

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 5 月 13 日
学 号	2021213140	姓 名	张贺
专业班级	计算机科学与技术 01 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

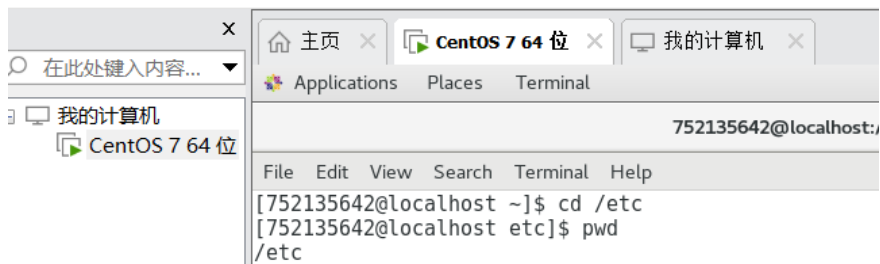
- 1、掌握Linux下文件和目录操作命令：cd、ls、mkdir、rmdir、rm
- 2、掌握Linux下文件信息显示命令：cat、more、head、tail
- 3、掌握Linux下文件复制、删除及移动命令：cp、mv
- 4、掌握 Linux 的文件排序命令：sort

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 VMware 虚拟机软件及 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



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

```
[752135642@localhost etc]$ ls -a /home/752135642
.          .bash_profile Desktop    foo        .ICEauthority Pictures  try.txt
..         .bashrc      Documents hhh        .local     Public   Videos
.bash_history .cache      Downloads hhh1.txt   .mozilla   Templates
.bash_logout .config     .esd_auth hhh.txt    Music      test
[752135642@localhost etc]$ S
```

3、使用命令创建目录/home/lyj/linux，然后删除该目录。

```
[752135642@localhost etc]$ mkdir /home/752135642/linux
[752135642@localhost etc]$ rm -rf /home/752135642/linux
[752135642@localhost etc]$
```

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

```
[752135642@localhost etc]$ cd /home/752135642
[752135642@localhost ~]$ cat > foo.txt
Hello Linux!
[752135642@localhost ~]$ cat foo.txt
Hello Linux!
```

5、使用命令创建目录/home/lyj/ak，然后将/home/lyj/abc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。

```
[752135642@localhost ~]$ mkdir /home/752135642/foo.bak
[752135642@localhost ~]$ mv /home/752135642/foo.txt /foo.bak
```

```
[752135642@localhost ~]$ rm -rf foo.bak
```

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。

```
[752135642@localhost ~]$ rm -rf foo.bak
[752135642@localhost ~]$ head -3 /etc/adduser.conf
head: cannot open '/etc/adduser.conf' for reading: No such file or directory
```

```
[752135642@localhost ~]$ tail -5 /etc/adduser.conf
```

7、分屏查看文件/etc/adduser.conf 的内容。

```
head: cannot open '/etc/adduser.conf' for reading: No such file or directory
[752135642@localhost ~]$ less /etc/adduser.conf
/etc/adduser.conf: No such file or directory
```

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

google 110 5000

baidu 100 5000

guge 50 3000

sohu 100 4500

```
[752135642@localhost ~]$ cat >/home/752135642/bar.txt
google 110 5000
baidu 100 5000
guge 50 3000
sohu 100 4500
[752135642@localhost ~]$ cat bar.txt
google 110 5000
baidu 100 5000
guge 50 3000
sohu 100 4500
[752135642@localhost ~]$
```

9. 第一列为公司名称，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

```
sohu 100 4500
guge 50 3000
google 110 5000
baidu 100 5000
[752135642@localhost ~]$ sort -r bar.txt
```

(2) 按公司人数排序

```
[752135642@localhost ~]$ sort -n bar.txt
baidu 100 5000
google 110 5000
guge 50 3000
sohu 100 4500
```

(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

```
sohu 100 4500
[752135642@localhost ~]$ sort -n -t ' ' -k 3 bar.txt
guge 50 3000
sohu 100 4500
baidu 100 5000
google 110 5000
[752135642@localhost ~]$
```

(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

```
google 110 5000
[752135642@localhost ~]$ sort -n -t ' ' -k 2 bar.txt
guge 50 3000
baidu 100 5000
sohu 100 4500
google 110 5000
[752135642@localhost ~]$
```

(5) 从公司英文名称的第2个字母开始进行排序。

```
sohu 100 4500
google 110 5000
[752135642@localhost ~]$ sort -t ' ' -k 1.2 bar.txt
baidu 100 5000
sohu 100 4500
google 110 5000
guge 50 3000
[752135642@localhost ~]$
```

四、 实验过程分析与讨论

遇到的困难在最后那个实验，排序的部分，对于多重要求和非第一行的排序命令还是不太熟悉，在查询 CSDN 之后学会了相关命令。

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 5 月 13 日
学 号	2021213140	姓 名	张贺
专业班级	计算机科学与技术 01 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

五、 实验目的

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 下符号链接命令和文件比较命令： `ln` 、 `comm` 、 `diff` ；

