改四个基本功能大致实现。

排序学生信息的思路，分为顺序（从小到大）和逆序（从大到小）两个方面进行排序，并且可以细分为用户自行选择排序的学科，因此需要写两个菜单，一是选择排序方式的菜单，二是选择学科的菜单，当用户选择第一个菜单之后，无论是顺序还是逆序排序，都会调用第二个菜单。“（head -n 1 $stu\_message && tail -n +2 $stu\_message | sort -t $'\t' -k5,5n) | cat”的指令修改几个参数即可实现所有科目的排序，一般来说，head打印文件前10行，tail打印文件后10行，加入限制条件后，即可实现打印第一行的文本行之后再打印后续数据行且不影响文本行的输出。其中易错点在于，当我实现增删查改并进行调试时，发现姓名和性别的间距太小，输入“zhangsan”后，所有数据行向后移动，导致文本与数据行无法对齐影响用户浏览效果。因此我在姓名和性别之间用空格进行分割，该操作导致之前所有关于grep查找学号相关的行指令出现问题，也导致学号唯一性无法验证。将“^$sno[\t]\*”改成“^$sno[ \t]\*”之后（[]之间有空格），才得以解决上述问题。此时出现一个新问题，原本写的排序指令无法实现排序功能，原因在于“(head -n 1 $stu\_message && tail -n +2 $stu\_message | sort -t $'\t' -k5,5n) | cat”指令中原本的第五列变成了第六列，导致每列的指令与信息不对应，得在原本的数字上加一才能修复该问题。

这时基本功能已全部实现。接下来是对操作系统的用户身份进行划分。上述所有功能全部归类为管理员可操作的范围，而学生端仅能查看自己的成绩，因此需要几个菜单来实现身份选择的功能。一是登录界面菜单，进入系统，首先选择管理员或学生身份，确定身份后，此时出现第二个菜单，该菜单是“登录”“注册”和“退出”菜单。选择注册操作，输入账号密码验证两次密码是否正确之后，才能将输入账号的数据存储起来，于是在stu\_message.txt下方还需要创建两个文本文件，分别用于存放学生账号信息和管理员账号信息。如何验证登录时输入的账号信息和已存在文本中的账号信息相比较，这是一个困扰我的要点。又是一番搜寻资料后得知，可用“if awk -v id="$id" -v p1="$p1" -F'\t' '$1 == id && $2 == p1 {found=1} END {if (found != 1) exit 2}' account.txt”指令进行操作。将账号密码的值赋给新的变量，并判断此时输入的值是否等于已经覆盖新变量的值，该判断指令中若为真则返回0，为假返回-1，但是可能操作时会出现异常，以防万一，设置返回2也表示无法输出判断两者是否相同。假设选择的身份是管理员，若是登录成功，则调用一开始写的主函数进行相应操作。假设选择的身份是学生，若是登录成功，则进入第三个菜单，即学生登录系统的菜单。该菜单只有两个操作，一是查看自己的成绩信息，二是退出系统。查看信息的代码可直接套用上述已经写好的查询学号代码，只不过要在此之前增加一条验证消息，即让学生输入自己的学号，用该学号比对登录该系统的学号，若是相同则输出stu\_message.txt中相同学号的成绩信息，反之提示不可查询。到这步，已实现了除统计外的所有功能。

统计学生模块分为“统计不同分数段”和“统计相同分数的人数”两方面。在统计不同分数段中，由于之前理论课学习不够投入，对指令掌握不熟悉，考虑不到可以使用IFS来实现切割，“while IFS=$' \t' read -r sno sname sex English Linux Java Mysql sum avg”该指令可读取一行内容，使用空格和制表符来分割这一行的内容，并将每个字段的值赋给对应的变量。在这个实现过程中，无法正确统计成绩信息、所有优秀成绩均被类为不及格等级、总分和平均分无法计算等问题接踵而至。设置一个新的变量：line\_number，让该变量在循环中自加并跳过第一个数据，这样才能不影响文本行的输出。把原本筛选的顺序倒置，即从高分往下进行筛选，先判断是否大于90，再判断是否大于80，才得以解决分数和等级不匹配的问题。在“统计不同分数段”的代码基础上进行修改，添加关联数组和键，使用for循环遍历数组中每个元素并打印出相应成绩，即可统计相同分数的人数。由于统计模块的两个操作都需要对各科成绩和总分平均分进行统计，因此需要调用同一个菜单，该菜单与排序模块中的课程菜单相同。

把所有代码完成并调试后确认没有问题，将脚本进行分割，共有四个脚本。一是global.sh，用于存放全局变量和创建文本文件；二是main.sh，实现用户进入学生系统后对身份进行选择；三是add\_del\_find\_modify\_sort.sh，实现增删查改和排序指令；四是statistics.sh，实现统计这个扩展功能。将分类完成的脚本再次执行，确认功能无误，至此实现该项目。

