****



**SHANGHAI UNIVERSITY**

**操作系统（一）实验报告**

|  |  |
| --- | --- |
| **组 号** | **第 7 组** |
| **学号姓名** | **20121034 胡才郁** |
| **实验序号** | **6** |
| **日期** | **2022年10月27日** |

实验六 SHELL 编程

# **1 实验目的与要求**

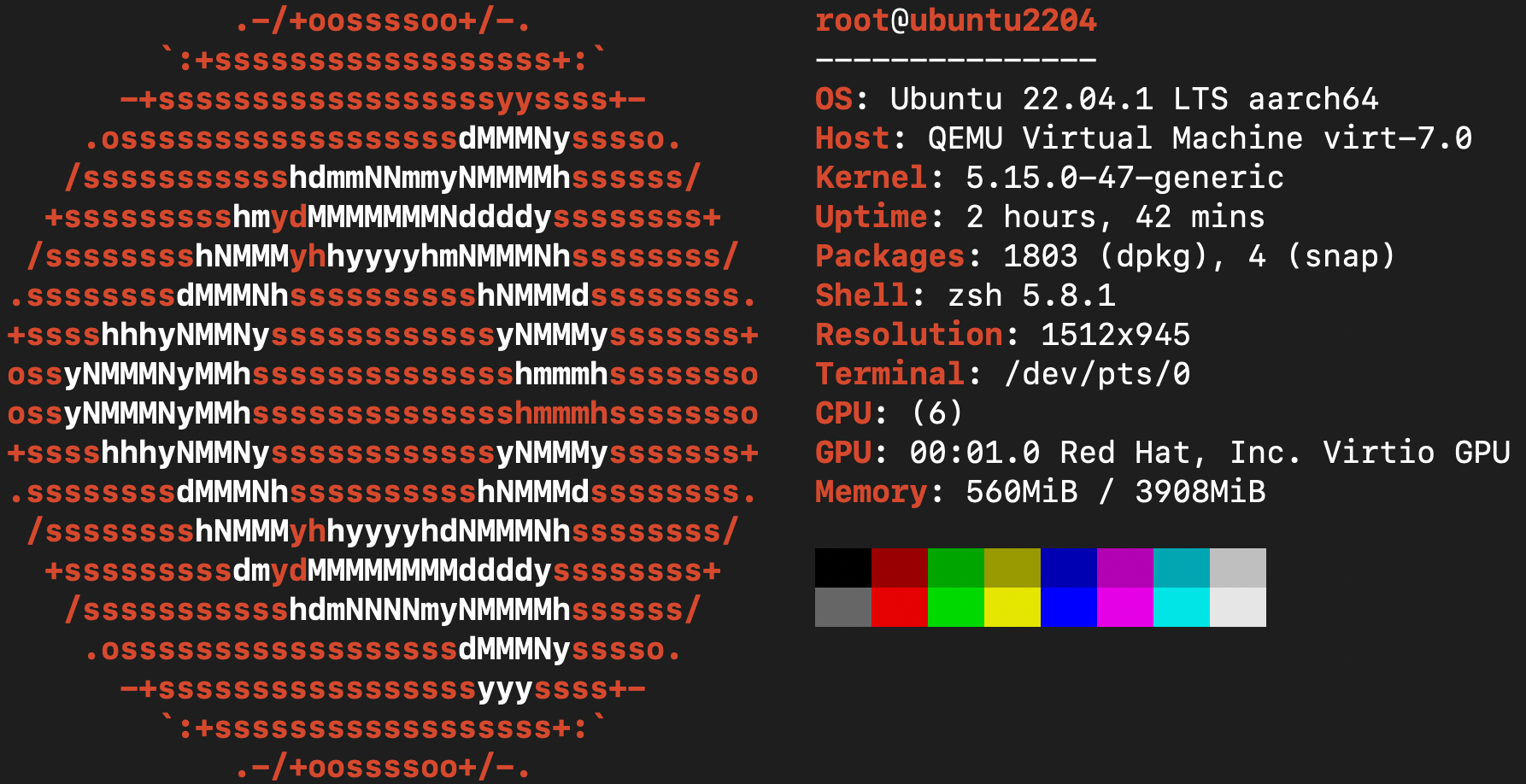
1. 掌握vi 的三种工作方式，熟悉vi 编辑程序的使用。

2. 学习Shell 程序设计方法。掌握编程要领。

# **2 实验环境**

* 操作系统: Ubuntu 22.04
* 宿主机:QEMU Virtual Machine

其余主要环境如下图所示：



**图1. 实验环境**

# **3 实验内容及其设计与实现**

**问题一**：**按本《实验指导》第三部分的内容。熟悉vi 的三种工作方式。熟悉使用各种编辑功能。**

**思考题：试一试vi 的三种工作方式各用在何时？用什么命令进入插入方式？怎样退出插**

**入方式？文件怎样存盘？注意存盘后的提示信息。**

**答：**vi或vim主要分为命令模式、末行模式、编辑模式三种，灵活在三种模式之间切换来完成文本编辑工作。

①命令模式：

1）在该模式中，可以输入命令来执行许多种功能；

2）打开文件首先进入命令模式，它是使用 vim 编辑器的入口

②末行模式：

1）将文件保存或退出 vi，也可以设置编辑环境，如寻找字符串、列出行号等；

2）末行模式是 vim 编辑器对的出口，要退出 vim，必须要在末行模式下。

③编辑模式：可以对文本进行编辑操作。

Vi或Vim的各种快捷键十分复杂，对于Vim各种操作的各种总结有很多，其中最著名的应当是Vim keymap，如下图，对应每一个键位对应的操作。



**图2. Vim Keymap**

Vim一定是所有Linux程序员都会经历的一个坎，曾经看到过一个笑话：

|  |
| --- |
| Q: 如何生成一个随机字符串?  A: 让Vim新手退出Vim. |

只能说十分的生动形象了…Vim确实对于新手不太友好，但是熟练使用Vim后会对于工作效率大幅度提升，不过学习的过程可能会异常复杂。由于Vim的十分灵活，可以自定义配置各种快捷键以及大量的插件支持，至少就我而言，尝试过使用[Vimtutor](http://www2.geog.ucl.ac.uk/~plewis/teaching/unix/vimtutor)来练习Vim的操作，但是对于Vim还是用不来，Vscode+远程SSH连接的方式还是更适合我。

**问题二：创建和执行Shell 程序**

**用前面介绍的 Vi 或其他文本编辑器编写 Shell 程序，并将文件以文本文件方式保存在相应的目录中。用chmod 将文件的权限设置为可执行模式，如若文件名为 shdemo.h,则命令如下：**

|  |