4. 掌握 Linux 的文件权限管理命令： `chmod`。

六、 实验环境

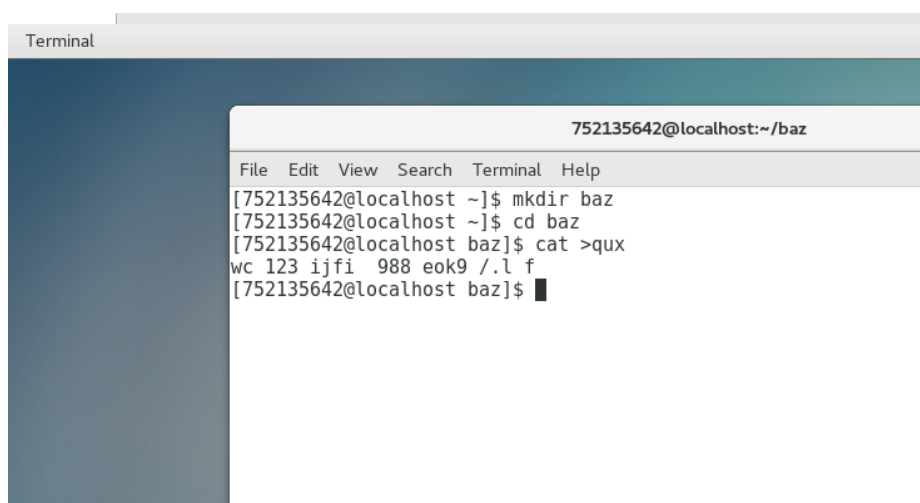
(1) 计算机的硬件配置 PC 系列微机。

(2) 计算机的软件配置 VMware 虚拟机软件及 Ubuntu 虚拟机。

七、 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` ，并写如任意几行内容；



```
Terminal
752135642@localhost:~/baz
File Edit View Search Terminal Help
[752135642@localhost ~]$ mkdir baz
[752135642@localhost ~]$ cd baz
[752135642@localhost baz]$ cat >qux
wc 123 ijfi 988 eok9 /.l f
[752135642@localhost baz]$
```

(2) 在用户目录下查找文件 `qux` ，并显示该文件位置信息；


```
[752135642@localhost baz]$ find ~ -name qux
/home/752135642/baz/qux
[752135642@localhost baz]$
```

(3) 统计文件 `qux` 中所包含内容的行数、字数和字节数;

```
[752135642@localhost baz]$ wc qux
1 7 28 qux
[752135642@localhost baz]$
```

(4) 在用户目录下查找文件 `qux` , 并删除该文件;

```
752135642@localhost baz]$ find ~ -name qux -delete
752135642@localhost baz]$ ls -l
total 0
752135642@localhost baz]$
```

(5) 查看文件夹 `baz` 内容, 看一下是否删除了文件 `qux` 。

```
752135642@localhost baz]$ find ~ -name qux -delete
752135642@localhost baz]$ ls -l
total 0
752135642@localhost baz]$
```

2. 文件打包

(1) 在用户目录下新建文件夹 `path1` , 在 `path1` 下新建文件 `file1` 和 `file2` ;

```
[752135642@localhost baz]$ ls -l
total 0
[752135642@localhost baz]$ mkdir path1
[752135642@localhost baz]$ cd path1/
[752135642@localhost path1]$ touch file1 file2
[752135642@localhost path1]$
```

(2) 在用户目录下新建文件夹 `path2` , 在 `path2` 下新建文件 `file3` ;

```
[752135642@localhost path1]$ mkdir path2
[752135642@localhost path1]$ cd path2/
[752135642@localhost path2]$ touch file3
[752135642@localhost path2]$
```

(3) 在用户目录下新建文件 `file4` ;

```
[752135642@localhost ~]$ touch file4
[752135642@localhost ~]$ ls -l
total 20
-rw-rw-r--. 1 752135642 752135642 58 May 15 18:06 bar.txt
drwxrwxr-x. 3 752135642 752135642 19 May 22 19:41 baz
drwxr-xr-x. 2 752135642 752135642 6 Mar 8 16:53 Desktop
drwxr-xr-x. 2 752135642 752135642 6 Mar 8 16:53 Documents
drwxr-xr-x. 2 752135642 752135642 6 Mar 8 16:53 Downloads
-rw-rw-r--. 1 752135642 752135642 0 May 22 19:46 file4
drwxrwxr-x. 2 752135642 752135642 6 Mar 21 15:48 foo
-rw-rw-r--. 1 752135642 752135642 13 May 15 17:26 foo.txt
-rw-rw-r--. 1 752135642 752135642 0 Mar 21 16:39 hhh
-rw-rw-r--. 1 752135642 752135642 82 Mar 21 16:40 hhh1.txt
-rw-rw-r--. 1 752135642 752135642 32 Mar 21 16:39 hhh.txt
drwxr-xr-x. 2 752135642 752135642 6 Mar 8 16:53 Music
drwxr-xr-x. 2 752135642 752135642 6 Mar 8 16:53 Pictures
drwxr-xr-x. 2 752135642 752135642 6 Mar 8 16:53 Public
drwxr-xr-x. 2 752135642 752135642 6 Mar 8 16:53 Templates
-rw-rw-r--. 1 752135642 752135642 17 Mar 30 16:03 test
-rw-r--r. 1 752135642 752135642 29 Mar 21 15:57 try.txt
-r-xr-x. 2 752135642 752135642 6 Mar 8 16:53 Videos
752135642@localhost ~]$
```