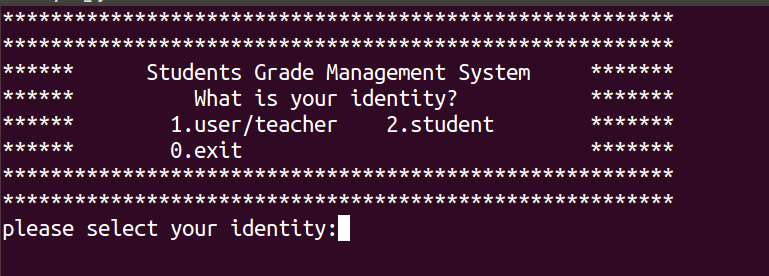
本次是一次对理论知识转化为实际应用的锻炼，设计思路－编写代码－调试脚本－修改代码－实现功能－完善功能，这几个步骤反复执行。该过程中，我的心态由“应该和C语言实现差不多吧”到“怎么这么难啊”，再到“哦！原来还可以这样实现”转变，才知道可以使用“bash -x ”对shell脚本进行逐语句调试，以此查看错误之处。整个项目多次用到grep指令，通过翻书才发现这是大三上课讲过，但是自己没有认真听，或是没有记住的知识点，awk和sed等指令也是如此。在设计功能和实现指令中，总是想到了如何实现，但不知如何下手，即便学过、即便记得该指令或该语句，但是在使用时会出现这样或那样的问题，这是对知识掌握不牢的一个重要表现。该项目的过程遇到“sno没有加上$”“把增加语句的代码复制到修改模块时个别变量没有改名”“密码输入-s之后提示语句全部放在同一行”等细节方面的问题，让我认识到学习之路永无止境，就是一个检验学习成果的机会，相信自己将会在后续的Linux学习中贯彻“求知、多练”的原则，为接下来的课程奠定扎实的理论基础，从源头上降低基础性问题再犯的概率。

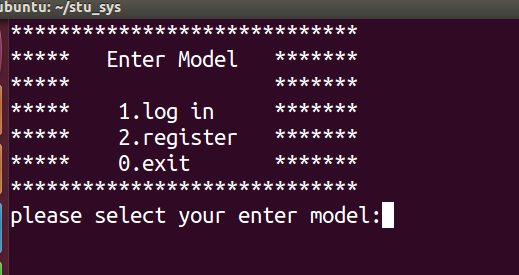
6 参考文献

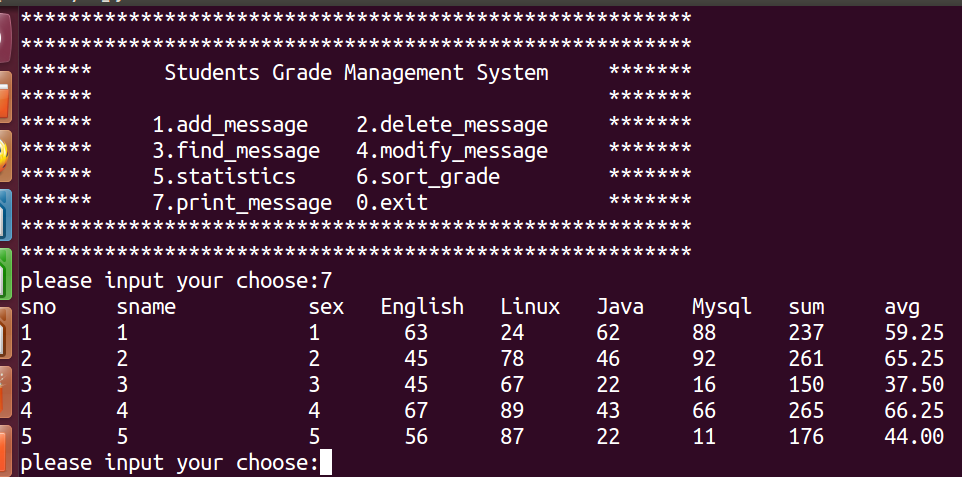
1. 韩桂明．嵌入式Ubuntu应用编程实战.杭州：浙江大学出版社，2014.10
2. 康维新．嵌入式Linux系统开发与应用．北京：机械工业出版社，2011.4

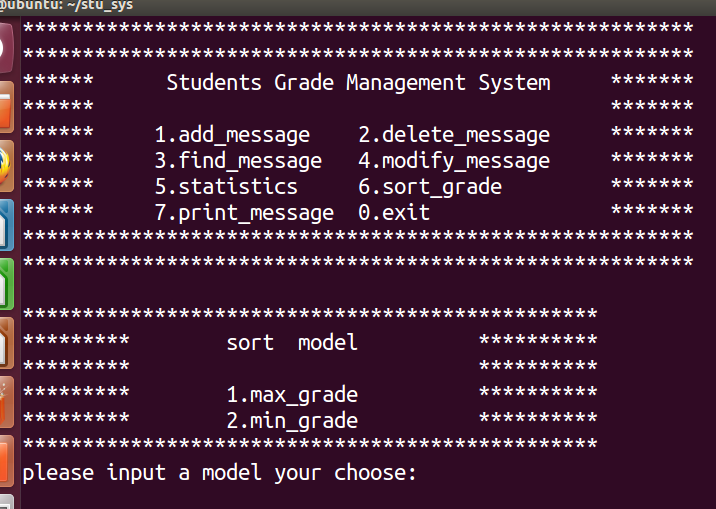
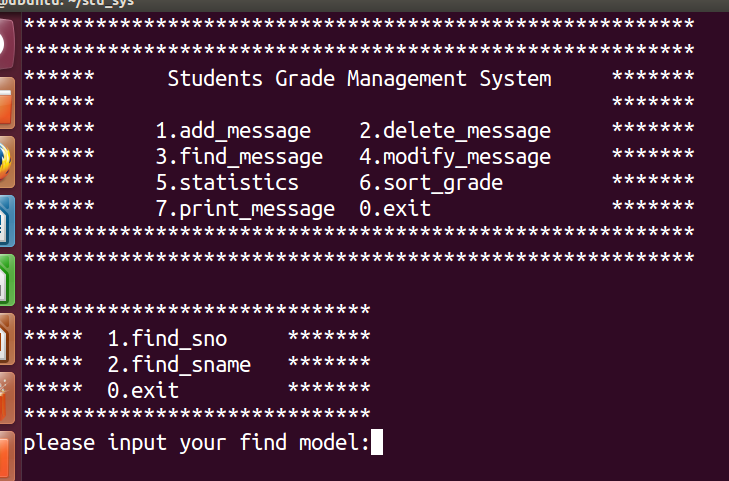
附录

