

实验报告

实验名称	实验一 Linux 常用命令（一）		
实验教室	丹青 922	实验日期	2023 年 3 月 10 日
学 号	2021213277	姓 名	周陆琦
专业班级	计算机科学与技术 05 班		
指导教师	卢洋		

东北林业大学
信息与计算机科学技术实验中心

一、 实验目的

- 1、掌握Linux下文件和目录操作命令：cd、ls、mkdir、rmdir、rm
- 2、掌握Linux下文件信息显示命令：cat、more、head、tail
- 3、掌握Linux下文件复制、删除及移动命令：cp、mv
- 4、掌握 Linux 的文件排序命令：sort

二、 实验环境

- (1) 计算机的硬件配置 PC 系列微机。
- (2) 计算机的软件配置通过 wsl 安装的 ubuntu 虚拟机。

三、 实验内容及结果

1. 使用命令切换到/etc 目录，并显示当前工作目录路径

```
lqzhou@LAPTOP-BOMFC06B:~$ cd /etc
lqzhou@LAPTOP-BOMFC06B:/etc$ pwd
/etc
lqzhou@LAPTOP-BOMFC06B:/etc$ |
```

- 2、使用命令显示/home/lqzhou 目录下所有文件目录的详细信息，包括隐藏文件。

```
lqzhou@LAPTOP-BOMFC06B:~$ cd /home/lqzhou
lqzhou@LAPTOP-BOMFC06B:~$ pwd
/home/lqzhou
lqzhou@LAPTOP-BOMFC06B:~$ ls -a
.          .git          .ssh          a.tar.gz
..         .gitconfig    .sudo_as_admin_successful  bar.txt
.bash_history .lessht       .vim           hello.c
.bash_logout .local        .viminfo       linux-test-report
.bashrc      .motd_shown   .vscode-server readme
.dotnet      .profile      .wget-hsts
```

- 3、使用命令创建目录/home/lqzhou/linux，然后删除该目录。

```

lqzhou@LAPTOP-BOMFC06B:~$ ls
a.tar.gz bar.txt hello.c linux-test-report readme
lqzhou@LAPTOP-BOMFC06B:~$ mkdir /home/lqzhou/linux
lqzhou@LAPTOP-BOMFC06B:~$ ls
a.tar.gz bar.txt hello.c linux linux-test-report readme
lqzhou@LAPTOP-BOMFC06B:~$ pwd
/home/lqzhou
lqzhou@LAPTOP-BOMFC06B:~$ cd linux
lqzhou@LAPTOP-BOMFC06B:~/linux$ cd
lqzhou@LAPTOP-BOMFC06B:~$ rmdir linux
lqzhou@LAPTOP-BOMFC06B:~$ ls
a.tar.gz bar.txt hello.c linux-test-report readme
lqzhou@LAPTOP-BOMFC06B:~$ cd linux
-bash: cd: linux: No such file or directory

```

4、使用命令 cat 用输出重定向在/home/lqzhou 目录下创建文件 foo, 文件内容为“Hello, Linux!”, 并查看该文件的内容

```

lqzhou@LAPTOP-BOMFC06B:~$ ls
a.tar.gz bar.txt hello.c linux-test-report readme
lqzhou@LAPTOP-BOMFC06B:~$ cat > /home/lqzhou/foo <& EOF
> "Hello, Linux!"
> EOF
lqzhou@LAPTOP-BOMFC06B:~$ ls
a.tar.gz bar.txt foo hello.c linux-test-report readme
lqzhou@LAPTOP-BOMFC06B:~$ cat foo
"Hello, Linux!"

```

5、使用命令创建目录 /home/lqzhou/foo.bak, 然后将 /home/lqzhou/foo 文件复制到该目录下, 最后将该目录及其目录下的文件一起删除。

```

lqzhou@LAPTOP-BOMFC06B:~$ ls
a.tar.gz bar.txt foo hello.c linux-test-report readme
lqzhou@LAPTOP-BOMFC06B:~$ mkdir /home/lqzhou/foo.bak
lqzhou@LAPTOP-BOMFC06B:~$ ls
a.tar.gz bar.txt foo foo.bak hello.c linux-test-report readme
lqzhou@LAPTOP-BOMFC06B:~$ cp /home/lqzhou/foo /home/lqzhou/foo.bak
lqzhou@LAPTOP-BOMFC06B:~$ ls
a.tar.gz bar.txt foo foo.bak hello.c linux-test-report readme
lqzhou@LAPTOP-BOMFC06B:~$ cd /home/lqzhou/foo.bak
lqzhou@LAPTOP-BOMFC06B:~/foo.bak$ ls -a
. .. foo

```

```
lqzhou@LAPTOP-BOMFC06B:~$ rm -rf /home/lqzhou/foo.bak
lqzhou@LAPTOP-BOMFC06B:~$ ls
a.tar.gz  bar.txt  foo  hello.c  linux-test-report  readme
lqzhou@LAPTOP-BOMFC06B:~$ |
```

6、查看文件 /etc/adduser.conf 的前 3 行内容，查看文件 /etc/adduser.conf 的最后 5 行内容。

```
lqzhou@LAPTOP-BOMFC06B:~$ head -3 /etc/adduser.conf
# /etc/adduser.conf: 'adduser' configuration.
# See adduser(8) and adduser.conf(5) for full documentation.

lqzhou@LAPTOP-BOMFC06B:~$ |
```

```
lqzhou@LAPTOP-BOMFC06B:~$ tail -5 /etc/adduser.conf
# check user and group names also against this regular expression.
#NAME_REGEX="^[a-z][-a-z0-9_]*\$"

# use extrausers by default
#USE_EXTRAUSERS=1
lqzhou@LAPTOP-BOMFC06B:~$ |
```

7、分屏查看文件/etc/adduser.conf 的内容。

```
# /etc/adduser.conf: 'adduser' configuration.
# See adduser(8) and adduser.conf(5) for full documentation.

# The DSHELL variable specifies the default login shell on your
# system.
DSHELL=/bin/bash

# The DHOME variable specifies the directory containing users' home
# directories.
DHOME=/home

# If GROUPTHOMES is "yes", then the home directories will be created as
# /home/groupname/user.
GROUPTHOMES=no

# If LETTERHOMES is "yes", then the created home directories will have
# an extra directory - the first letter of the user name. For example:
# /home/u/user.
LETTERHOMES=no

# The SKEL variable specifies the directory containing "skeletal" user
# files; in other words, files such as a sample .profile that will be
# copied to the new user's home directory when it is created.
SKEL=/etc/skel

# FIRST_SYSTEM_[GU]ID to LAST_SYSTEM_[GU]ID inclusive is the range for UIDs
# for dynamically allocated administrative and system accounts/groups.
# Please note that system software, such as the users allocated by the base-passwd
# package, may assume that UIDs less than 100 are unallocated.
/etc/adduser.conf
```

```
lqzhou@LAPTOP-BOMFC06B:~$ less /etc/adduser.conf
lqzhou@LAPTOP-BOMFC06B:~$ |
```

8、使用命令cat用输出重定向在/home/lqzhou目录下创建文件bar.txt，文件内容为：

```
google 110 5000
baidu 100 5000
guge 50 3000
sohu 100 4500
```

```
lqzhou@LAPTOP-BOMFC06B:~$ cat > /home/lqzhou/bar.txt << EOF
google 110 5000
baidu 100 5000
guge 50 3000
sohu 100 4500
EOF
lqzhou@LAPTOP-BOMFC06B:~$ ls
a.tar.gz bar.txt hello.c linux-test-report readme
```