| --- |
| $ chmod 755 shdemo.h (文件主可读、写、执行，同组人和其他人可读和执行)  在提示符后执行 Shell 程序：  $ shdemo.h （直接键入程序文件名执行）  或 $ sh shdemo.h （执行 Shell 程序）  或 $ .shdemo.h （没有设置权限时可用点号引导） |

**答：**可以使用umask -S指令查看默认文件、文件夹的权限。其中的-S选项输出符号选项，帮助更清晰了解文件权限。通过下图输出可知，默认的文件权限为文件拥有者具有读、写、执行权限，同组用户具有读、执行，非同组的其他用户具有读、执行权限。

**文本

描述已自动生成**

**图3. 查看默认权限**

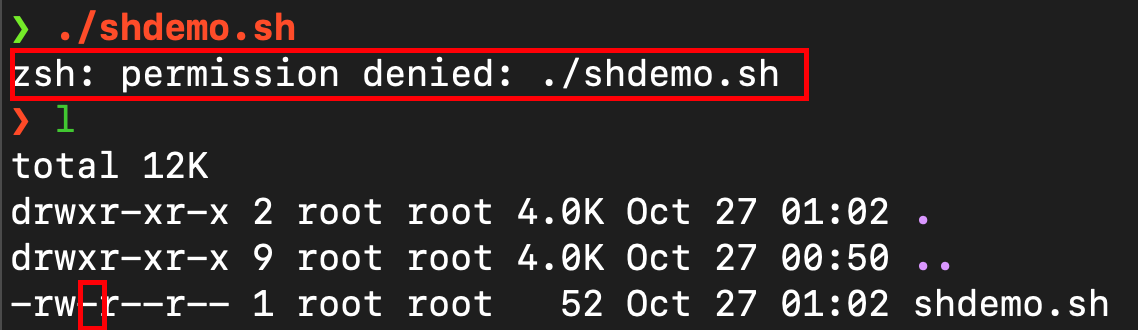
尽管在Linux操作系统下万物皆文件，但是目录(d)和文件(-)的三种权限之间可以允许的操作是不同的，具体而言如下表所示：

**表1. 文件与目录不同权限的操作**

|  |  |  |
| --- | --- | --- |
| **权限** | **文件** | **目录** |
| 读取权限r | 具有读取文件内容的权限 | 具有浏览目录与子目录的权限 |
| 写入权限w | 具有修改文件内容的权限 | 具有增加和删除目录内文件的权限 |
| 执行权限x | 具有执行文件的权限 | 具有进入目录的权限 |

可以将目录类比为抽屉，文件类比为抽屉中的日记本，如果目录具有读取权限r，而文件没有，就相当于可以打开抽屉看到一个个日记本，知道这些日记本的名字，知道日记本是谁的，但由于没有读取文件内容的权限，不能打开日记本阅读。

可能是出于安全的考虑，Linux系统新建文件默认没有执行权限，其最大权限为rw-rw-rw-（666），新建目录的最大权限为rwx-rwx-rwx（777）。而默认的权限与umask命令有关。修改文件的权限之前，并没有权限执行新创建的shdemo.sh文件。使用chmod命令修改，加入权限类型后，即可执行。如下图所示：

****

**文本

描述已自动生成**

**图4. 权限修改前后运行情况**

**问题三：用 vi 编写《实验指导》“第四部分 Shell 程序设计”中的例 1，练习内部变量和位置参数的用法。**

|  |
| --- |
| 用chmod将文件的权限设置为可执行模式，并在提示符后键入命令行：  $./prog1.sh 或 $sh prog1.sh #没有参数  屏幕显示:  Name not provided  在提示符后键入命令行：  $./prog1.sh Theodore #有一个参数  屏幕显示:  Your name is Theodore #引用$1 参数的效果 |