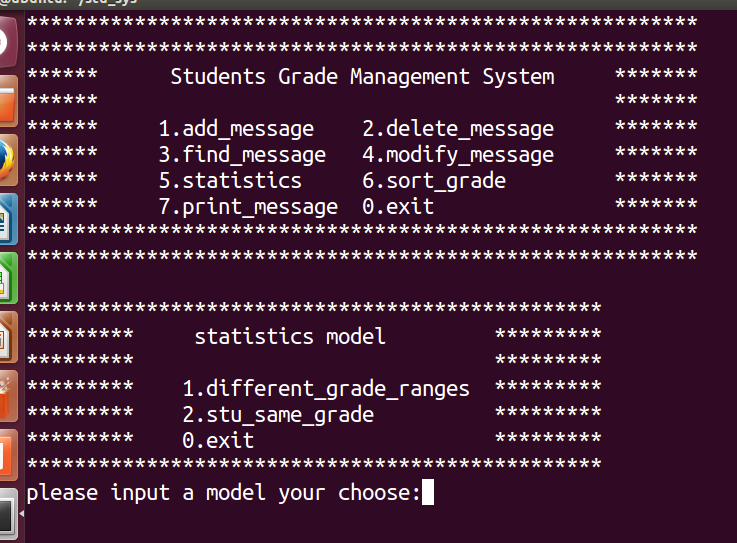
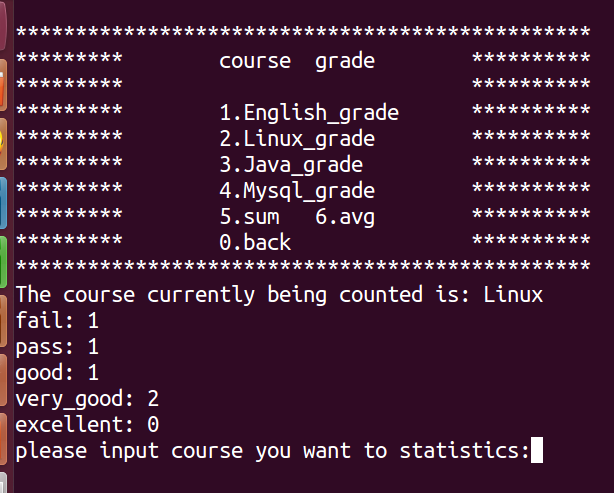
附录1：代码运行界面(管理员)



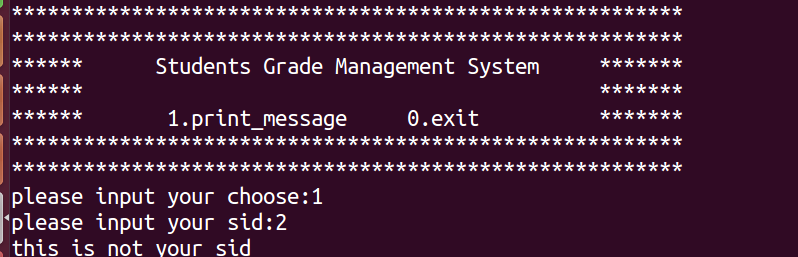
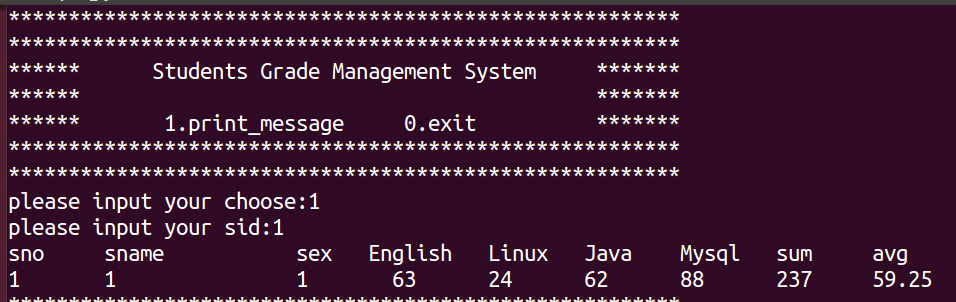








附录2：代码运行界面(学生)

附录2:程序清单

1. **main.sh**

#!/bin/bash

source /root/stu\_sys/global.sh

source /root/stu\_sys/add\_del\_find\_modify\_sort.sh

source /root/stu\_sys/statistics.sh

sys\_menu()

{

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo "\*\*\*\*\*\* Students Grade Management System \*\*\*\*\*\*\*"

echo "\*\*\*\*\*\* What is your identity? \*\*\*\*\*\*\*"

echo "\*\*\*\*\*\* 1.user/teacher 2.student \*\*\*\*\*\*\*"

echo "\*\*\*\*\*\* 0.exit \*\*\*\*\*\*\*"

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

}

enter\_menu()

{

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo "\*\*\*\*\* Enter Model \*\*\*\*\*\*\*"

echo "\*\*\*\*\* \*\*\*\*\*\*\*"

echo "\*\*\*\*\* 1.log in \*\*\*\*\*\*\*"

echo "\*\*\*\*\* 2.register \*\*\*\*\*\*\*"

echo "\*\*\*\*\* 0.exit \*\*\*\*\*\*\*"