9. 第1列为公司名称，第2列为公司人数，第3列为员工平均工资。
利用sort命令完成下列排序：

(1) 按公司字母顺序排序

```
lqzhou@LAPTOP-BOMFC06B:~$ sort bar.txt -k 1,1
baidu 100 5000
google 110 5000
guge 50 3000
sohu 100 4500
```

(2) 按公司人数排序

```
lqzhou@LAPTOP-BOMFC06B:~$ sort bar.txt -k 2n,2
guge 50 3000
baidu 100 5000
sohu 100 4500
google 110 5000
```

(3) 按公司人数排序，人数相同的按照员工平均工资升序排序

```
lqzhou@LAPTOP-BOMFC06B:~$ sort -n -t ' ' bar.txt -k 2,2 -k 3,3
guge 50 3000
sohu 100 4500
baidu 100 5000
google 110 5000
```

(4) 按员工工资降序排序，如工资相同，则按公司人数升序排序

```
lqzhou@LAPTOP-BOMFC06B:~$ sort -n -t ' ' bar.txt -k 3r,3 -k 2,2
baidu 100 5000
google 110 5000
sohu 100 4500
guge 50 3000
```

(5) 从公司英文名称的第2个字母开始进行排序。

```
lqzhou@LAPTOP-BOMFC06B:~$ sort -t ' ' bar.txt -k 1.2,1.2
baidu 100 5000
google 110 5000
sohu 100 4500
guge 50 3000
```

四、 实验过程分析与讨论

`sort [-bcdfimMnr][--o<输出文件>][--t<分隔字符>][+<起始栏位>-<结束栏位>][--help][--version][文件][--k field1[,field2]]`

sort 默认 `ascii` 排序，同时从左往右比较，所以需要注意的是排序数字的时候要加 `n`。否则位数不同的数字会出现排序错误。

五、指导教师意见

指导教师签字：卢洋

实验报告

实验名称	实验二 Linux 常用命令（二）		
实验教室	丹青 922	实验日期	2023 年 4 月 10 日
学 号	2021213277	姓 名	周陆琦
专业班级	计算机科学与技术 05 班		
指导教师	卢洋		

东北林业大学
信息与计算机科学技术实验中心

一、 实验目的

- 1、掌握Linux下查找文件和统计文件行数、字数和字节数命令：find, wc
- 2、掌握Linux下文件打包命令：tar
- 3、掌握Linux下符号链接和文件比较命令：ln, comm, diff
- 4、掌握 Linux 的文件权限管理命令：chmod

二、 实验环境

- (1) 计算机的硬件配置 PC 系列微机。
- (2) 计算机的软件配置 VMware 虚拟机软件及 Ubuntu 虚拟机。

三、 实验内容及结果

1. 查找指定文件

- (1) 在用户目录下新建目录baz，在baz下新建文件qux，并写任意几行内容

```
lqzhou@LAPTOP-BOMFC06B:~$ mkdir -p baz
lqzhou@LAPTOP-BOMFC06B:~$ cd baz
lqzhou@LAPTOP-BOMFC06B:~/baz$ cat > qux << EOF
> hello,world
> this is a test
> EOF
```

- (2) 在用户目录下查找文件qux，并显示该文件位置信息

```
lqzhou@LAPTOP-BOMFC06B:~$ find -name qux
./baz/qux
```

- (3) 统计文件qux中所包含内容的行数，字数和字节数

```
lqzhou@LAPTOP-BOMFC06B:~/baz$ wc qux
 2  5 27 qux
```

- (4) 在用户目录下查找文件qux，并删除该文件

```
lqzhou@LAPTOP-BOMFC06B:~$ find -name qux
./baz/qux
lqzhou@LAPTOP-BOMFC06B:~$ rm ./baz/qux
lqzhou@LAPTOP-BOMFC06B:~$ find -name qux
```

- (5) 查看文件夹baz内容，看一下是否删除了文件qux

```
lqzhou@LAPTOP-BOMFC06B:~$ find baz
baz
```

2. 文件打包

- (1) 在用户目录下新建目录path1，在path1下新建文件file1和file2

```
lqzhou@LAPTOP-BOMFC06B:~$ mkdir path1
lqzhou@LAPTOP-BOMFC06B:~$ cd path1
lqzhou@LAPTOP-BOMFC06B:~/path1$ touch file1 file2
lqzhou@LAPTOP-BOMFC06B:~/path1$ cd
lqzhou@LAPTOP-BOMFC06B:~$ find path1
path1
path1/file1
path1/file2
```

- (2) 在用户目录下新建目录path2，在path2下新建文件file3

```
lqzhou@LAPTOP-BOMFC06B:~$ mkdir path2
lqzhou@LAPTOP-BOMFC06B:~$ cd path2
lqzhou@LAPTOP-BOMFC06B:~/path2$ touch file3
lqzhou@LAPTOP-BOMFC06B:~/path2$ ls
file3
```

- (3) 在用户目录下新建文件file4

```
lqzhou@LAPTOP-BOMFC06B:~$ touch file4
lqzhou@LAPTOP-BOMFC06B:~$ find file4
file4
```

- (4) 在用户目录下对文件夹path1和file4进行打包，生成文件package.tar

```
lqzhou@LAPTOP-BOMFC06B:~$ tar -cvf package.tar file4 path1
file4
path1/
path1/file1
path1/file2
lqzhou@LAPTOP-BOMFC06B:~$ ls
Linux实验1by周陆琦2021213277.doc  f2                maketest          pic
baz                                file4             package.tar       shell
c                                  hello.sh          path1             test
c.tar.gz                          linux-test-report path2             test.txt
```

(5) 查看包package.tar的内容

```
lqzhou@LAPTOP-BOMFC06B:~$ tar -tvf package.tar
-rw-r--r-- lqzhou/lqzhou      0 2023-04-10 11:58 file4
drwxr-xr-x lqzhou/lqzhou      0 2023-04-10 11:44 path1/
-rw-r--r-- lqzhou/lqzhou      0 2023-04-10 11:44 path1/file1
-rw-r--r-- lqzhou/lqzhou      0 2023-04-10 11:44 path1/file2
```

(6) 向包package.tar里添加文件夹path2的内容

```
lqzhou@LAPTOP-BOMFC06B:~$ tar -rvf package.tar path2

lqzhou@LAPTOP-BOMFC06B:~$ tar -tvf package.tar
drwxr-xr-x lqzhou/lqzhou      0 2023-04-10 11:44 path1/
-rw-r--r-- lqzhou/lqzhou      0 2023-04-10 11:44 path1/file1
-rw-r--r-- lqzhou/lqzhou      0 2023-04-10 11:44 path1/file2
-rw-r--r-- lqzhou/lqzhou      0 2023-04-10 11:58 file4
drwxr-xr-x lqzhou/lqzhou      0 2023-04-10 11:46 path2/
-rw-r--r-- lqzhou/lqzhou      0 2023-04-10 11:46 path2/file3
```

(7) 将包package.tar复制到用户目录下的新建文件夹path3中

```
lqzhou@LAPTOP-BOMFC06B:~$ mkdir path3
lqzhou@LAPTOP-BOMFC06B:~$ cp package.tar ./path3
lqzhou@LAPTOP-BOMFC06B:~$ find path3
path3
path3/package.tar
```