**答：**针对本问题，$#为内部变量，内部变量是Linux提供的一种特殊变量，具体而言$#表示传送给Shell程序的位置参数的个数。

图形用户界面, 文本, 应用程序

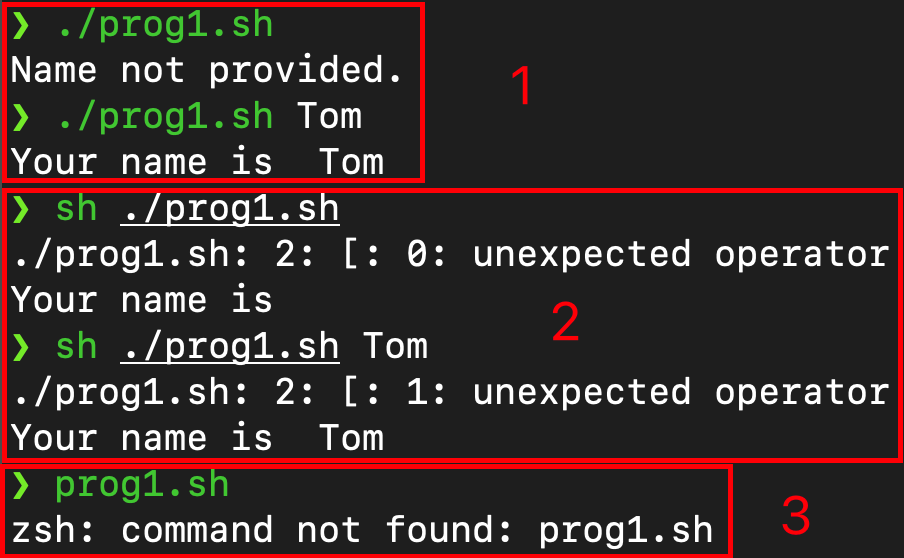
描述已自动生成

**图5. 问题三代码**

与这个问题的代码完成相比，更值得讨论的是执行Shell脚本的方法。根据实验指导书的写法，实际上分为了3种运行Shell脚本的方式，分别为

1. ./ 相对路径引用
2. sh 用sh执行
3. 可执行文件名

观察下图分别为1、2、3的结果可以观察到，对于上述代码，只有第一种方式可以正常执行，而第2种直接报错unexpected operator，第3种报错command not found。关于其中的原因，接下来进行解释。



**图6. 三种方式运行sh文件**

第一种情况使用./ 的方式执行可执行文件，观察代码第一行的#!/bin/bash，#在shell脚本种用作注释行，但是在第一行是一个例外，#后面的!会告诉shell用哪一个shell来执行脚本，在本例中是/bin目录下的bash。这个操作也就意味着在命令行敲下./prog.sh时，这个命令行在我的实验环境中，此处为zsh，他作为父进程，创建一个子进程/bin/bash来运行我的脚本。

而第二种情况相当于使用二进制文件sh来执行脚本文件prog1.sh，至于为什么在此处的运行结果会出错，观察下图：



**图7. 寻找sh根源位置**

当我在shell中敲下命令sh时，系统会优先在环境变量中有无名字为sh的文件，再在当前目录下寻找有无名字为sh的文件。这里使用which命令查看sh的绝对路径，发现路径为/usr/bin/sh，不过，这里还存在着软链接、硬链接等类似于“快捷方式”的问题，因此使用readlink -f命令查找到sh的最终指向为/usr/bin/dash。

到这里，第二种方式会报错的原因就比较显然了。此处相当于使用dash这个Shell来执行prog1.h这个文件，而不同的shell之间语法有差异，不可以完全兼容，具体到此处，在dash中，bash的中括号语法[]在dash中不兼容，因此有了之前的报错unexpected operator。

而第三种方式直接敲下prog1.sh时，系统会认为这是命令，自然出错。

综上，最建议的运行脚本方式是使用第一种方式./ 相对路径引用，如此使用可以在不清楚这个脚本的前提下保证不因为shell选择而出错。其次在清楚此脚本是由何种脚本语言编写的情况下，使用类似bash prog1.sh与zsh prog.sh此类方式执行。