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

```
kr-xr-x. 2 752135642 752135642 6 Mar 8 16:53 Public
kr-xr-x. 2 752135642 752135642 6 Mar 8 16:53 Templates
krwxr-x. 3 752135642 752135642 17 Mar 30 16:03 test
-rw-r--r. 1 752135642 752135642 29 Mar 21 15:57 try.txt
-r-xr-x. 2 752135642 752135642 6 Mar 8 16:53 Videos
752135642@localhost ~]$ tar -cvf package.tar path1 file4
path1/
file4
```

(5) 查看包 `package.tar` 的内容；

```
[752135642@localhost ~]$ tar -cvf package.tar path1 file4
path1/
file4
```

(6) 向包 `package.tar` 里添加文件夹 `path2` 的内容；

```
752135642@localhost ~]$ tar -rvf package.tar path2
path2/
```

(7) 将包 `package.tar` 复制到用户目录下的新建文件夹 `path3` 中；

```
[752135642@localhost ~]$ tar -rvf package.tar path2
path2/
[752135642@localhost ~]$ mkdir path3
[752135642@localhost ~]$ cp package.tar path3
[752135642@localhost ~]$
```

(8) 进入 `path3` 文件夹，并还原包 `package.tar` 的内容。

```
[752135642@localhost ~]$ cd path3
[752135642@localhost path3]$ tar -xvf package.tar
path1/
file4
path2/
```

3. 符号链接内容

(1) 新建文件 `foo.txt` ，内容为 `123` ；

```
11111111
path2/
[752135642@localhost path3]$ echo "123" > foo.txt
[752135642@localhost path3]$
```

(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;

```
[752135642@localhost path3]$ ln foo.txt bar.txt
[752135642@localhost path3]$ comm foo.txt bar.txt
123
[752135642@localhost path3]$ diff foo.txt bar.txt
[752135642@localhost path3]$
```

(3) 查看 foo.txt 和 bar.txt 的 i 节点号 (inode) 是否相同;

```
123
[752135642@localhost path3]$ diff foo.txt bar.txt
[752135642@localhost path3]$ ls -l -i foo.txt bar.txt
17638714 -rw-rw-r--. 2 752135642 752135642 4 May 22 20:11 bar.txt
17638714 -rw-rw-r--. 2 752135642 752135642 4 May 22 20:11 foo.txt
[752135642@localhost path3]$
```

(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;

```
17638714 -rw-rw-r--. 2 752135642 752135642 4 May 22 20:11 foo.txt
[752135642@localhost path3]$ echo "abc" >bar.txt
[752135642@localhost path3]$ diff foo.txt bar.txt
[752135642@localhost path3]$
```

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

```
[752135642@localhost path3]$ diff foo.txt bar.txt
[752135642@localhost path3]$ rm foo.txt
[752135642@localhost path3]$ ls -l -i bar.txt
17638714 -rw-rw-r--. 1 752135642 752135642 4 May 22 20:18 bar.txt
[752135642@localhost path3]$ cat bar.txt
abc
-----
```

(6) 创建文件 bar.txt 的符号链接文件 baz.txt , 然后查看 bar.txt 和 baz.txt 的 inode 号, 并观察两者是否相同, 比较 bar.txt 和 baz.txt 的文件内容是否相同;

```
[752135642@localhost path3]$ ln -s bar.txt baz.txt
[752135642@localhost path3]$ ls -i bar.txt baz
ls: cannot access baz: No such file or directory
17638714 bar.txt
[752135642@localhost path3]$ ls -i bar.txt baz.txt
17638714 bar.txt 17638716 baz.txt
[752135642@localhost path3]$ diff bar.txt baz.txt
[752135642@localhost path3]$
```

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

```
52135642@localhost path3]$ rm bar.txt baz
52135642@localhost path3]$ rm bar.txt
52135642@localhost path3]$ cd baz.txt
sh: cd: baz.txt: No such file or directory
52135642@localhost path3]$ █
```

4. 权限管理

(1) 新建文件 qux.txt ；

(2) 为文件 qux.txt 增加执行权限（所有用户都可以执行）

八、 实验过程分析与讨论

本次实验进行了 Linux 系统中的查找、压缩、链接文件和修改权限命令的练习。。

```
[752135642@localhost path3]$ touch qux.txt  
[752135642@localhost path3]$ chmod +x qux.txt  
[752135642@localhost path3]$ ls -l  
total 12  
-rwxr-xr-x 1 752135642 752135642 7 May 22 20
```

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 5 月 13 日
学 号	2021213140	姓 名	张贺
专业班级	计算机科学与技术 01 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