(8) 进入path3文件夹，并还原包package.tar的内容

```
lqzhou@LAPTOP-BOMFC06B:~$ cd path3
lqzhou@LAPTOP-BOMFC06B:~/path3$ tar -xvf package.tar
path1/
path1/file1
path1/file2
file4
path2/
path2/file3
lqzhou@LAPTOP-BOMFC06B:~/path3$ ls
file4  package.tar  path1  path2
```

3. 符号链接内容

(1) 新建文件foo.txt, 内容为123

```
lqzhou@LAPTOP-BOMFC06B:~$ cat > foo.txt << EOF
> 123
> EOF
lqzhou@LAPTOP-BOMFC06B:~$ cat foo.txt
123
```

(2) 建立foo.txt的硬链接文件bar.txt, 并比较bar.txt的内容和foo.txt是否相同, 要求用comm或diff命令

```
lqzhou@LAPTOP-BOMFC06B:~$ ln foo.txt bar.txt
lqzhou@LAPTOP-BOMFC06B:~$ comm foo.txt bar.txt
123
lqzhou@LAPTOP-BOMFC06B:~$ diff foo.txt bar.txt
```

(3) 查看foo.txt和bar.txt的i节点号(inode)是否相同

```
lqzhou@LAPTOP-BOMFC06B:~$ ls -li foo.txt bar.txt
37436 bar.txt 37436 foo.txt
```

(4) 修改bar.txt的内容为abc, 然后通过命令判断foo.txt与bar.txt是否相同

```
lqzhou@LAPTOP-BOMFC06B:~$ vim bar.txt
lqzhou@LAPTOP-BOMFC06B:~$ comm foo.txt bar.txt
abc
```

(5) 删除foo.txt文件, 然后查看bar.txt文件的inode及内容

```
lqzhou@LAPTOP-BOMFC06B:~$ rm foo.txt
lqzhou@LAPTOP-BOMFC06B:~$ cat bar.txt
abc
lqzhou@LAPTOP-BOMFC06B:~$ ls -li bar.txt
37436 bar.txt
```

- (6) 创建文件bar.txt的符号链接文件baz.txt，然后查看bar.txt和bat.txt的inode号，并观察是否相同，比较bar.txt和baz.txt的文件内容是否相同

```
lqzhou@LAPTOP-BOMFC06B:~$ ln -s bar.txt baz.txt
lqzhou@LAPTOP-BOMFC06B:~$ comm bar.txt baz.txt
abc
lqzhou@LAPTOP-BOMFC06B:~$ diff bar.txt baz.txt
lqzhou@LAPTOP-BOMFC06B:~$ ls -li bar.txt baz.txt
37436 bar.txt 37573 baz.txt
```

- (7) 删除bar.txt，查看文件baz.txt，观察系统给出什么提示信息

```
lqzhou@LAPTOP-BOMFC06B:~$ rm bar.txt
lqzhou@LAPTOP-BOMFC06B:~$ cat baz.txt
cat: baz.txt: No such file or directory
```

4. 权限管理

- (1) 新建文件qux.txt

```
lqzhou@LAPTOP-BOMFC06B:~$ touch qux.txt
lqzhou@LAPTOP-BOMFC06B:~$ ls -al qux.txt
-rw-r--r-- 1 lqzhou lqzhou 0 Apr 10 13:01 qux.txt
```

- (2) 为文件qux.txt增加执行权限(所有用户都可以执行)

```
lqzhou@LAPTOP-BOMFC06B:~$ chmod +x qux.txt
lqzhou@LAPTOP-BOMFC06B:~$ ls -al qux.txt
-rwxr-xr-x 1 lqzhou lqzhou 0 Apr 10 13:01 qux.txt
```

四、 实验过程分析与讨论

`tar -zcvf` 对文件打包并压缩 `z` 是压缩命令, `-zxvf` 是解压并拿出文件, `-cvf` 是打包, `-xvf` 是解包命令 `-rvf` 是打包 `olderpkg newfile`

五、 指导教师意见

指导教师签字：卢洋

实验报告

实验名称	实验三 vim 编辑器及 gcc 编译器的使用		
实验教室	丹青 922	实验日期	2023 年 4 月 14 日
学 号	2021213277	姓 名	周陆琦
专业班级	计算机科学与技术 05 班		
指导教师	卢洋		

东北林业大学
信息与计算机科学技术实验中心

一、 实验目的

掌握 vim 编辑器及 gcc 编译器的使用方法

二、 实验环境

- (1) 计算机的硬件配置 PC 系列微机。
- (2) 计算机的软件配置 VMware 虚拟机软件及 Ubuntu 虚拟机。

三、 实验内容及结果

1. vim 编辑器和 gcc 编译器的简单使用：

- (1) 在用户目录下新建一个目录，命名为 workspace1 ；

```
lqzhou@LAPTOP-BOMFC06B:~$ mkdir workspace1
```

- (2) 进入目录 workspace1 ；

```
lqzhou@LAPTOP-BOMFC06B:~$ cd workspace1
lqzhou@LAPTOP-BOMFC06B:~/workspace1$ pwd
/home/lqzhou/workspace1
```

- (3) 在 workspace1 下用 vim 编辑器新建一个 c 语言程序文件，文件名为 test.c ， 内容为：


```
#include <stdio.h>

int main( )
{
    printf("hello world!\n");
    return 0;
}
```

```
#include <stdio.h>

int main ()
{
    printf("hello world!\n");
    return 0;
}
```

(4) 保存 test.c 的内容，并退出；

```
:wq|
```

(5) 编译 test.c 文件，生成可执行文件 test，并执行，查看执行结果。

```
lqzhou@LAPTOP-BOMFC06B:~/workspace1$ gcc test.c -o test
lqzhou@LAPTOP-BOMFC06B:~/workspace1$ ./test
hello world!
```

2. vim 编辑器的详细使用：

(1) 在用户目录下创建一个名为 workspace2 的目录；

```
lqzhou@LAPTOP-BOMFC06B:~$ mkdir workspace2
lqzhou@LAPTOP-BOMFC06B:~$ find workspace2
workspace2
```

(2) 进入 workspace2 目录；

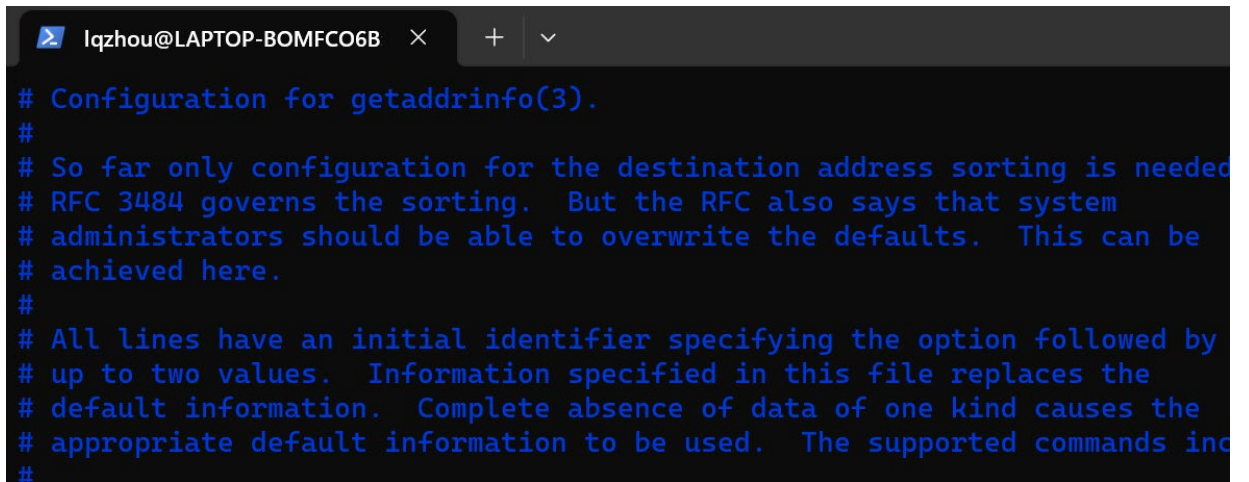
```
lqzhou@LAPTOP-BOMFC06B:~$ cd workspace2
lqzhou@LAPTOP-BOMFC06B:~/workspace2$ |
```