**问题四：进一步修改程序prog1.h，要求显示参数个数、程序名字，并逐个显示参数。**

**答：**此题主要是对 $ 特殊参数以及循环编写的考察，将没有传入参数与传入参数的执行逻辑分开，代码如下图所示：

**文本

描述已自动生成**

**图8. 问题四代码**

运行结果如下图，标号1为无参数传入情况，标号2为有参数传入情况。

**文本

描述已自动生成**

**图9. 问题四运行结果**

**问题五：修改例1程序（即上面的 prog1.h），用read命令接受键盘输入。若没有输入显示第一种提示，否则第二种提示。**

**答：**运行结果如下图所示，标号1为无参数传入情况，标号2为有参数传入情况。

文本

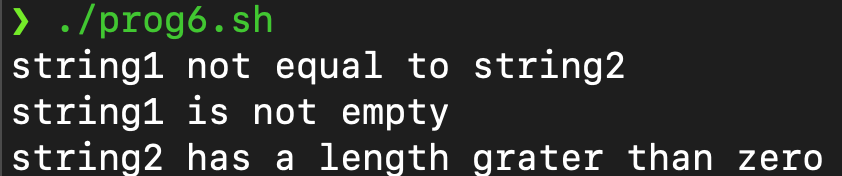
描述已自动生成

**图10. 问题五运行结果**

**问题六：用vi 编写《实验指导》“第四部分 Shell 程序设计”中的例2、例3，练习字符串比较运算符、数据比较运算符和文件运算符的用法，观察运行结果。**

**答：**例2部分：

文本

描述已自动生成

**图11. 例2代码与运行结果**

例3部分：由于在我的实验环境下为root用户，拥有write权限，因此第4行输出与指导书上不同。

文本

中度可信度描述已自动生成**文本

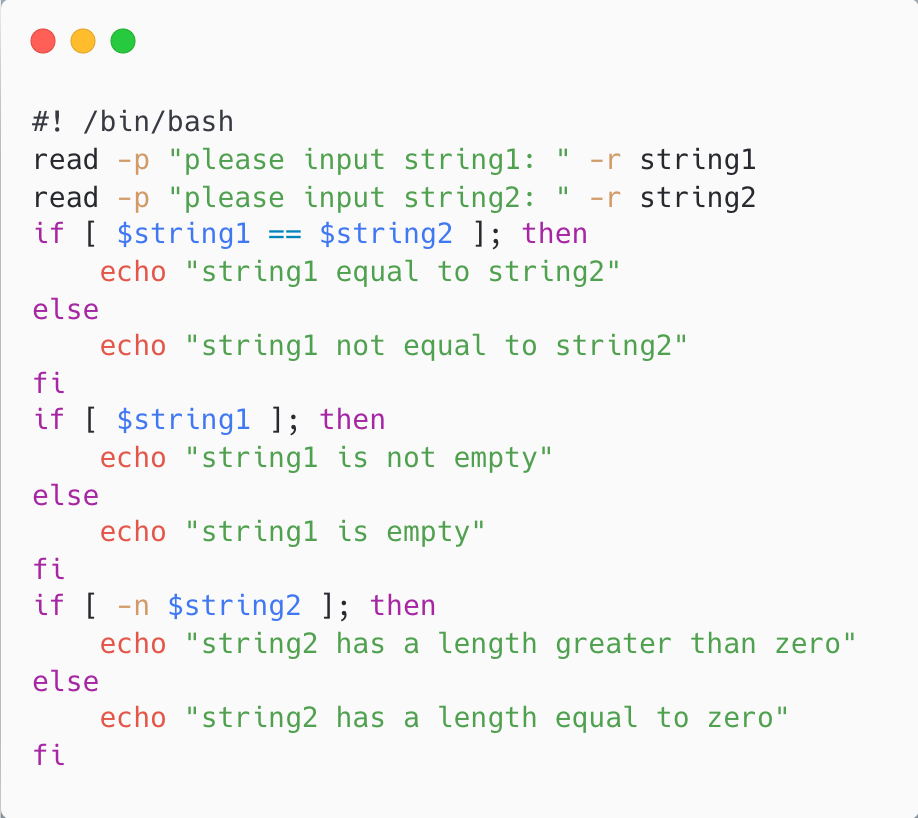
描述已自动生成**

**图12. 例3代码与运行结果**

**问题七：修改例2程序，使在程序运行中能随机输入字符串，然后进行字符串比较。**

**答：**以下是比较字符串时需要注意的几点：

1. 必须在二元运算符和操作数之间使用空格。
2. 始终在变量名称周围使用双引号以避免任何单词拆分或通配问题。
3. Bash不按“类型”隔离变量，变量根据上下文被视为整数或字符串。