九、 实验目的

1. 掌握 vim 编辑器及 gcc 编译器的使用方法

十、 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。

十一、实验过程分析与讨论

本次实验进行了 Linux 系统中的查找、压缩、链接文件和修改权限命令的练习。。

十二、实验过程分析与讨论

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 ；

```
752135642@localhost:~/workspace1
File Edit View Search Terminal Help
[752135642@localhost ~]$ mkdir workspace1
[752135642@localhost ~]$ cd workspace1/
[752135642@localhost workspace1]$
```

(2) 进入目录 workspace1 ；

```
752135642@localhost:~/workspace1
File Edit View Search Terminal Help
[752135642@localhost ~]$ mkdir workspace1
[752135642@localhost ~]$ cd workspace1/
[752135642@localhost 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 ，

[illegible]

- ```
[752135642@localhost workspace1]$ vim test.c
[752135642@localhost workspace1]$ gcc test.c -o test
[752135642@localhost workspace1]$ ls
test test.c
[752135642@localhost workspace1]$./test
hello world!
[752135642@localhost workspace1]$
```

- ```
[752135642@localhost workspace1]$ vim test.c
[752135642@localhost workspace1]$ gcc test.c -o test
[752135642@localhost workspace1]$ ls
test  test.c
[752135642@localhost workspace1]$ ./test
hello world!
[752135642@localhost workspace1]$
```

(1) 在用户目录下创建一个名为 workspace2 的目录;

```
netto world!  
[752135642@localhost workspace1]$ mkdir workspace2  
[752135642@localhost workspace1]$
```

(2) 进入 workspace2 目录;

```
[752135642@localhost workspace1]$ mkdir workspace2  
[752135642@localhost workspace1]$ cd workspace2  
[752135642@localhost 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;

```
[752135642@localhost workspace2]$ cat /etc/gai.conf > ./gai.conf
```

(4) 使用 vim 编辑当前目录下的 gai.conf ;

```
[752135642@localhost workspace2]$ vim gai.conf  
[752135642@localhost workspace2]$
```

(5) 将光标移到第 18 行;

Set -nu

```
13 f  
14 sf  
15 :set nu  
16  
17 :set nu  
18  
19  
20  
21
```

(6) 复制该行内容;

```
17 :set nu  
18 dsfsff  
19
```

(7) 将光标移到最后一行行首;

```
27  
28 :wq  
29  
~  
~  
:$
```

(8) 粘贴复制行的内容;

Yy

(8) 撤销第 8 步的动作;

u

(9) 存盘但不退出;

A screenshot of a terminal window. The background is dark blue/black. On the left side, there is a vertical bar with a light blue gradient. In the main area, there are three lines of text: a light blue prompt character resembling a tilde (~), followed by another light blue prompt character, and finally a white prompt character followed by the letter 'W'.

(10) 将光标移到首行;

```
28 :wq
29 █
~
":gg" [New] 29L, 93C written
```

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

```
19
20
21
22 hello,this is vim world
```

(12) 删除字符串 "this" ;

```
18 us1311  
19  
20  
21  
22 hello, is vim world
```

~
~
~
~
~
~
~
~
~
~

```
:%s/this//g
```

(13) 强制退出 vim，不存盘

~
~
~
~
~
:q!

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 用户与用户组管理		
实验教室	丹青 922	实验日期	2023 年 5 月 13 日
学 号	2021213140	姓 名	张贺
专业班级	计算机科学与技术 01 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

十三、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` 。

十四、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。

十五、实验过程分析与讨论

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` ；

```
752135642 is not in the sudoers file. This incident will be reported.
[752135642@localhost ~]$ su root
Password:
[root@localhost 752135642]# sudo useradd -c "bar" -s /bin/sh -d /home/foo -p $
penssl passwd -l "123456") foo
[root@localhost 752135642]#
```

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` ；

```
penssl passwd -l "123456") foo
[root@localhost 752135642]# usermod -u 2000 -s /bin/csh foo
[root@localhost 752135642]# su - foo
```

3. 从用户 `foo` 切换到 `root` ；

```
[root@localhost 752135642]# su - foo
[foo@localhost ~]$ su - root
Password:
Last login: Tue May 23 11:15:40 CST 2023 on pts/0
[root@localhost ~]#
```

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@localhost ~]# su - root
Password:
Last login: Tue May 23 11:15:40 CST 2023 on pts/0
[root@localhost ~]# userdel foo -r
userdel: user foo is currently used by process 2361
[root@localhost ~]#
```

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；


```
usermod: user 'user1' does not exist
[root@localhost ~]# usermod -g group1 user2
usermod: user 'user2' does not exist
```

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

```
[root@localhost ~]# sudo -u user1 -i -bash -4.2$ sudo cat /etc/shadow
usage: sudo -h | -K | -k | -V
usage: sudo -v [-AknS] [-g group] [-h host] [-p prompt] [-u user]
usage: sudo -l [-AknS] [-g group] [-h host] [-p prompt] [-U user] [-u user]
        [command]
usage: sudo [-AbEHknPS] [-r role] [-t type] [-C num] [-g group] [-h host] [-p
        prompt] [-u user] [VAR=value] [-i|-s] [<command>]
usage: sudo -e [-AknS] [-r role] [-t type] [-C num] [-g group] [-h host] [-p
        prompt] [-u user] file ...
[root@localhost ~]#
```

十六、实验过程分析与讨论

文件系统上的每个文件有一个用户所有者和一个组所有者 如何在 linux 中查询一个组有哪些用户？ 执行 `cat /etc/group | less` 命令，寻找相应的组名称，查看其 最后一个字段即可 如何在 linux 中查询一个用户属于哪些组？ 执行 `cat /etc/group | grep username` 即可（将 `username` 替换 为查找的用户名）。

五、指导教师意见

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 5 月 13 日
学 号	2021213140	姓 名	张贺
专业班级	计算机科学与技术 01 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

十七、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。

十八、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。

十九、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）；