(3) 使用以下命令：

将文件 /etc/gai.conf 的内容复制到当前目录下的新建文件 gai.conf 中；

```
lqzhou@LAPTOP-BOMFC06B:~/workspace2$ cp /etc/gai.conf ./gai.conf
lqzhou@LAPTOP-BOMFC06B:~/workspace2$ ls
gai.conf
```

(4) 使用 vim 编辑当前目录下的 gai.conf ；



```
lqzhou@LAPTOP-BOMFC06B × + v
# Configuration for getaddrinfo(3).
#
# So far only configuration for the destination address sorting is needed
# RFC 3484 governs the sorting.  But the RFC also says that system
# administrators should be able to overwrite the defaults.  This can be
# achieved here.
#
# All lines have an initial identifier specifying the option followed by
# up to two values.  Information specified in this file replaces the
# default information.  Complete absence of data of one kind causes the
# appropriate default information to be used.  The supported commands inc
#
```

(5) 将光标移到第 18 行；

用 ngg 此时 n=18



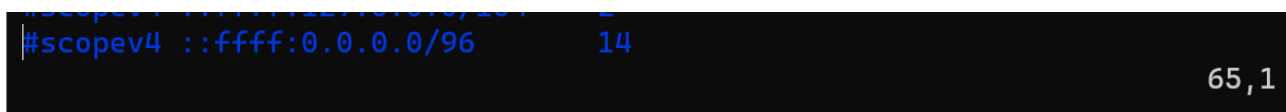
18,1

(6) 复制该行内容；

用 yy 复制行

(7) 将光标移到最后一行行首；

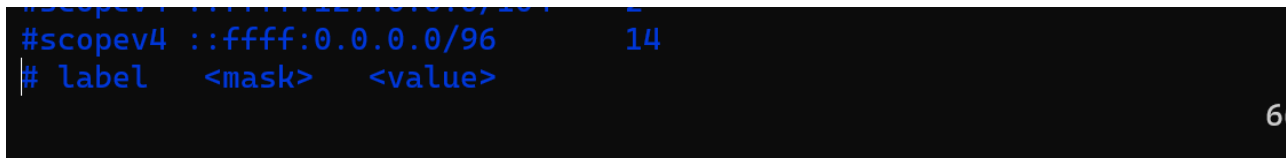
用 G 直接跳到末行行首



```
lqzhou@LAPTOP-BOMFC06B × + v
#scopev4 ::ffff:0.0.0.0/96 14
65,1
```

(8) 粘贴复制行的内容；

用 p 命令即可粘贴



```
lqzhou@LAPTOP-BOMFC06B × + v
#scopev4 ::ffff:0.0.0.0/96 14
# label <mask> <value>
66
```

(9) 撤销第 8 步的动作；

```
#scopev4 ::ffff:0.0.0.0/96 14
~
1 line less; before #1 51 seconds ago 65,1
```

(10) 存盘但不退出;

用:w 即可

```
"gai.conf" 65L, 2584B written
```

(11) 将光标移到首行;

用 gg 移到首行

```
1,1 | # Configuration for getaddrinfo(3).
    | #
```

(12) 插入模式下输入 "Hello, this is vim world!" ;

```
# "Hello, this is vim world!"
# Configuration for getaddrinfo(3).
#
```

(13) 删除字符串 "this" ;

可以用 x 删除或者在插入模式下删除

```
# "Hello, is vim world!"
```

(14) 强制退出 vim , 不存盘

```
:q!
```

四、 实验过程分析与讨论

Vim 命令比较多，难以记住，因此使用的时候有些需要查询。

五、指导教师意见

指导教师签字：卢洋

实验报告

实验名称	实验四 用户和用户组管理		
实验教室	丹青 922	实验日期	2023 年 4 月 1 日
学 号	2021213277	姓 名	周陆琦
专业班级	计算机科学与技术 05 班		
指导教师	卢洋		

东北林业大学
信息与计算机科学技术实验中心

一、 实验目的

1. 掌握用户管理命令，包括命令 `useradd` 、 `usermod` 、 `userdel` 、 `newusers` ；
2. 掌握用户组管理命令，包括命令 `groupadd` 、 `groupdel` 、 `gpasswd` ；
3. 掌握用户和用户组维护命令，包括命令 `passwd` 、 `su` 、 `sudo` 。

二、 实验环境

- (1) 计算机的硬件配置 PC 系列微机。
- (2) 计算机的软件配置 VMware 虚拟机软件及 Ubuntu 虚拟机。

三、 实验内容及结果

1. 创建一个名为 `foo` ，描述信息为 `bar` ，登录 `shell` 为 `/bin/sh` ，家目录为 `/home/foo` 的用户，并设置登陆口令为 `123456` ；

```
root@LAPTOP-BOMFC06B:~# useradd -c 'bar' -s /bin/sh -d /home/foo -m foo
root@LAPTOP-BOMFC06B:~# passwd foo
New password:
Retype new password:
passwd: password updated successfully
```

```
lqzhou:x:1000:1000:,,,:/home/lqzhou:/bin/bash
foo:x:1001:1001:bar:/home/foo:/bin/sh
```

2. 使用命令从 `root` 用户切换到用户 `foo` ，修改 `foo` 的 `UID` 为 `2000` ，其 `shell` 类型为 `/bin/csh` ；

```
root@LAPTOP-BOMFC06B:~# su foo
$ usermod -u 2000 -s /bin/csh foo
usermod: user foo is currently used by process 66
$ sudo usermod -u 2000 -s /bin/csh foo
[sudo] password for foo:
usermod: user foo is currently used by process 66
```

一直有问题，我实在不知道怎么在自己用户这里修改自己了。

```
root@LAPTOP-BOMFC06B:~# su foo
$
```

```
root@LAPTOP-BOMFC06B:~# usermod -u 2000 -s /bin/csh foo
```

```
lqzhou:x:2000:1001:bar:/home/foo:/bin/csh
```

只能去 root 那去改，我用 sudo 也不太行，因为一直显示被占用

3. 从用户 foo 切换到 root ；

```
# su root
```

4. 删除 foo 用户，并在删除该用户的同时一并删除其家目录；

```
root@LAPTOP-BOMFC06B:~# userdel -r foo
```

```
root@LAPTOP-BOMFC06B:/home# ls
lqzhou
```

5. 使用命令 newusers 批量创建用户，并使用命令 chpasswd 为这些批量创建的用户设置密码（密码也需要批量设置），查看 /etc/passwd 文件检查用户是否创建成功；

```

root@LAPTOP-BOMFC06B:~# cat > user.txt << EOF
> user001::600:100:user:/home/user001:/bin/bash
user002::601:100:user:/home/user002:/bin/bash
user003::602:100:user:/home/user003:/bin/bash
user004::603:100:user:/home/user004:/bin/bash
user005::604:100:user:/home/user005:/bin/bash
user006::605:100:user:/home/user006:/bin/bash
> EOF
root@LAPTOP-BOMFC06B:~# cat user.txt
user001::600:100:user:/home/user001:/bin/bash
user002::601:100:user:/home/user002:/bin/bash
user003::602:100:user:/home/user003:/bin/bash
user004::603:100:user:/home/user004:/bin/bash
user005::604:100:user:/home/user005:/bin/bash
user006::605:100:user:/home/user006:/bin/bash