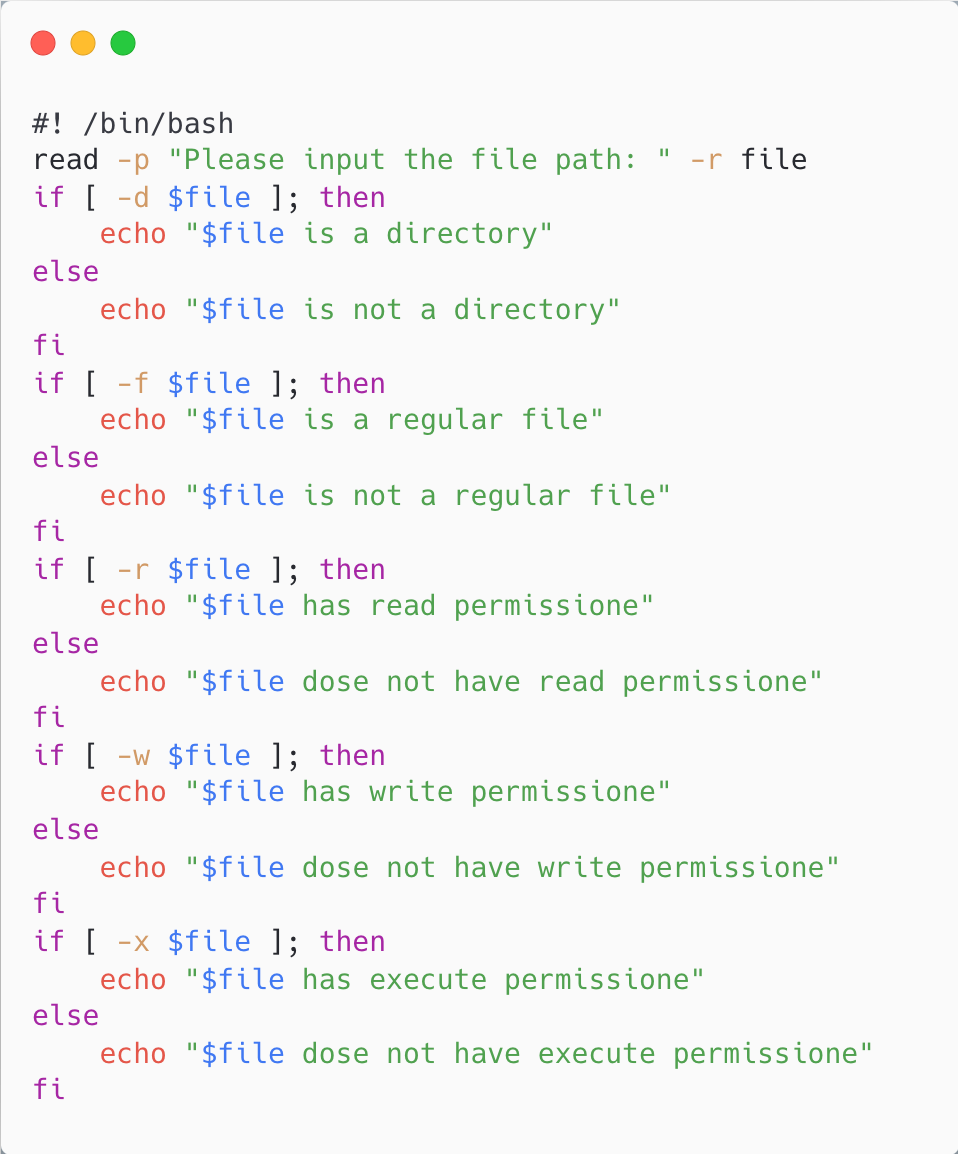
文本

描述已自动生成

**图13. 问题七代码与运行结果**

**问题八：修改例 3 程序，使在程序运行中能随机输入文件名，然后进行文件属性判断。**

**答：**两种类型分别进行测试：

文本

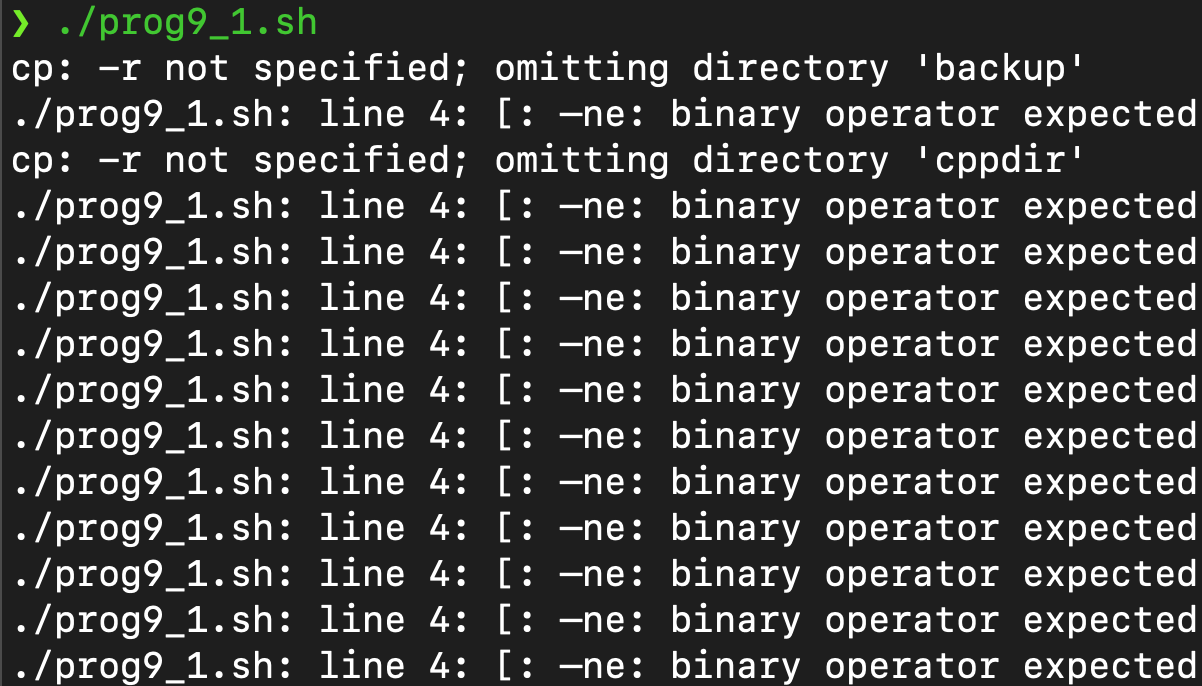
描述已自动生成

**图14. 问题八代码与运行结果**

**问题九：**用 vi 编写《实验指导》“第四部分 Shell 程序设计”中的例 4、例 5、例6、例 7，掌握控制语句的用法，观察运行结果。

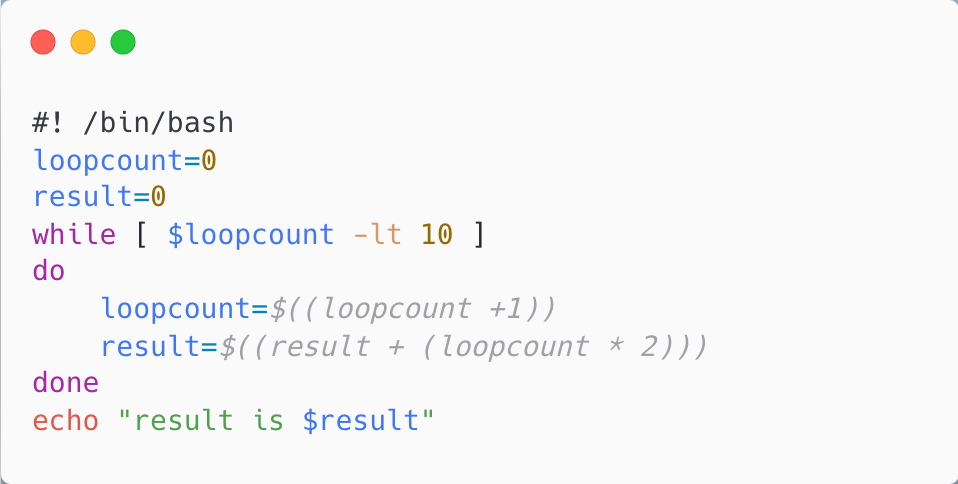
**答：**例四：

文本

中度可信度描述已自动生成

**图15. 例四代码与运行结果**

例五：



**图16. 例五代码**

例六：

图形用户界面, 文本, 应用程序

描述已自动生成

**图17. 例六代码与运行结果**

文本

描述已自动生成

**图18. 例五、例六运行结果**

例七：

图片包含 文本

描述已自动生成文本

描述已自动生成

**图19. 例七代码与运行结果**

**问题十：**用vi 编写《实验指导》“第四部分 Shell 程序设计”中的例8 及例9掌握条件语句的用法，函数的用法，观察运行结果。

**答：**这两个例子练习了bash中的case语法与函数的写法。例：

文本

描述已自动生成文本

中度可信度描述已自动生成

**图20. 例八代码与运行结果**

例9：

文本

描述已自动生成图形用户界面

中度可信度描述已自动生成

**图21. 例九代码与运行结果**

**问题十一：编程，在屏幕上显示用户主目录名（HOME）、命令搜索路径（PATH），并显示由位置参数指定的文件的类型和操作权限。**

**答：**使用echo命令输出相应的环境变量即可。

文本