```
bash: su: command not found...
[root@localhost ~]# exit
logout
[752135642@localhost ~]$ foo=200
[752135642@localhost ~]$ echo $foo
200
[752135642@localhost ~]$
```

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）；

```
[752135642@localhost ~]$ bar=100
[752135642@localhost ~]$ test $b
[752135642@localhost ~]$ echo $?
1
```

- 和系统时间 (`date`);

[illegible]

```
[752135642@localhost ~]$ sh myscript2.sh
sh: myscript2: No such file or directory
[752135642@localhost ~]$ sh myscript2.sh
Hostname: localhost.localdomain
Date :Tue May 23 18:09:18 CST 2023
[752135642@localhost ~]$
```

- 数是否为水仙花数;

```
752135642@localhost:~  
File Edit View Search Terminal Help  
echo "Total paramter are : "  
num=$1  
  
n=$num  
while[$n -gt 0 ]  
do  
    m=$((n%10))  
    sum=$((sum+m*m*m))  
    n=$((n/10))  
done  
  
if[ $sum -eq $num]  
then  
    echo "yes"  
else  
    echo "no"  
fi
```

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；

```
[752135642@localhost ~]$ ./sum.sh 2 3 4  
./sum.sh: line 7: echoThe sum is : 9: comm  
and not found  
[752135642@localhost ~]$ ^C  
[752135642@localhost ~]$ vim sum.sh  
[752135642@localhost ~]$ ./sum.sh 2 3 4  
The sum is : 9
```

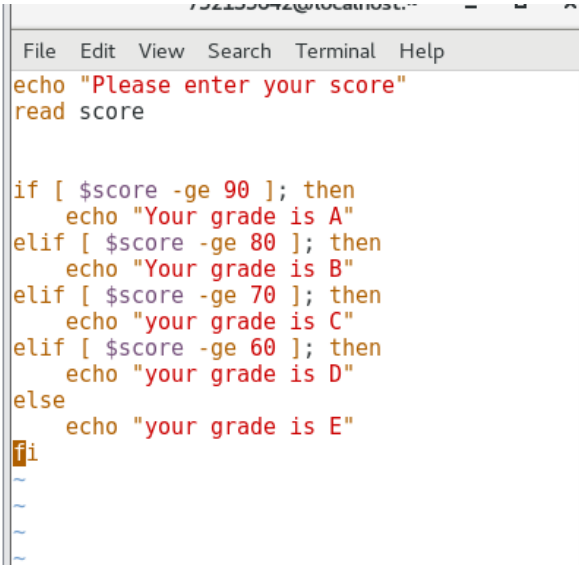
```
752135642@localhost:~  
File Edit View Search Terminal Help  
num1=$1  
num2=$2  
num3=$3  
  
sum=$((num1+num2+num3))  
echo " The sum is : $sum"  
  
~  
~  
~  
~
```

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 。要求使用 实现 四、 实验过程分析与讨论 if 条

件判断[]内的语法格式： 常用参数： 文件/目录判断： [-a FILE] 如果 FILE 存在则为真

```
[752135642@localhost ~]$ vim fs.sh
\ [752135642@localhost ~]$ sh fs.sh 75
Please enter your score
75
your grade is C
[752135642@localhost ~]$
```

7 64 位



```
File Edit View Search Terminal Help
echo "Please enter your score"
read score