```

先建立一个 user.txt 保存要创建的用户

```

root@LAPTOP-BOMFC06B:~# newusers < user.txt

```

取消 shadow password 功能并创建密码文件

```

root@LAPTOP-BOMFC06B:~# pwunconv

```

```

root@LAPTOP-BOMFC06B:~# cat > passwd.txt << EOF
> user001:123456
user002:123456
user003:123456
user004:123456
user005:123456
user006:123456
> EOF

```

写入并编码

```

root@LAPTOP-BOMFC06B:~# chpasswd < passwd.txt
root@LAPTOP-BOMFC06B:~# pwconv

```

```

lqzhou:x:1000:1000:,,,:/home/lqzhou:/bin/bash
user001:x:600:100:user:/home/user001:/bin/bash
user002:x:601:100:user:/home/user002:/bin/bash
user003:x:602:100:user:/home/user003:/bin/bash
user004:x:603:100:user:/home/user004:/bin/bash
user005:x:604:100:user:/home/user005:/bin/bash
user006:x:605:100:user:/home/user006:/bin/bash
"/etc/passwd" 33L 1702B

```

全都建立完毕，并且密码隐藏为 x

6. 创建用户组 group1 ，并在创建时设置其 GID 为 3000 ；

```
root@LAPTOP-BOMFC06B:~# groupadd -g 3000 group1
root@LAPTOP-BOMFC06B:~#
```

7. 在用户组 group1 中添加两个之前批量创建的用户；

```
root@LAPTOP-BOMFC06B:~# usermod -g 3000 user001
root@LAPTOP-BOMFC06B:~# usermod -g 3000 user002
```

8. 切换到 group1 组中的任一用户，在该用户下使用 sudo 命令查看 /etc/shadow 文件，检查上述操作是否可以执行；若不能执行，修改 sudoers 文件使得该用户可以查看文件 /etc/shadow 的内容。

```
user001@LAPTOP-BOMFC06B:~$ sudo vim /etc/shadow
user001 is not in the sudoers file. This incident will be reported.
```

不具备 sudo 权限 因此不可以执行此命令

```
# User privilege specification
root    ALL=(ALL:ALL) ALL
user001 ALL=(ALL:ALL) ALL
# Members of the admin group may
```

在 sudoers 中添加该用户的 sudo 权限

```
user001@LAPTOP-BOMFC06B:/root$ sudo cat /etc/shadow
[sudo] password for user001:
root:$y$j9T$PAAtnp5jUHVbGazM0zJpM0$bcdPCGBZDbNAmy5rDil
7:::
daemon*:19468:0:99999:7:::
bin*:19468:0:99999:7:::
sys*:19468:0:99999:7:::
sync*:19468:0:99999:7:::
games*:19468:0:99999:7:::
man*:19468:0:99999:7:::
```

完成查看。

四、 实验过程分析与讨论

对于用户权限的了解有一定缺失。

`sudo` 命令的使用需要在 `sudoers` 添加权限。

五、 指导教师意见

指导教师签字：卢洋

实验报告

实验名称	实验五 Shell 程序的创建及条件判断语句		
实验教室	丹青 922	实验日期	2023 年 4 月 22 日
学 号	2021213277	姓 名	周陆琦
专业班级	计算机科学与技术 05 班		
指导教师	卢洋		

东北林业大学
信息与计算机科学技术实验中心

一、 实验目的

1. 掌握Shell程序的创建过程及Shell程序的执行方法；
2. 掌握Shell变量的定义方法，及用户定义变量、参数位置等；
3. 掌握变量表达式，包括字符串比较、数字比较、逻辑测试、文件测试；
4. 掌握条件判断语句，如 if 语句、 case 语句。

二、 实验环境

- (1) 计算机的硬件配置 PC 系列微机。
- (2) 计算机的软件配置通过 wsl 安装的 ubuntu 虚拟机。

三、 实验内容及结果

1. 定义变量 foo 的值为 200 ，并将其显示在屏幕上（终端上执行）；

```
lqzhou@LAPTOP-BOMFC06B:~/shell/实验5$ foo=200
lqzhou@LAPTOP-BOMFC06B:~/shell/实验5$ echo $foo
200
```

2. 定义变量 bar 的值为 100 ，并使用 test 命令比较其值是否大于 150 ，并显示 test 命令的退出码（终端上执行）；

```
lqzhou@LAPTOP-BOMFC06B:~/shell/lib5$ bar=100
lqzhou@LAPTOP-BOMFC06B:~/shell/lib5$ test $bar -gt 150
lqzhou@LAPTOP-BOMFC06B:~/shell/lib5$ echo $?
1
```

3. 创建一个 Shell 程序，其功能为显示计算机主机名（ hostname ）和系统时间（ date ）；

```
lqzhou@LAPTOP-BOMFC06B:~/shell/lib5$ vim l3.sh
#!/bin/bash
hn=$(hostname)
da=$(date)
echo $hn
echo $da
~
~
~
```

```
lqzhou@LAPTOP-BOMFC06B:~/shell/lib5$ vim l3.sh
lqzhou@LAPTOP-BOMFC06B:~/shell/lib5$ ./l3.sh
LAPTOP-BOMFC06B
Sat Apr 22 09:27:47 CST 2023
```

4. 创建一个Shell程序，要求可以处理一个输入参数，判断该输入参数是否为水仙花数

```
#!/bin/bash
read -p "Input number:" num
num1=`expr $num % 10`
#echo $num1
num2=`expr $num / 10 % 10`
#echo $num2
num3=`expr $num / 100 % 100`
#echo $num3
num13=`expr $num1 \* $num1 \* $num1`
num23=`expr $num2 \* $num2 \* $num2`
num33=`expr $num3 \* $num3 \* $num3`
num4=`expr $num13 + $num23 + $num33`
if (( $num4 == $num ))
then
    echo "true"
else
    echo "false"
fi
```

```
lqzhou@LAPTOP-BOMFC06B:~/shell/lib5$ ./l4.sh
Input number:153
true
lqzhou@LAPTOP-BOMFC06B:~/shell/lib5$ ./l4.sh
Input number:124
false
lqzhou@LAPTOP-BOMFC06B:~/shell/lib5$ ./l4.sh
Input number:370
true
```

5. 创建一个Shell程序，输入 3 个参数，计算 3 个输入变量的和并输出；

```
lqzhou@LAPTOP-BOMFC06B:~/shell/lib5$ ./l5.sh 1 2 3  
sum = 6
```

```
#!/bin/bash  
#Sat Apr 22 10:36:17 CST 2023  
echo 'sum = ' `expr $1 + $2 + $3`
```

6. 创建一个Shell程序，输入学生成绩，给出该成绩对应的等级：
90 分以上为 A ， 80-90 为 B ， 70-80为 C ， 60-70 为 D , 小于 60 分为 E 。

```
lqzhou@LAPTOP-BOMFC06B:~/shell/lib5$ ./l6.sh  
Please input your grade:100  
level is A  
lqzhou@LAPTOP-BOMFC06B:~/shell/lib5$ ./l6.sh  
Please input your grade:55  
level is E  
lqzhou@LAPTOP-BOMFC06B:~/shell/lib5$ ./l6.sh  
Please input your grade:77  
level is C
```

```
#!/bin/bash  
#Sat Apr 22 10:54:20 CST 2023  
read -p "Please input your grade:" grade  
if (($grade < 60))  
then  
    echo "level is E"  
elif (($grade < 70))  
then  
    echo "level is D"  
elif (($grade < 80))  
then  
    echo "level is C"  
elif (($grade < 90))  
then  
    echo "level is B"  
else echo "level is A"  
fi
```