描述已自动生成

文本

中度可信度描述已自动生成

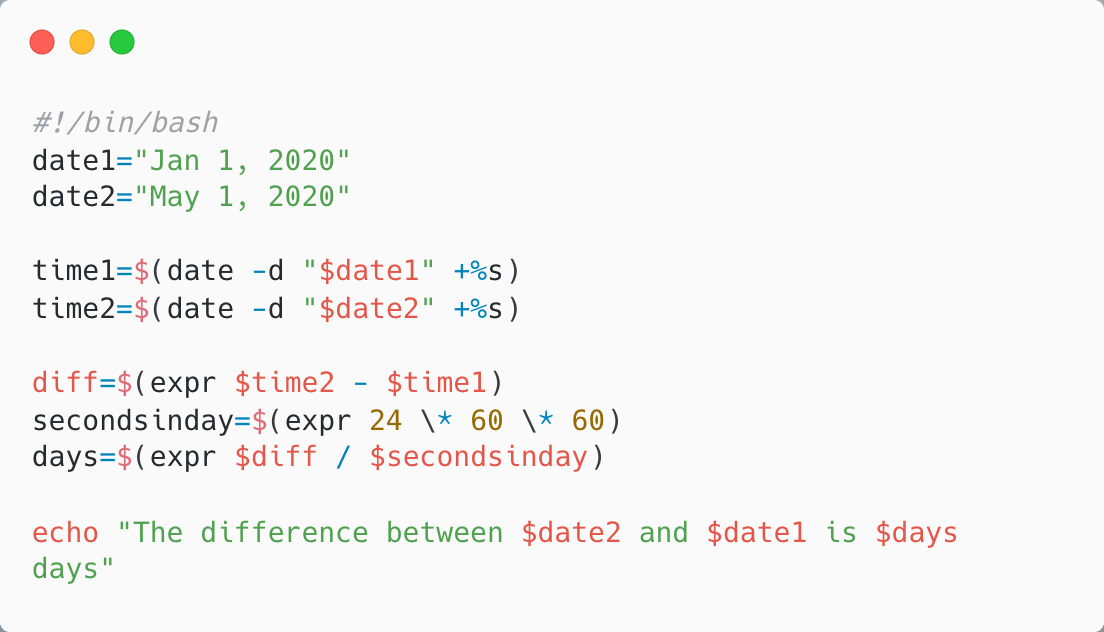
**图22. 问题十一代码与运行结果**

**思考题：到此为止你对Shell 有所认识了吧？怎么样？自己再编两个程序：**

**① 做个批处理程序，体会一下批处理概念。**

**② 做个菜单，显示系统环境参数。将此程序设置为人人可用。**

**答：**对于第1个问题，编写了一个计算两个日期之间的时间之差:

****

****

**图23. 思考题1代码与运行结果**

观察运行结果，发现符合预期，这个脚本主要练习了shell中的数值计算方式。值得一提的是，bash中默认是不支持浮点数运算的，在bash中使用浮点数运算可以使用“awk”脚本，另一种shell语言zsh是支持浮点数运算的。

第2个问题代码以及运行结果如下图所示，其中env top set分别为查看系统参数的命令，输入quit时退出。

图形用户界面, 文本, 应用程序

描述已自动生成

文本

描述已自动生成

**图24. 思考题2代码与运行结果**

**4 讨论**

**1.Linux 的Shell 有什么特点？**

普通意义上的shell就是可以接受用户输入命令的程序。它之所以被称作shell是因为它隐藏了操作系统低层的细节。同样的Linux下的图形用户界面GNOME和KDE，有时也被叫做“虚拟shell”或“图形shell”。Linux操作系统下的shell既是用户交互的界面，也是控制系统的脚本语言。

Shell 在几乎所有操作系统上都很常见，因为它高效且易于更新，监控计算机系统并执行例行备份。使用Shell时，无需切换语法，因为 shell 的语法和命令与在命令行中输入的相同。 此外，编写 shell 脚本既简单又快捷，它的启动速度快，而且易于调试。