if [ $score -ge 90 ]; then
    echo "Your grade is A"
elif [ $score -ge 80 ]; then
    echo "Your grade is B"
elif [ $score -ge 70 ]; then
    echo "your grade is C"
elif [ $score -ge 60 ]; then
    echo "your grade is D"
else
    echo "your grade is E"
fi
~
~
~
```

二十、实验过程分析与讨论

if 条件判断[]内的语法格式：

常用参数： 文件/目录判断：

[-a FILE] 如果 FILE 存在则为真。

[-b FILE] 如果 FILE 存在且是一个块文件则返回为真。

[-c FILE] 如果 FILE 存在且是一个字符文件则返回为真。

[-d FILE] 如果 FILE 存在且是一个目录则返回为真。

[-e FILE] 如果 指定的文件或目录存在时返回为真。

[-f FILE] 如果 FILE 存在且是一个普通文件则返回为真。

数值判断 [INT1 -eq INT2] INT1 和 INT2 两数相等返回为真 ,=[INT1 -ne INT2] INT1 和 INT2 两数不等返回为真 ,<> [INT1 -gt INT2] INT1 大于 INT2 返回为真 ,> [INT1 -ge INT2] INT1 大于等于 INT2 返回为真,>= [INT1 -lt INT2] INT1 小于 INT2 返回为真 ,< [INT1 -le INT2] INT1 小于等于 INT2 返回为真,<= 逻辑判断 [!EXPR] 逻辑非, 如果 EXPR 是 false 则返回为真。

[EXPR1 -a EXPR2] 逻辑与, 如果 EXPR1 and EXPR2 全真则返回为真。 [EXPR1 -o EXPR2] 逻辑或, 如果 EXPR1 或者 EXPR2 为真则 返回为真。 [] || [] 用 OR 来合并两个条件 [] && [] 用 AND 来合并两个条件 其他判断 [-t FD] 如果文件描述符 FD (默认值为 1) 打开且指向一个终端 则返回为真 [-o optionname] 如果 shell 选项 optionname 开启则返回为 IF 高级特性: 双圆括号(()): 表示数学表达式 在判断命令中只允许在比较中进行简单的算术操作, 而双圆括号 提供更多的数学符号, 而且在双圆括号里面的 '>','<' 号不需要转意。

双方括号[[]]: 表示高级字符串处理函数 双方括号中判断命令使用标准的字符串比较, 还可以使用匹配模式, 从而定义与字符串相匹配的正则表达式。 双括号的作用: 在 shell 中, [\$a != 1 || \$b = 2] 是不允许出, 要用[\$a != 1] || [\$b = 2], 而双括号就可以解决这个问题, [[\$a != 1 || \$b = 2]]。又比如这个 ["\$a" -lt "\$b"], 也可以改成双括号的形式(("a" < "b"))

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 5 月 13 日
学 号	2021213140	姓 名	张贺
专业班级	计算机科学与技术 01 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)；可以事先在当前目录下建立若干 *.c 文件用于测试。

```
64 位 | 752135642@localhost:~ | - | □
File Edit View Search Terminal Help
target_dir=~/.workspace/
for file in *.c;
do
    cp "$file" "$target_dir"
done
echo "success"
~
~
~
```

2. 编写 Shell 脚本，利用 while 循环求前 10 个偶数之和，并输出结果；

```
File Edit View Search Terminal Help
n=1
sum=0
while((n<=20))
do
    if ((n<=20))
    then
        sum=$((sum+n))
    fi
    n=$((n+1))
done
echo "$sum"

[752135642@localhost ~]$ vim jj.sh
[752135642@localhost ~]$ sh jj.sh
210
[752135642@localhost ~]$
```

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

```
#!/bin/bash
OS 7 64 位
File Edit View Search Terminal Help
n=1
sum=0
until ((n>10))
do
    square=$((n*n))
    sum=$((sum+square))
    n=$((n+1))
done
echo "$sum"

[752135642@localhost ~]$ vim kkk.sh
[752135642@localhost ~]$ sh kkk.sh
385
```

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 、 break 2 、 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"
```

```
File Edit View Search Terminal Help
[752135642@localhost ~]$ sh lll.sh
a-n 1
-n 2
-n 3
-n 4

b-n 1
-n 2
-n 3
-n 4

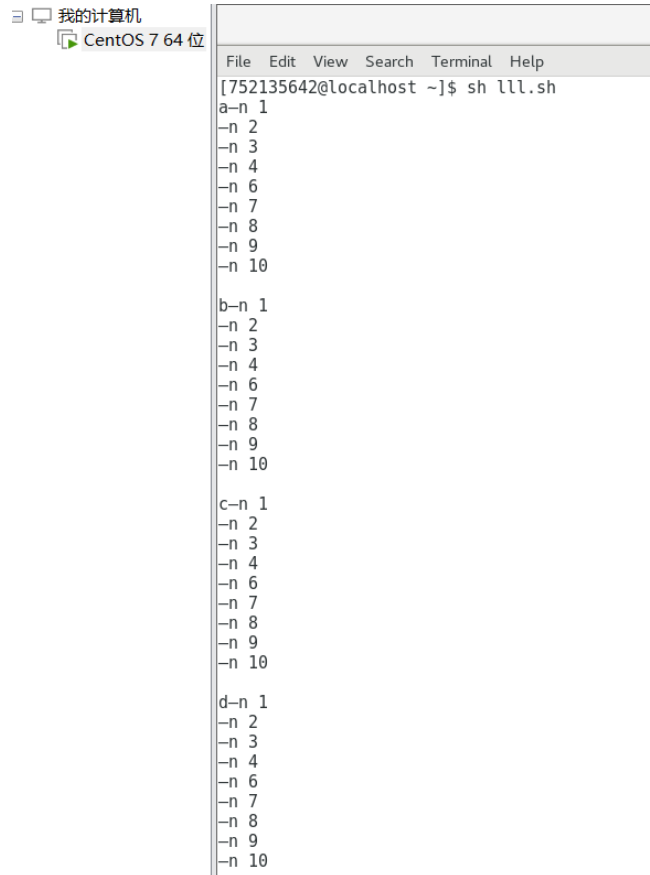
c-n 1
-n 2
-n 3
-n 4

d-n 1
-n 2
-n 3
-n 4
```

```

[752135642@localhost ~]$ vim lll.sh
[752135642@localhost ~]$ sh lll.sh
a-n 1
-n 2
-n 3
-n 4
[752135642@localhost ~]$