四、 实验过程分析与讨论

发现每个脚本文件都要写一个`#!/bin/bash` 我觉得太累了，写一个 shell 生成 shell 脚本的脚本，感觉速度要快一点，体验到自动化的快乐。

原本以为可以直接通过运算符计算，但是搜索后发现，需要 `expr`

五、指导教师意见

指导教师签字：卢洋

实验报告

实验名称	实验六 Shell 循环控制语句		
实验教室	丹青 922	实验日期	2023 年 4 月 23 日
学 号	2021213277	姓 名	周陆琦
专业班级	计算机科学与技术 05 班		
指导教师	卢洋		

东北林业大学
信息与计算机科学技术实验中心

一、 实验目的

1. 熟练掌握Shell循环语句： for 、 while 、 until ；
2. 熟练掌握 Shell 循环控制语句： break 、 continue 。

二、 实验环境

- (1) 计算机的硬件配置 PC 系列微机。
- (2) 计算机的软件配置通过 wsl 安装的 ubuntu 虚拟机。

三、 实验内容及结果

1. 编写一个 Shell 脚本，利用 for 循环把当前目录下的所有 *.c 文件复制到指定的目录中（如~/workspace ）；

```
lqzhou@LAPTOP-BOMFC06B:~/shell/lib6$ mkdir workspace
lqzhou@LAPTOP-BOMFC06B:~/shell/lib6$ ls
1.c 2.c 3.c creat l1.sh workspace
lqzhou@LAPTOP-BOMFC06B:~/shell/lib6$ ./l1.sh
lqzhou@LAPTOP-BOMFC06B:~/shell/lib6$ ls
creat l1.sh workspace
lqzhou@LAPTOP-BOMFC06B:~/shell/lib6$ ls workspace/
1.c 2.c 3.c
```

```
lqzhou@LAPTOP-BOMFC06B:~/shell/lib6$ cat l1.sh
#!/bin/bash
#Sat Apr 22 11:15:48 CST 2023
for file in `find *.c`
do
    mv $file ./workspace
done
```

2. 编写 Shell 脚本，利用 while 循环求前 10 个偶数之和，并

输出结果；

终于正确了，一直写的是 `$sum=` 但这是不正确的，我们应该使用 `sum` 形式，相当于对变量重新赋值，而不是对一个值去赋值，这就很奇怪了。

```
lqzhou@LAPTOP-BOMFC06B:~/shell/lib6$ ./l2.sh
110
lqzhou@LAPTOP-BOMFC06B:~/shell/lib6$ cat l2.sh
#!/bin/bash
#Sun Apr 23 22:53:19 CST 2023
count=0
sum=0
num=2
while (($count<10))
do
    sum=`expr $sum + $num`
    num=`expr $num + 2`
    let "count++"
done
echo $sum
```

3. 编写 Shell 脚本，利用 `until` 循环求 1 到 10 的平方和，并输出结果；

```
lqzhou@LAPTOP-BOMFC06B:~/shell/lib6$ ./l3.sh
385
lqzhou@LAPTOP-BOMFC06B:~/shell/lib6$ cat l3.sh
#!/bin/bash
#Sun Apr 23 23:24:14 CST 2023
count=1
sum=0
until (($count == 11))
do
    sum=`expr $sum + $count \* $count`
    let "count++"
done
echo $sum
```

4. 运行下列程序，并观察程序的运行结果。将程序中的 `---` 分别替换为 `break`、`break2`、`continue`、`continue 2`，

并观察四种情况下的实验结果。

```
#!/bin/bash

for i in a b c d; do
    echo -n $i
    for j in 1 2 3 4 5 6 7 8 9 10; do
        if [[ $j -eq 5 ]]; then
            ---
        fi
        echo -n $j
    done
    echo ' '
done
```

break 时

```
lqzhou@LAPTOP-BOMFC06B:~/shell/lib6$ ./l4.sh
a1234
b1234
c1234
d1234
```

Break 2 时

```
lqzhou@LAPTOP-BOMFC06B:~/shell/lib6$ ./l4.sh
a1234lqzhou@LAPTOP-BOMFC06B:~/shell/lib6$ |
```

直接跳过 2

个循环了

Continue

```
lqzhou@LAPTOP-BOMFC06B:~/shell/lib6$ ./l4.sh
a1234678910
b1234678910
c1234678910
d1234678910
lqzhou@LAPTOP-BOMFC06B:~/shell/lib6$ |
```

Continue 2

```
lqzhou@LAPTOP-BOMFC06B:~/shell/lib6$ ./l4.sh  
a1234b1234c1234d1234lqzhou@LAPTOP-BOMFC06B:~/shell/lib6$ |
```

跳过两次操作吧，因为换行都没啦 有点意思的

四、 实验过程分析与讨论

对 until 和 while 有一定基础了解了。

Break continue 后面 2 的意思是应该是执行 2 遍？

五、指导教师意见

指导教师签字：卢洋

实验报告

实验名称	实验七 Shell 函数		
实验教室	丹青 922	实验日期	2023 年 4 月 26 日
学 号	2021213277	姓 名	周陆琦
专业班级	计算机科学与技术 05 班		
指导教师	卢洋		

东北林业大学
信息与计算机科学技术实验中心

一、 实验目的

1. 掌握Shell函数的定义方法；
2. 掌握Shell函数的参数传递、调用和返回值；
3. 掌握Shell函数的递归调用方法；
4. 理解 Shell 函数的嵌套。

二、 实验环境

- (1) 计算机的硬件配置 PC 系列微机。
- (2) 计算机的软件配置通过 wsl 安装的 ubuntu 虚拟机。

三、 实验内容及结果

1. 编写 Shell 脚本，实现一个函数，对两个数的和进行求解，并输出结果；

```
Sum of two number is 5542  
lqzhou@LAPTOP-B0MFC06B:~/shell/lib7$ ./l1.sh  
Please input first number:1230  
Please input second number:4312  
Sum of two number is 5542
```

```
#!/bin/bash
#Wed Apr 26 18:12:43 CST 2023
function AddTwoNum()
{
    s=`expr $1 + $2`
    echo "Sum of two number is $s"
}
read -p "Please input first number:" num1
read -p "Please input second number:" num2
AddTwoNum $num1 $num2
~
```

2. 编写 Shell 脚本，在脚本中定义一个递归函数，实现 n 的阶乘的求解；

我看错了，以为是斐波那契，然后一直犹豫怎么返回参数呢，因为 `return` 只能返回 0-255，之后才了解到 `echo` 可以返回到调用的位置。

```
lqzhou@LAPTOP-BOMFC06B x + v
#!/bin/bash
#Wed Apr 26 18:33:10 CST 2023
fib()
{
    local k=$1
    if [ $k -eq 1 -o $k -eq 2 ]
    then
        echo 1
    else
        let sum=`fib $["$1"-1]`+`fib $["$1"-2]`
        echo $sum
    fi
}
read a
fib $a
~
```

```
lqzhou@LAPTOP-BOMFC06B:~/shell/lib7$ ./l2.sh
5
5
```