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

}

#user\_sys\_menu

user\_sys\_menu(){

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo "\*\*\*\*\*\* Students Grade Management System \*\*\*\*\*\*\*"

echo "\*\*\*\*\*\* \*\*\*\*\*\*\*"

echo "\*\*\*\*\*\* 1.add\_message 2.delete\_message \*\*\*\*\*\*\*"

echo "\*\*\*\*\*\* 3.find\_message 4.modify\_message \*\*\*\*\*\*\*"

echo "\*\*\*\*\*\* 5.statistics 6.sort\_grade \*\*\*\*\*\*\*"

echo "\*\*\*\*\*\* 7.print\_message 0.exit \*\*\*\*\*\*\*"

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

}

#stu\_sys\_menu

stu\_sys\_menu()

{

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo "\*\*\*\*\*\* Students Grade Management System \*\*\*\*\*\*\*"

echo "\*\*\*\*\*\* \*\*\*\*\*\*\*"

echo "\*\*\*\*\*\* 1.print\_message 0.exit \*\*\*\*\*\*\*"

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

}

#clear screen

startUI()

{

clear

user\_sys\_menu

}

#user\_main

usermain()

{

startUI

while [ true ]

do

read -p "please input your choose:" input

case $input in

1)

add\_message;;

2)

delete\_message;;

3)

find\_message;;

4)

modify\_message;;

5)

statistics;;

6)

sort\_grade;;

7)

cat $stu\_message;;

0)

echo "exit";

clear;

break;;

\*)

echo "Invalid option";;

esac

done

}

s\_find\_sno()

{

read -p "please input your sid:" sno

count=$(grep -c "^$sno" $s\_account)

stu\_info=$(grep "^$sno[ \t]\*" $stu\_message)

if [ "$count" -ne 0 ]

then

echo -ne "sno\tsname\t\tsex English\tLinux\tJava\tMysql\tsum\tavg\n"

echo "$stu\_info"

else

echo "this is not your sid"

fi

}

#smain

s\_main()

{

clear

while true

do

stu\_sys\_menu

read -p "please input your choose:" choose

if [ "$choose" == "1" ]

then

s\_find\_sno

elif [ "$choose" == "0" ]

then

clear

break

else

echo "error"

fi

done

}

s\_register()

{

while true

do

read -p "please input your sid:" sid

count=$(grep -c "^$sid" $s\_account)

if [ "$count" -gt 0 ]

then

echo "account exist!"

return

else

break

fi

done

read -sp "please input your spassword:" sp1

echo ""

read -sp "please input your spassword again:" sp2

if [ "$sp1" != "$sp2" ]

then

echo "spassword different!"

else

echo -e "\nsuccessfully created!\n"

echo -ne "$sid\t$sp1\n" >> $s\_account

fi

}

s\_login()

{

read -p "please input your sid:" sid

read -sp "please input your spassword:" sp1

if awk -v sid="$sid" -v sp1="$sp1" -F'\t' '$1 == sid && $2 == sp1 {found=1} END {if (found != 1) exit 2}' s\_account.txt

then

echo -e "\nLogin successful!\n"

s\_main

else

echo -e "\nsid or spassword is incorrect!\n"

fi

}

s\_sys()

{

clear

while true

do

enter\_menu

read -p "please select your enter model:" enter

if [ "$enter" == "1" ]

then

s\_login

elif [ "$enter" == "2" ]

then

s\_register

elif [ "$enter" == "0" ]

then

clear

break

else

echo "error"

fi

done

sys\_menu

}

register()

{

while true

do

read -p "please input your id:" id

count=$(grep -c "^$id" $account)

if [ "$count" -gt 0 ]

then

echo "account exist!"

return

else

break

fi

done

read -sp "please input your password:" p1

echo ""

read -sp "please input your password again:" p2

if [ "$p1" != "$p2" ]

then

echo -e "\npassword different!\n"

else

echo -e "\nsuccessfully created!\n"

echo -ne "$id\t$p1\n" >> $account

fi

}

login()

{

read -p "please input your id:" id

read -sp "please input your password:" p1

if awk -v id="$id" -v p1="$p1" -F'\t' '$1 == id && $2 == p1 {found=1} END {

if (found != 1) exit 2}' account.txt

then

echo -e "\nLogin successful!\n"

usermain

else

echo -e "\nid or password is incorrect!\n"

fi

}

user\_sys()

{

clear

while true

do

enter\_menu

read -p "please select your enter model:" enter

if [ "$enter" == "1" ]

then

login

elif [ "$enter" == "2" ]

then

register

elif [ "$enter" == "0" ]

then

clear

break

else

echo "error"

fi

done

sys\_menu

}

#sys\_menu

clear

sys\_menu

while true

do

read -p "please select your identity:" identity

if [ "$identity" == "1" ]

then

user\_sys

elif [ "$identity" == "2" ]

then

s\_sys

elif [ "$identity" == "0" ]

then

break

else

echo "error"

fi

done

1. **global.sh**

#!/bin/bash

#global val

account=""

p1=""

p2=""

sno=""

sname=""

sex=""

English=""

Linux=""

Java=""

Mysql=""

sum=""

avg=""

account=/root/stu\_sys/account.txt

s\_account=/root/stu\_sys/s\_account.txt

stu\_message=/root/stu\_sys/stu\_message.txt

if [ ! -f "$account" ]

then

touch "$account"

echo -ne "id\tpassword\n" >> $account

fi

if [ ! -f "$s\_account" ]

then

touch "$s\_account"

echo -ne "sid\tspassword\n" >> $s\_account

fi

if [ ! -f "$stu\_message" ]

then

touch "$stu\_message"

echo -ne "sno\tsname\t\tsex English\tLinux\tJava\tMysql\tsum\tavg\n" >> $stu\_message

fi

1. **add\_del\_find\_modify\_sort.sh**

#!/bin/bash

source /root/stu\_sys/global.sh

add\_message()