```



```

-n 8
-n 9
-n 10

[752135642@localhost ~]$ vim lll.sh
[752135642@localhost ~]$ sh lll.sh
a-n 1
-n 2
-n 3
-n 4
b-n 1
-n 2
-n 3
-n 4
c-n 1
-n 2
-n 3
-n 4
d-n 1
-n 2
-n 3
-n 4

```

三、 实验过程分析与讨论

for 循环，while 循环的使用

1、for 循环

(1) for 循环有三种结构：一种是列表 for 循环，第二种是不带列表 for 循环。第三种是

类 C 风格的 for 循环。

(2) 列表 for 循环 do 和 done 之间的命令称为循环体, 执行次数和 list 列表中常数或字符串的个数相同。for 循环, 首先将 in 后 list 列表的第一个常数或字符串赋值给循环变量, 然后执行循环体, 以此执行 list, 最后执行 done 命令后的命令序列。Shell 支持列表 for 循环使用略写的计数方式, 1~5 的范围用{1..5}表示(大括号不能去掉, 否则会当作一个字符串处理)。Shell 中还支持按规定的步数进行跳跃的方式实现列表 for 循环 for 循环对字符串进行操作, 也可一使用 for file in *, 通配符*产生文件名扩展, 匹配当前目录下的所有文件。

(3) 不带列表 for 循环 由用户制定参数和参数的个数, 与上述的 for 循环列表参数功能相同。#!/bin/bash echo "number of arguments is \$# " echo "What you input is: " for argument do echo "\$argument" done 比上述代码少了\$@参数列表, \$*参数字符串。

2、while 循环

也称为前测试循环语句, 重复次数是利用一个条件来控制是否继续重复执行这个语句。为了避免死循环, 必须保证循环体中包含循环出口条件即表达式存在退出状态为非 0 的情况。

(1) 计数器控制的 while 循环 #!/bin/bash sum=0 i=1 while((i <= 100)) do let "sum+=i" let "i += 2" done echo "sum=\$sum" 指定了循环的次数 500, 初始化计数器值为 1, 不断测试循环条件 i 是否小于等于 100。在循环条件中设置了计数器加 2 来计算 1~100 内所有的奇数之和。

(2) 结束标记控制的 while 循环 设置一个特殊的数据值(结束标记)来结束 while 循环。#!/bin/bash echo "Please input the num(1-10) " read num while [["\$num" != 4]] do if ["\$num" -lt 4] then echo "Too small. Try again!" read num elif ["\$num" -gt 4] then echo "To high. Try again" read num else exit 0 fi done echo "Congratulation, you are right! "

(4) 命令行控制的 while 循环 使用命令行来指定输出参数和参数个数, 通常与 shift 结合使用, shift 命令使位置变量 下移一位(\$2 代替\$1、\$3 代替\$2, 并使\$#变量递减), 当最后一个参数显示给用户,\$#会 等于 0,\$*也等于空。#!/bin/bash echo "number of arguments is \$# " echo "What you input is: " while [["\$*" != ""]] do echo "\$1" shift done 循环条件可以改写为 while[["\$#" -ne 0]], 等于 0 时退出 while 循环

3、until 循环 until 命令和 while 命令类似, while 能实现的脚本 until 同样也可以实现, 但区别是 until 循环的退出状态是不为 0, 退出状态是为 0(与 while 刚好相反), 即 while 循环在条件为真 时继续执行循环而 until 则在条件为假时执行循环。#!/bin/bash i=0 until [["\$i" -gt 5]] #大于 5 do let "square=i*i" echo "\$i * \$i = \$square" let "i++" done

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 5 月 13 日
学 号	2021213140	姓 名	张贺
专业班级	计算机科学与技术 01 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

二十一、 实验目的

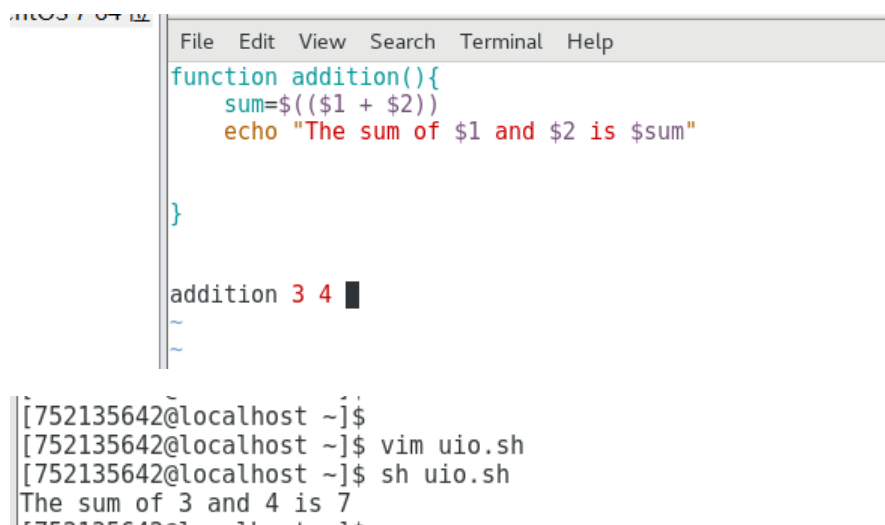
1. 掌握 Shell 函数的定义方法;
2. 掌握 Shell 函数的参数传递、调用和返回值;
3. 掌握 Shell 函数的递归调用方法;
4. 理解 Shell 函数的嵌套