以上是斐波那契，以下是阶乘

```
#!/bin/bash
#Wed Apr 26 23:48:57 CST 2023
f(){
    if [ $1 -eq 0 ]
    then
        echo 1
    else
        let sum=$1*`f $["$1"-1]`
        echo $sum
    fi
}
read a
f $a
```

```
lqzhou@LAPTOP-BOMFC06B:~/shell/lib7$ ./l2.sh
5
120
```

虽然不是很明白为什么加``

下面是查询结果

shell 调用能够直接在linux终端调用的命令

能够在linux终端运行的命令在shell脚本中也能够运行
如果命令放在句首不需要加`` 或者 `$()`
如果命令放在赋值运算符的右边将命令执行的结果赋值给变量,这个时候命令就需要加``和`$()`
调用shell自己定义的函数是一样的道理,如果放句首不需要`` or `$()` 如果放赋值运算符右边就需要加`` or `$()`
注意:
调用了``相当于开了一个子shell去运行程序,然后将结果返回,子shell是可以访问父shell的变量
但是子shell生成的变量以及所作的修改不影响父shell,还有shell中默认都是全局变量,无论是写在函数内还是函数外.如

``和`$()`的作用域问题

``和`$()`是开了子shell去运行命令,然后将命令运行的结果回传,但是子shell所作的操作不会更改父shell变量数据,这里要注意一下,还有父shell不能使用除子shell返回之外的数据,但是子shell可以使用父shell的变量.

原来是增加了一个子 shell 怪不得 echo 不会输出

3. 试运行该程序，并观察程序运行结果，理解函数嵌套的含义。

```
#!/bin/bash
```

```
function first() {  
    function second() {  
        function third() {  
            echo "-3- here is in the third func."  
        }  
        echo "-2- here is in the second func."  
        third  
    }  
    echo "-1- here is in the first func."  
    second  
}  
echo "starting..."  
first
```

```
lqzhou@LAPTOP-BOMFC06B:~/shell/lib7$ ./l4.sh  
starting...  
-1- here is in the first func.  
-2- here is in the second func.  
-3- here is the third func.
```

```
#!/bin/bash  
#Thu Apr 27 15:58:50 CST 2023  
function first(){  
    function second(){  
        function third(){  
            echo "-3- here is the third func."  
        }  
        echo "-2- here is in the second func."  
        third  
    }  
    echo "-1- here is in the first func."  
    second  
}  
echo "starting..."  
first
```

通过运行结果，我觉得应该是在函数当中定义函数并且调用，顺着猜测，我去尝试在 first 外调用 second 结果如下

```
lqzhou@LAPTOP-BOMFC06B:~/shell/lib7$ ./l4.sh
starting...
./l4.sh: line 15: second: command not found
```

```
#!/bin/bash
#Thu Apr 27 15:58:50 CST 2023
function first(){
    function second(){
        function third(){
            echo "-3- here is the third func."
        }
        echo "-2- here is in the second func."
        third
    }
    echo "-1- here is in the first func."
    second
}
echo "starting..."
second
```

结果确实没有调用到 second

通过查阅资料，得到如下结论：

bash 中的函数必须在其第一次调用之前被完成。没有如 c 语言中那种函数声明的形式。即使提前定义了函数的变量也不可以。但是只要执行顺序中，函数的定义在函数的使用之前即可。

可以看出，其实函数的定义其实只是一种命令。而 bash 脚本是一种线性执行的语言。因此只要保证函数使用的流程正确，就可以将函数应用在各种可能的地方。

四、 实验过程分析与讨论

理解一些 shell 和子 shell 的关系，可以让 echo 不显示在当前 shell 的屏幕上。这对于只能返回 0-255 的 return 来说，帮助很大，能过通过递归返回很大的数值。

五、指导教师意见

指导教师签字：卢洋

实验报告

实验名称	实验八 Sed 和 awk		
实验教室	丹青 922	实验日期	2023 年 4 月 27 日
学 号	2021213277	姓 名	周陆琦
专业班级	计算机科学与技术 05 班		
指导教师	卢洋		

东北林业大学
信息与计算机科学技术实验中心

一、 实验目的

1. 掌握 sed 基本编辑命令的使用方法；
2. 掌握 sed 与Shell变量的交互方法；
3. 掌握 awk 命令的使用方法；
4. 掌握 awk 与 Shell 变量的交互方法。

二、 实验环境

- (1) 计算机的硬件配置 PC 系列微机。
- (2) 计算机的软件配置通过 wsl 安装的 ubuntu 虚拟机。

三、 实验内容及结果

1. 文件 quote.txt 的内容如下所示：

```
The honeysuckle band played all night long for only $90.  
It was an evening of splendid music and company.  
Too bad the disco floor fell through at 23:10.  
The local nurse Miss P.Neave was in attendance.
```

试使用 sed 命令实现如下功能：

```
lqzhou@LAPTOP-BOMFC06B:~/shell/lib8$ cat > quote.txt << EOF  
> The honeysuckle band played all night long for only $90.  
It was an evening of splendid music and company.  
Too bad the disco floor fell through at 23:10.  
The local nurse Miss P.Neave was in attendance.  
> EOF
```

- (1) 删除 \$ 符号；

```
lqzhou@LAPTOP-BOMFC06B:~/shell/lib8$ cat quote.txt | sed 's/\$/g'
The honeysuckle band played all night long for only 90.
It was an evening of splendid music and company.
Too bad the disco floor fell through at 23:10.
The local nurse Miss P.Neave was in attendance.
```

(2) 显示包含 music 文字的行内容及行号;

```
lqzhou@LAPTOP-BOMFC06B:~/shell/lib8$ nl quote.txt | sed -n '/music/p'
2 It was an evening of splendid music and company.
```

(3) 在第 4 行后面追加内容: "hello world!" ;

```
lqzhou@LAPTOP-BOMFC06B:~/shell/lib8$ sed -e 4a'hello world!' quote.txt
The honeysuckle band played all night long for only $90.
It was an evening of splendid music and company.
Too bad the disco floor fell through at 23:10.
The local nurse Miss P.Neave was in attendance.
hello world!
```

(4) 将文本 "The" 替换为 "Quod" ;

```
lqzhou@LAPTOP-BOMFC06B:~/shell/lib8$ sed 's/The/Quod/g' quote.txt
Quod honeysuckle band played all night long for only $90.
It was an evening of splendid music and company.
Too bad the disco floor fell through at 23:10.
Quod local nurse Miss P.Neave was in attendance.
```

(5) 将第 3 行内容修改为: "This is the third line." ;

```
lqzhou@LAPTOP-BOMFC06B:~/shell/lib8$ sed '3c This is the third line\.' quote.txt
The honeysuckle band played all night long for only $90.
It was an evening of splendid music and company.
This is the third line.
The local nurse Miss P.Neave was in attendance.
```

```
lqzhou@LAPTOP-BOMFC06B:~/shell/lib8$ nl quote.txt | sed '3c This is the third line\.'
1 The honeysuckle band played all night long for only $90.
2 It was an evening of splendid music and company.
This is the third line.
4 The local nurse Miss P.Neave was in attendance.
```

(6) 删除第 2 行内容;

```
lqzhou@LAPTOP-BOMFC06B:~/shell/lib8$ nl quote.txt | sed '2d'
1 The honeysuckle band played all night long for only $90.
3 Too bad the disco floor fell through at 23:10.
4 The local nurse Miss P.Neave was in attendance.
```