Linux 中的 shell 有多种类型，例如zsh、bash、sh、dash等等。本次实验中，使用的shell语法是bash。

**2.怎样进行Shell编程？如何运行？有什么条件？**

怎样进行Shell编程这个问题太宽泛了。就我个人而言，学习的书籍是《Linux命令行与shell脚本编程大全》这本书，这本书前几章介绍Linux相关环境，后几章介绍具体的脚本编写的语法等等，并且有配套练习的资源可供下载，学习价值很高。Shell编程对于不同类型的Shell语法之间不能完全兼容，主流的是bash，zsh等等也有许多开发者在使用，Shell编程用到时忘记了再查，反复练习效率会高。

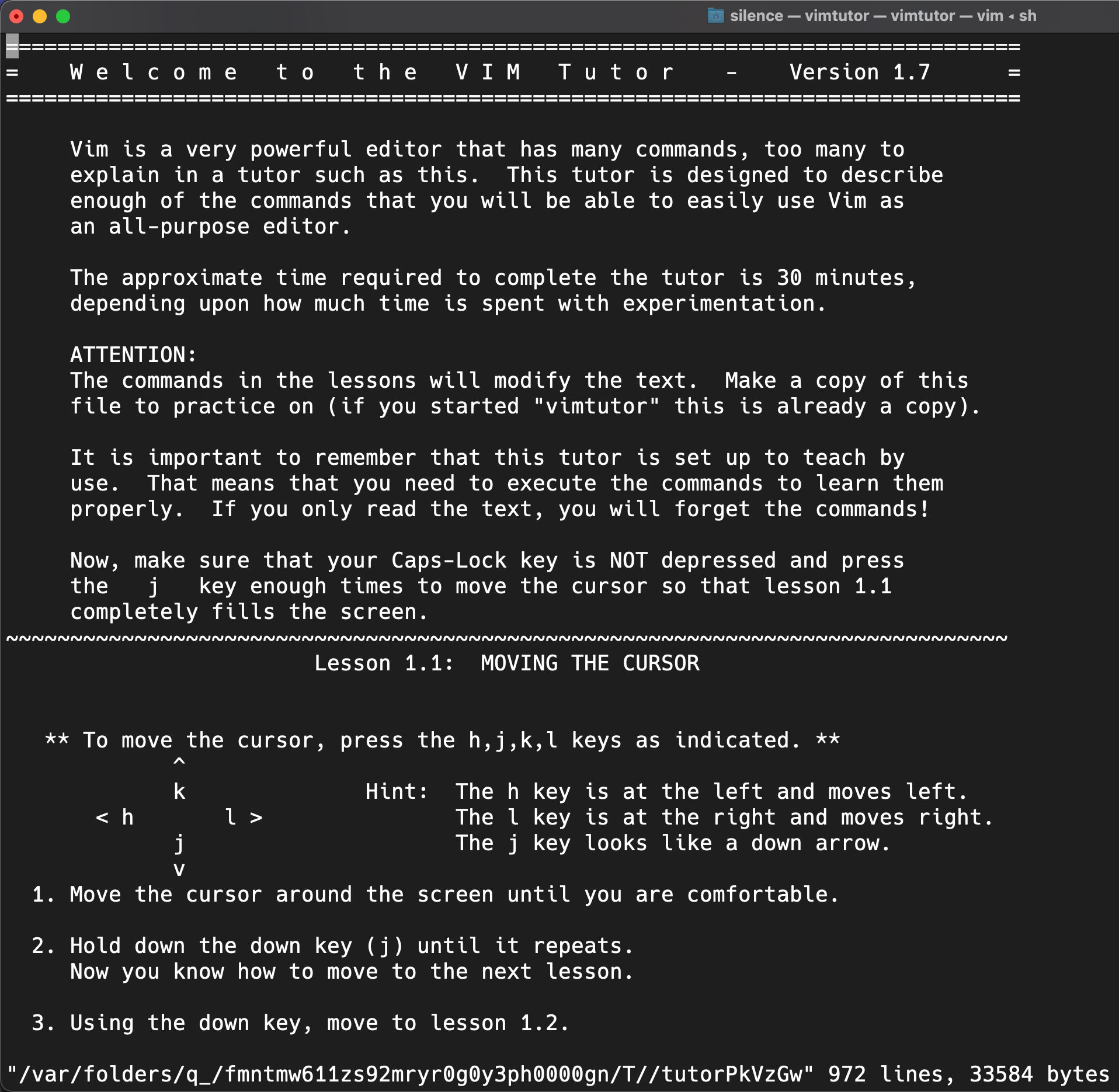
关于Shell运行的几种方式，在回答问题三的扩展时已经有了具体的提及以及理由。总结而言为最建议的运行脚本方式是使用 ./ 相对路径引用，如此使用可以在不清楚这个脚本的前提下保证不因为shell选择而出错。其次在清楚此脚本是由何种脚本语言编写的情况下，使用类似bash prog1.sh与zsh prog.sh此类方式执行。

Shell脚本运行的条件与逻辑与其他一般的编程语言类似，具体有下面几种方式：

1. 顺序执行：程序从上到下顺序执行
2. 选择执行：程序执行过程中，根据条件的不同，进行选择不同分支继续执行
3. 循环执行：程序执行过程中需要重复执行多次某段语句