{

startUI

read -p "please input sno you want to add:" sno

count=$(grep "^$sno[ \t]\*" $stu\_message | wc -l)

if [ "$count" == 1 ]

then

echo "sno already repeat!"

else

read -p "please input sname:" sname

read -p "please input sex:" sex

read -p "please input English\_grade:" English

read -p "please input Linux\_grade:" Linux

read -p "please input Java\_grade:" Java

read -p "please input Mysql\_grade:" Mysql

sum=$((English + Linux + Java + Mysql))

avg=$(echo "scale=2; $sum / 4" | bc)

echo ""

echo "successfully added!"

echo -ne "$sno\t$sname\t\t$sex\t$English\t$Linux\t$Java\t$Mysql\t$sum\t$avg\n" >> $stu\_message

fi

}

delete\_message()

{

startUI

read -p "please input sno you want to delect:" sno

stu\_info=$(grep "^$sno[ \t]\*" $stu\_message)

if [ -z "$stu\_info" ]

then

echo "sno not found"

else

sed -i "/^$sno/d" $stu\_message

echo "successfully deleted!"

fi

}

find\_sno()

{

read -p "please input sno you want to find:" sno

stu\_info=$(grep "^$sno[ \t]\*" $stu\_message)

if [ -z "$stu\_info" ]

then

echo "sno not found"

else

echo "$(head -n 1 $stu\_message)"

echo "$stu\_info"

fi

}

find\_sname()

{

read -p "please input sname you want to find:" sname

stu\_info=$(awk -v name="$sname" -F '\t' '$2 == name {print}' $stu\_message)

if [ -z "$stu\_info" ]

then

echo "sname not found"

else

echo "$(head -n 1 $stu\_message)"

echo "$stu\_info"

fi

}

find\_menu()

{

echo ""

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo "\*\*\*\*\* 1.find\_sno \*\*\*\*\*\*\*"

echo "\*\*\*\*\* 2.find\_sname \*\*\*\*\*\*\*"

echo "\*\*\*\*\* 0.exit \*\*\*\*\*\*\*"

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

}

find\_message()

{

startUI

while true

do

find\_menu

read -p "please input your find model:" model

if [ "$model" == "1" ]

then

find\_sno

elif [ "$model" == "2" ]

then

find\_sname

elif [ "$model" == "0" ]

then

clear

break

else

echo "error!"

fi

done

startUI

}

modify\_message()

{

startUI

read -p "please input sno you want to modify:" sno

stu\_info=$(grep "^$sno[ \t]\*" $stu\_message)

if [ -z "$stu\_info" ]

then

echo "sno not found"

return

else

sed -i "/^$sno/d" $stu\_message

read -p "please input sname:" sname

read -p "please input sex:" sex

read -p "please input English\_grade:" English

read -p "please input Linux\_grade:" Linux

read -p "please input Java\_grade:" Java

read -p "please input Mysql\_grade:" Mysql

sum=$((English + Linux + Java + Mysql))

avg=$(echo "scale=2; $sum / 4" | bc)

echo ""

echo "successfully modified!"

echo -ne "$sno\t$sname\t\t$sex\t$English\t$Linux\t$Java\t$Mysql\t$sum\t$avg\n" >> $stu\_message

fi

}

sort\_menu()

{

echo ""

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo "\*\*\*\*\*\*\*\*\* sort model \*\*\*\*\*\*\*\*\*\*"

echo "\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*"

echo "\*\*\*\*\*\*\*\*\* 1.max\_grade \*\*\*\*\*\*\*\*\*\*"

echo "\*\*\*\*\*\*\*\*\* 2.min\_grade \*\*\*\*\*\*\*\*\*\*"

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

}

max\_grade()

{

#English-Linux-Java-Mysql-sum-avg

clear

course\_menu

while true

do

read -p "please input a course your want to sort:" course

if [ "$course" == "1" ]

then

clear

course\_menu

echo "the current subject being sorted is:English"

(head -n 1 $stu\_message && tail -n +2 $stu\_message | sort -t $'\t' -k5,5rn) | cat

elif [ "$course" == "2" ]

then

clear

course\_menu

echo "the current subject being sorted is:Linux"

(head -n 1 $stu\_message && tail -n +2 $stu\_message | sort -t $'\t' -k6,6rn) | cat

elif [ "$course" == "3" ]

then

clear

course\_menu

echo "the current subject being sorted is:Java"

(head -n 1 $stu\_message && tail -n +2 $stu\_message | sort -t $'\t' -k7,7rn) | cat

elif [ "$course" == "4" ]

then

clear

course\_menu

echo "the current subject being sorted is:Mysql"

(head -n 1 $stu\_message && tail -n +2 $stu\_message | sort -t $'\t' -k8,8rn) | cat

elif [ "$course" == "5" ]

then

clear

course\_menu

echo "the current subject being sorted is:sum"

(head -n 1 $stu\_message && tail -n +2 $stu\_message | sort -t $'\t' -k9,9rn) | cat

elif [ "$course" == "6" ]

then

clear

course\_menu

echo "the current subject being sorted is:avg"

(head -n 1 $stu\_message && tail -n +2 $stu\_message | sort -t $'\t' -k10,10rn) | cat

elif [ "$course" == "0" ]

then

break

else

clear

course\_menu

echo "error"

fi

done

startUI

}

min\_grade()

{

#English-Linux-Java-Mysql-sum-avg

clear

course\_menu

while true

do

read -p "please input a course your want to sort:" course

if [ "$course" == "1" ]

then

clear

course\_menu

echo "the current subject being sorted is:English"