(7) 设置 Shell 变量 var 的值为 evening , 用 sed 命令查找匹配 var 变量值的行。


```
lqzhou@LAPTOP-BOMFC06B:~/shell/lib8$ var='evening'
lqzhou@LAPTOP-BOMFC06B:~/shell/lib8$ echo $var
evening

lqzhou@LAPTOP-BOMFC06B:~/shell/lib8$ set | nl - | sed -n '/evening/p'
69 var=evening
```

2. 文件 numbers.txt 的内容如下所示:

```
one : two : three
four : five : six
```

注: 每个冒号前后都有空格。

试使用 awk 命令实现如下功能: 分别以 空格 和 冒号 做分隔符, 显示第 2 列的内容, 观察两者的区别;

```
lqzhou@LAPTOP-BOMFC06B:~/shell/lib8$ cat numbers.txt
one : two : three
four : five : six
```

```
lqzhou@LAPTOP-BOMFC06B:~/shell/lib8$ awk -F ' ' '{print $2}' numbers.txt
:
:
```

```
lqzhou@LAPTOP-BOMFC06B:~/shell/lib8$ awk -F ':' '{print $2}' numbers.txt
two
five
```

3. 已知文件 foo.txt 中存储的都是数字, 且每行都包含 3 个数字, 数字之前以空格作为分隔符。试找出 foo.txt 中的所有偶数进行打印, 并输出偶数的个数

要求: 判断每行的 3 个数字是否为偶数时用循环结果, 即要求程序里包含循环和分支结构。

```
lqzhou@LAPTOP-BOMFC06B:~/shell/lib8$ cat foo.txt
2 4 3
15 46 79
9 6 7
```

```

lqzhou@LAPTOP-B0MFC06B:~/shell/lib8$ ./l3.sh
please input filename: foo.txt
even:
2
1
4
2
46
3
6
4
numbers:
0

```

在每个 if 里输出或者是在 while 里输出都有显示,但是为什么在 while 结束后就是 0 呢

```

#!/bin/bash
#Thu May 4 19:00:29 CST 2023
read -p "please input filename: " filename
echo "even:"
export num=0
cat $filename | while read line
do
    a1='echo $line | awk '{print $1}''
    a2='echo $line | awk '{print $2}''
    a3='echo $line | awk '{print $3}''
    if (($a1%2 == 0))
    then
        let num=num+1
        echo $a1
        echo $num
    fi
    if (($a2%2 == 0))
    then
        let num=num+1
        echo $a2
        echo $num
    fi
    if (($a3%2 == 0))
    then
        let num=num+1
        echo $a3
        echo $num
    fi
done
echo "numbers:"
echo $num

```

```

lqzhou@LAPTOP-BOMFC06B:~/shell/lib8$ vim l3.sh
lqzhou@LAPTOP-BOMFC06B:~/shell/lib8$ ./l3.sh
please input filename: foo.txt
even:
2
4
46
6
numbers:
4

```

原来是，`cat file |` 又开了一个子 shell，导致产生了局部变量
最后换成 `done < file` 的形式就解决了。

```

read -p "please input filename: " filename
echo "even:"
export num=0
while read line
do
    a1='echo $line | awk '{print $1}''
    a2='echo $line | awk '{print $2}''
    a3='echo $line | awk '{print $3}''
    if (($a1%2 == 0))
    then
        let num=$num+1
        echo $a1
        #echo $num
    fi
    if (($a2%2 == 0))
    then
        let num=$num+1
        echo $a2
        #echo $num
    fi
    if (($a3%2 == 0))
    then
        let num=$num+1
        echo $a3
        #echo $num
    fi
done < $filename
echo "numbers:"
echo $num

```

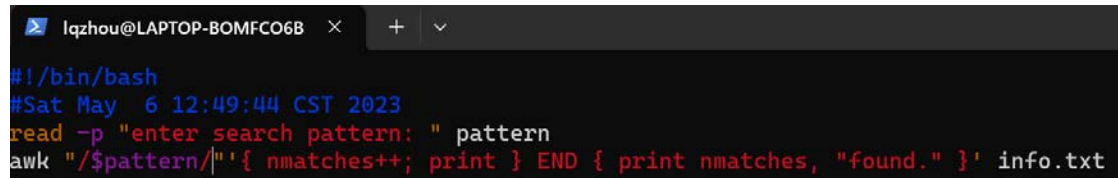
4. 脚本的内容如下所示：

```
#!/bin/bash

read -p "enter search pattern: " pattern

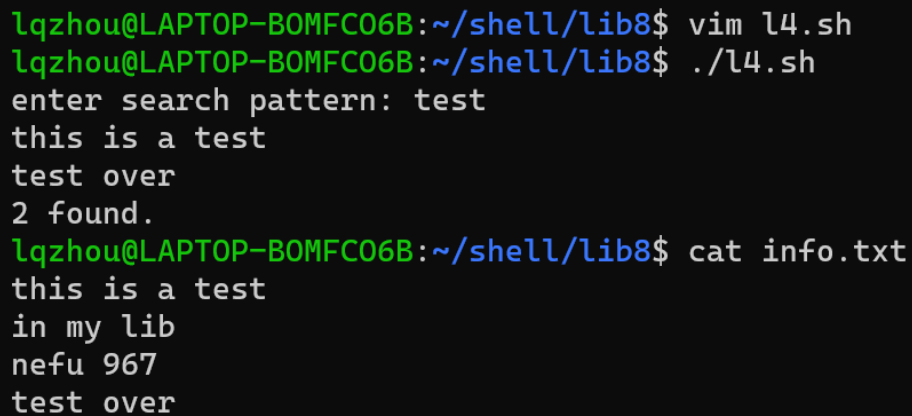
awk "/$pattern/" '{ nmatches++; print } END { print nmatches, "found." }' info.txt
```

试运行该脚本，并理解该脚本实现的功能。

A terminal window titled 'lqzhou@LAPTOP-BOMFC06B' with a dark background. It shows the execution of the script: the prompt is '#!/bin/bash', followed by the date and time '#Sat May 6 12:49:44 CST 2023'. Then, the script's input prompt 'enter search pattern: ' is shown, followed by the user input 'pattern'. Finally, the script's output is displayed: '2 found.'.

```
lqzhou@LAPTOP-BOMFC06B x + v
#!/bin/bash
#Sat May 6 12:49:44 CST 2023
read -p "enter search pattern: " pattern
awk "/$pattern/" '{ nmatches++; print } END { print nmatches, "found." }' info.txt
```

该脚本对 info.txt 查找 pattern 只要该行存在 pattern 就输出该行且计数，在最后输出总数。如下所示：

A terminal window showing the script being edited and then executed. The user runs 'vim l4.sh' and then './l4.sh'. The script prompts for a search pattern, and the user enters 'test'. The script then outputs the lines of 'info.txt' that contain 'test' and the total count '2 found.'. Finally, the user runs 'cat info.txt' to show the full content of the file.

```
lqzhou@LAPTOP-BOMFC06B:~/shell/lib8$ vim l4.sh
lqzhou@LAPTOP-BOMFC06B:~/shell/lib8$ ./l4.sh
enter search pattern: test
this is a test
test over
2 found.
lqzhou@LAPTOP-BOMFC06B:~/shell/lib8$ cat info.txt
this is a test
in my lib
nefu 967
test over
```

四、 实验过程分析与讨论

理解一些 shell 和子 shell 的关系，可以让 echo 不显示在当前 shell 的屏幕上。这对于只能返回 0-255 的 return 来说，帮助很大，能过通过递归返回很大的数值。

五、指导教师意见

指导教师签字：卢洋