**3. vi 编辑程序有几种工作方式？查找有关的详细资料，熟练掌握屏幕编辑方式、转移命令方式以及末行命令的操作。学习搜索、替换字符、字和行，行的复制、移动，以及在vi中执行Shell命令的方式。**

Vi编辑程序有3种方式，对于Vim操作的练习我尝试过Vimtutor。Vimtutor是一个帮助练习Vim的Linux下的软件，具体界面如下：

****

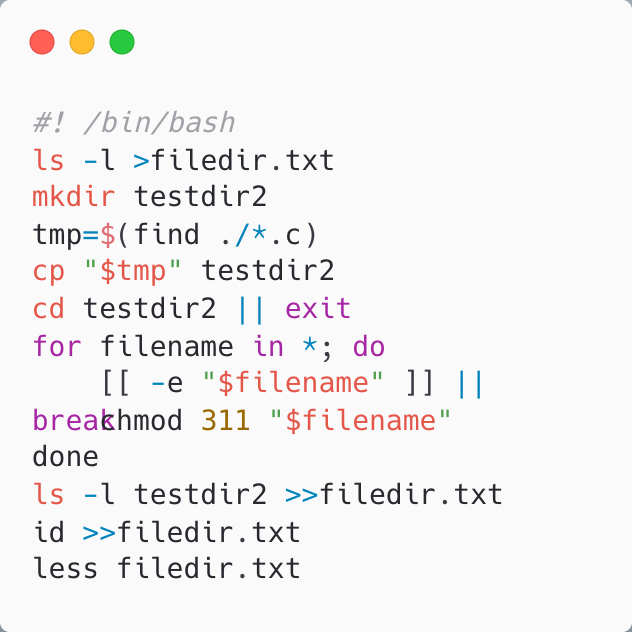
**图25. Vimtutor界面**

关于搜索、替换字符、字和行，行的复制、移动，以及在vi中执行Shell命令的方式在Vimtutor中都有详细的操作练习。Vim的强大之处不仅仅在于原生的完全使用键盘代替鼠标的操作，还有许多方便的插件，例如Vim代码补全，Vim快捷键扩展等等。

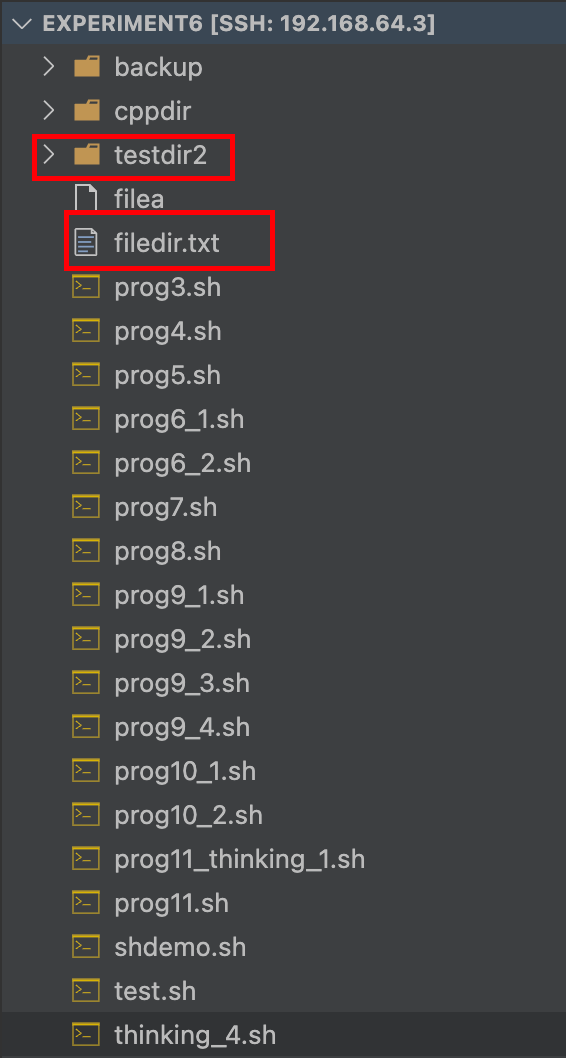
**4.** **编写一个具有以下功能的Shell程序。**

|  |
| --- |
| (1) 把当前目录下的文件目录信息输出到文件 filedir.txt 中；  (2) 在当前目录下建立一个子目录，目录名为 testdir2 ；  (3) 把当前目录下的所有扩展名为 c 的文件以原文件名复制到子目录testdir2中；  (4) 把子目录中的所有文件的存取权限改为不可读。（提示：用 for 循环控制语句实现，循环的控制列表用 ’ls’ 产生。）  (5) 在把子目录 testdir2 中所有文件的目录信息追加到文件 filedir.txt 中；  (6) 把你的用户信息追加到文件 filedir.txt 中；  (7) 分屏显示文件 filedir.txt |

具体的代码与结果如下图所示：

****

**图26. 讨论题4代码**

**图形用户界面, 文本, 应用程序

描述已自动生成**

**图27. 讨论题4结果**

可以看到由于less的分屏作用，最后脚本停留在了上图界面，并且testdir2目录也成功创建，filedir.txt文件也存储在其中。

# **5 收获与体会**

本次实验主要涉及了Shell编程，Shell编程的基础在服务器编程方面比较重要，如果工作去运维岗位的话，使用Shell编写脚本一定是必会技能。对于Vim，就算用其他的编辑器，至少要学会基本操作，毕竟从零安装Linux配置环境时没有那么多编辑器可以选择，还是需要用Vim更改，例如sshd文件中的登陆权限。