(head -n 1 $stu\_message && tail -n +2 $stu\_message | sort -t $'\t' -k5,5n) | cat

elif [ "$course" == "2" ]

then

clear

course\_menu

echo "the current subject being sorted is:Linux"

(head -n 1 $stu\_message && tail -n +2 $stu\_message | sort -t $'\t' -k6,6n) | cat

elif [ "$course" == "3" ]

then

clear

course\_menu

echo "the current subject being sorted is:Java"

(head -n 1 $stu\_message && tail -n +2 $stu\_message | sort -t $'\t' -k7,7n) | cat

elif [ "$course" == "4" ]

then

clear

course\_menu

echo "the current subject being sorted is:Mysql"

(head -n 1 $stu\_message && tail -n +2 $stu\_message | sort -t $'\t' -k8,8n) | cat

elif [ "$course" == "5" ]

then

clear

course\_menu

echo "the current subject being sorted is:sum"

(head -n 1 $stu\_message && tail -n +2 $stu\_message | sort -t $'\t' -k9,9n) | cat

elif [ "$course" == "6" ]

then

clear

course\_menu

echo "the current subject being sorted is:avg"

(head -n 1 $stu\_message && tail -n +2 $stu\_message | sort -t $'\t' -k10,10n) | cat

elif [ "$course" == "0" ]

then

break

else

echo "error"

fi

done

startUI

}

sort\_grade()

{

startUI

sort\_menu

read -p "please input a model your choose:" choose

if [ "$choose" == "1" ]

then

max\_grade

elif [ "$choose" == "2" ]

then

min\_grade

else

echo "error"

fi

}

1. **statistics.sh**

#!/bin/bash

source /root/stu\_sys/global.sh

course\_menu()

{

echo ""

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo "\*\*\*\*\*\*\*\*\* course grade \*\*\*\*\*\*\*\*\*\*"

echo "\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*"

echo "\*\*\*\*\*\*\*\*\* 1.English\_grade \*\*\*\*\*\*\*\*\*\*"

echo "\*\*\*\*\*\*\*\*\* 2.Linux\_grade \*\*\*\*\*\*\*\*\*\*"

echo "\*\*\*\*\*\*\*\*\* 3.Java\_grade \*\*\*\*\*\*\*\*\*\*"

echo "\*\*\*\*\*\*\*\*\* 4.Mysql\_grade \*\*\*\*\*\*\*\*\*\*"

echo "\*\*\*\*\*\*\*\*\* 5.sum 6.avg \*\*\*\*\*\*\*\*\*\*"

echo "\*\*\*\*\*\*\*\*\* 0.back \*\*\*\*\*\*\*\*\*\*"

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

}

select\_course\_same()

{

while true

do

read -p "please input course you want to statistics:" course

if [ "$course" == "1" ]

then

clear

course\_menu

echo "The course currently being counted is: English"

same\_grade\_English

elif [ "$course" == "2" ]

then

clear

course\_menu

echo "The course currently being counted is: Linux"

same\_grade\_Linux

elif [ "$course" == "3" ]

then

clear

course\_menu

echo "The course currently being counted is: Java"

same\_grade\_Java

elif [ "$course" == "4" ]

then

clear

course\_menu

echo "The course currently being counted is: Mysql"

same\_grade\_Mysql

elif [ "$course" == "5" ]

then

clear

course\_menu

echo "The course currently being counted is: sum"

same\_grade\_sum

elif [ "$course" == "6" ]

then

clear

course\_menu

echo "The course currently being counted is: avg"

same\_grade\_avg

elif [ "$course" == "0" ]

then

clear

break

else

clear

course\_menu

echo "error!"

fi

done

}

select\_course\_differ()

{

while true

do

read -p "please input course you want to statistics:" course

if [ "$course" == "1" ]

then

clear

course\_menu

echo "The course currently being counted is: English"

range\_grade\_English

elif [ "$course" == "2" ]

then

clear

course\_menu

echo "The course currently being counted is: Linux"

range\_grade\_Linux

elif [ "$course" == "3" ]

then

clear

course\_menu

echo "The course currently being counted is: Java"

range\_grade\_Java

elif [ "$course" == "4" ]

then

clear

course\_menu

echo "The course currently being counted is: Mysql"

range\_grade\_Mysql

elif [ "$course" == "5" ]

then

clear

course\_menu

echo "unable to judge"

elif [ "$course" == "6" ]

then

clear

course\_menu

echo "The course currently being counted is: avg"

range\_grade\_avg

elif [ "$course" == "0" ]

then

clear

break

else

clear

course\_menu

echo "error!"

fi

done

}

sta\_menu()

{

echo ""

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo "\*\*\*\*\*\*\*\*\* statistics model \*\*\*\*\*\*\*\*\*"

echo "\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*"

echo "\*\*\*\*\*\*\*\*\* 1.different\_grade\_ranges \*\*\*\*\*\*\*\*\*"

echo "\*\*\*\*\*\*\*\*\* 2.stu\_same\_grade \*\*\*\*\*\*\*\*\*"

echo "\*\*\*\*\*\*\*\*\* 0.exit \*\*\*\*\*\*\*\*\*"

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

}

stu\_same\_grade()

{

clear

course\_menu

select\_course\_same

}

different\_grade\_ranges()

{

clear

course\_menu

select\_course\_differ

}

declare -A score\_counts\_avg

declare -A score\_counts\_sum

declare -A score\_counts\_English

declare -A score\_counts\_Linux

declare -A score\_counts\_Java

declare -A score\_counts\_Mysql

same\_grade\_avg()

{

line\_number=0

# 读取一行内容，使用空格和制表符来分割这一行的内容，并将每个字段的值赋给对应的变量

while IFS=$' \t' read -r sno sname sex English Linux Java Mysql sum avg

do

((line\_number++))

if [ $line\_number == "1" ]

then

continue

fi

sum=$((English + Linux + Java + Mysql))

avg=$(echo "scale=2; $sum / 4" | bc)

key="$sno--$avg"

#第一对方括号 [[ ... ]] 是用来包含整个条件表达式的，而第二对方括号 [...] 是用来从关联数组 score\_counts\_avg 中获取键为 $avg 的元素

if [[ -v score\_counts\_avg[$key] ]]

then

#((...))：这是一个算术复合命令，用于执行算术运算

#$((...))：这是一个算术扩展，用于执行算术运算并返回结果

score\_counts\_avg[$key]=$((score\_counts[$key]+1))

else

score\_counts\_avg[$key]=1

fi

done < $stu\_message

for score in ${!score\_counts\_avg[@]}

do

echo "$score: appears ${score\_counts\_avg[$score]} times"

done

}

same\_grade\_sum()

{

line\_number=0

# 读取一行内容，使用空格和制表符来分割这一行的内容，并将每个字段的值赋给对应的变量

while IFS=$' \t' read -r sno sname sex English Linux Java Mysql sum avg

do

((line\_number++))

if [ $line\_number == "1" ]

then

continue

fi

sum=$((English + Linux + Java + Mysql))

key="$sno--$sum"

#第一对方括号 [[ ... ]] 是用来包含整个条件表达式的，而第二对方括号 [...] 是用来从关联数组 score\_counts 中获取键为 $avg 的元素

if [[ -v score\_counts\_sum[$key] ]]

then

#((...))：这是一个算术复合命令，用于执行算术运算

#$((...))：这是一个算术扩展，用于执行算术运算并返回结果

score\_counts\_sum[$key]=$((score\_counts\_sum[$key]+1))

else

score\_counts\_sum[$key]=1

fi

done < $stu\_message

for score in ${!score\_counts\_sum[@]}

do

echo "$score: appears ${score\_counts\_sum[$score]} times"

done

}

same\_grade\_English()

{

line\_number=0

# 读取一行内容，使用空格和制表符来分割这一行的内容，并将每个字段的值赋给对应的变量

while IFS=$' \t' read -r sno sname sex English Linux Java Mysql sum avg

do

((line\_number++))

if [ $line\_number == "1" ]

then

continue

fi

key="$sno--$English"

#第一对方括号 [[ ... ]] 是用来包含整个条件表达式的，而第二对方括号 [...] 是用来从关联数组 score\_counts\_English 中获取键为 $avg 的元素

if [[ -v score\_counts\_English\_English[$key] ]]

then

#((...))：这是一个算术复合命令，用于执行算术运算

#$((...))：这是一个算术扩展，用于执行算术运算并返回结果

score\_counts\_English[$key]=$((score\_counts[$key]+1))

else

score\_counts\_English[$key]=1

fi

done < $stu\_message

for score in ${!score\_counts\_English[@]}

do

echo "$score: appears ${score\_counts\_English[$score]} times"

done

}

same\_grade\_Linux()

{

line\_number=0

# 读取一行内容，使用空格和制表符来分割这一行的内容，并将每个字段的值赋给对应的变量

while IFS=$' \t' read -r sno sname sex English Linux Java Mysql sum avg

do

((line\_number++))

if [ $line\_number == "1" ]

then

continue

fi

key="$sno--$Linux"

#第一对方括号 [[ ... ]] 是用来包含整个条件表达式的，而第二对方括号 [...] 是用来从关联数组 score\_counts\_Linux 中获取键为 $avg 的元素

if [[ -v score\_counts\_Linux[$key] ]]

then

#((...))：这是一个算术复合命令，用于执行算术运算

#$((...))：这是一个算术扩展，用于执行算术运算并返回结果

score\_counts\_Linux[$key]=$((score\_counts[$key]+1))

else

score\_counts\_Linux[$key]=1

fi

done < $stu\_message

for score in ${!score\_counts\_Linux[@]}

do

echo "$score: appears ${score\_counts\_Linux[$score]} times"

done

}

same\_grade\_Java()

{

line\_number=0

# 读取一行内容，使用空格和制表符来分割这一行的内容，并将每个字段的值赋给对应的变量

while IFS=$' \t' read -r sno sname sex English Linux Java Mysql sum avg

do

((line\_number++))

if [ $line\_number == "1" ]

then

continue

fi

key="$sno--$Java"

#第一对方括号 [[ ... ]] 是用来包含整个条件表达式的，而第二对方括号 [...] 是用来从关联数组 score\_counts\_Java 中获取键为 $avg 的元素

if [[ -v score\_counts\_Java[$key] ]]

then

#((...))：这是一个算术复合命令，用于执行算术运算

#$((...))：这是一个算术扩展，用于执行算术运算并返回结果

score\_counts\_Java[$key]=$((score\_counts[$key]+1))

else

score\_counts\_Java[$key]=1

fi

done < $stu\_message

for score in ${!score\_counts\_Java[@]}

do

echo "$score: appears ${score\_counts\_Java[$score]} times"

done

}

same\_grade\_Mysql()

{

line\_number=0

# 读取一行内容，使用空格和制表符来分割这一行的内容，并将每个字段的值赋给对应的变量

while IFS=$' \t' read -r sno sname sex English Linux Java Mysql sum avg

do

((line\_number++))

if [ $line\_number == "1" ]

then

continue

fi

key="$sno--$Mysql"

#第一对方括号 [[ ... ]] 是用来包含整个条件表达式的，而第二对方括号 [...] 是用来从关联数组 score\_counts\_Mysql 中获取键为 $avg 的元素

if [[ -v score\_counts\_Mysql[$key] ]]

then

#((...))：这是一个算术复合命令，用于执行算术运算

#$((...))：这是一个算术扩展，用于执行算术运算并返回结果

score\_counts\_Mysql[$key]=$((score\_counts[$key]+1))

else

score\_counts\_Mysql[$key]=1

fi

done < $stu\_message

for score in ${!score\_counts\_Mysql[@]}

do

echo "$score: appears ${score\_counts\_Mysql[$score]} times"

done

}

range\_grade\_avg()

{

fail=0

pass=0

good=0

very\_good=0

excellent=0

line\_number=0

#读取一行内容，使用空格和制表符来分割这一行的内容，并将每个字段的值赋给对应的变量

while IFS=$' \t' read -r sno sname sex English Linux Java Mysql sum avg

do

((line\_number++))

if [ $line\_number == "1" ]

then

continue

fi

sum=$((English + Linux + Java + Mysql))

avg=$(echo "scale=2; $sum / 4" | bc)

if [ $(echo "$avg >= 90" | bc -l) == "1" ]

then

((excellent++))

elif [ $(echo "$avg >= 80" | bc -l) == "1" ]

then

((very\_good++))

elif [ $(echo "$avg >= 70" | bc -l) == "1" ]

then

((good++))

elif [ $(echo "$avg >= 60" | bc -l) == "1" ]

then

((pass++))

else

((fail++))

fi

done < $stu\_message

echo "fail: $fail"

echo "pass: $pass"

echo "good: $good"

echo "very\_good: $very\_good"

echo "excellent: $excellent"

}

range\_grade\_English()

{

fail=0

pass=0

good=0

very\_good=0

excellent=0

line\_number=0

#读取一行内容，使用空格和制表符来分割这一行的内容，并将每个字段的值赋给对应的变量

while IFS=$' \t' read -r sno sname sex English Linux Java Mysql sum avg

do

((line\_number++))

if [ $line\_number == "1" ]

then

continue

fi

if [ "$English" -ge 90 ]

then

((excellent++))

elif [ "$English" -ge 80 ]

then

((very\_good++))

elif [ "$English" -ge 70 ]

then

((good++))

elif [ "$English" -ge 60 ]

then

((pass++))

else

((fail++))

fi

done < $stu\_message

echo "fail: $fail"

echo "pass: $pass"

echo "good: $good"

echo "very\_good: $very\_good"

echo "excellent: $excellent"

}

range\_grade\_Java()

{

fail=0

pass=0

good=0

very\_good=0

excellent=0

line\_number=0

#读取一行内容，使用空格和制表符来分割这一行的内容，并将每个字段的值赋给对应的变量

while IFS=$' \t' read -r sno sname sex English Linux Java Mysql sum avg

do

((line\_number++))

if [ $line\_number == "1" ]

then

continue

fi

if [ "$Java" -ge 90 ]

then

((excellent++))

elif [ "$Java" -ge 80 ]

then

((very\_good++))

elif [ "$Java" -ge 70 ]

then

((good++))

elif [ "$Java" -ge 60 ]

then

((pass++))

else

((fail++))

fi

done < $stu\_message

echo "fail: $fail"

echo "pass: $pass"

echo "good: $good"

echo "very\_good: $very\_good"

echo "excellent: $excellent"

}

range\_grade\_Linux()

{

fail=0

pass=0

good=0

very\_good=0

excellent=0

line\_number=0

#读取一行内容，使用空格和制表符来分割这一行的内容，并将每个字段的值赋给对应的变量

while IFS=$' \t' read -r sno sname sex English Linux Java Mysql sum avg

do

((line\_number++))

if [ $line\_number == "1" ]

then

continue

fi

if [ "$Linux" -ge 90 ]

then

((excellent++))

elif [ "$Linux" -ge 80 ]

then

((very\_good++))

elif [ "$Linux" -ge 70 ]

then

((good++))

elif [ "$Linux" -ge 60 ]

then

((pass++))

else

((fail++))

fi

done < $stu\_message

echo "fail: $fail"

echo "pass: $pass"

echo "good: $good"

echo "very\_good: $very\_good"

echo "excellent: $excellent"

}

range\_grade\_Mysql()

{

fail=0

pass=0

good=0

very\_good=0

excellent=0

line\_number=0

#读取一行内容，使用空格和制表符来分割这一行的内容，并将每个字段的值赋给对应的变量

while IFS=$' \t' read -r sno sname sex English Linux Java Mysql sum avg

do

((line\_number++))

if [ $line\_number == "1" ]

then

continue

fi

if [ "$Mysql" -ge 90 ]

then

((excellent++))

elif [ "$Mysql" -ge 80 ]

then

((very\_good++))

elif [ "$Mysql" -ge 70 ]

then

((good++))

elif [ "$Mysql" -ge 60 ]

then

((pass++))

else

((fail++))

fi

done < $stu\_message

echo "fail: $fail"

echo "pass: $pass"

echo "good: $good"

echo "very\_good: $very\_good"

echo "excellent: $excellent"

}

statistics()

{

startUI

while true

do

sta\_menu

read -p "please input a model your choose:" choose

if [ "$choose" == "1" ]

then

different\_grade\_ranges

elif [ "$choose" == "2" ]

then

stu\_same\_grade

elif [ "$choose" == "0" ]

then

clear

break

else

echo "error"

fi

done

startUI

}