二十二、 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。

二十三、 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;

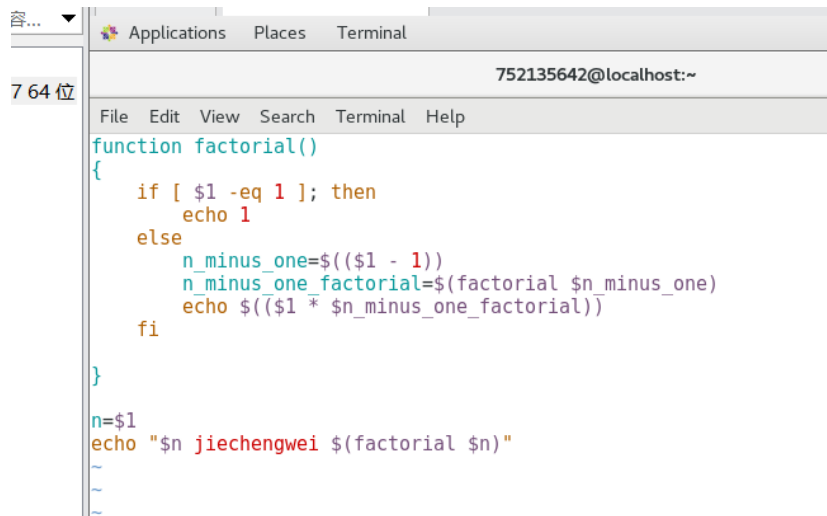


```
File Edit View Search Terminal Help
function addition(){
    sum=$(( $1 + $2 ))
    echo "The sum of $1 and $2 is $sum"
}

addition 3 4

[752135642@localhost ~]$
[752135642@localhost ~]$ vim uio.sh
[752135642@localhost ~]$ sh uio.sh
The sum of 3 and 4 is 7
```

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；



The screenshot shows a terminal window with a menu bar (Applications, Places, Terminal) and a title bar (752135642@localhost:~). The terminal content is as follows:

```
function factorial()
{
    if [ $1 -eq 1 ]; then
        echo 1
    else
        n_minus_one=$(( $1 - 1 ))
        n_minus_one_factorial=$(factorial $n_minus_one)
        echo $(( $1 * $n_minus_one_factorial ))
    fi
}

n=$1
echo "$n jiechengwei $(factorial $n)"
```

3. 一个 Shell 脚本的内容如下所示： 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
```

```
[752135642@localhost ~]$ vim myscript8.sh
[752135642@localhost ~]$ sh myscript8.sh
start...
this is the first
this is the second
-----this is third
[752135642@localhost ~]$
```


二十四、 实验过程分析与讨论

函数调用的相关知识。

Shell 函数定义的语法格式如下：

`function name() { statements [return value] }` 对各个部分的说明：

`function` 是 Shell 中的关键字，专门用来定义函数；`name` 是函数名；`statements` 是函数要执行的代码，也就是一组语句；`return value` 表示函数的返回值，其中 `return` 是 Shell 关键字，专门用在函数中返回一个值；这一部分可以写也可以不写。

由 `{ }` 包围的部分称为函数体，调用一个函数，实际上就是执行函数体中的代码。函数定义的简化写法 如果你嫌麻烦，函数定义时也可以不写 `function` 关键字：

`name() { statements [return value] }`

如果写了 `function` 关键字，也可以省略函数名后面的小括号：

`function name { statements [return value] }`

函数调用

调用 Shell 函数时可以给它传递参数，也可以不传递。如果不传递参数，直接给出函数名字即可：`name` 如果传递参数，那么多个参数之间以空格分隔：`name param1 param2 param3` 不管是哪种形式，函数名字后面都不需要带括号。

和其它编程语言不同的是，Shell 函数在定义时不能指明参数，但是在调用时却可以传递参数，并且给它传递什么参数它就接收什么参数。

Shell 也不限制定义和调用的顺序，你可以将定义放在调用的前面，也可以反过来，将定义放在调用的后面。

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 5 月 13 日
学 号	2021213140	姓 名	张贺
专业班级	计算机科学与技术 01 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

二十五、 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；

二十六、 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。

二十七、 实验内容及结果

1. 文件 quote.txt 的内容如下所示： 试使用 sed 命令实现如下功能：

- (1) 删除 \$ 符号；

```
[752135642@localhost ~]$ 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[752135642@localhost ~]$
```

- (2) 显示包含 music 文字的行内容及行号；

```
The local nurse Miss P.Neave was in attendance[752135642@localhost ~]$
[752135642@localhost ~]$ cat quote.txt | sed -n 'music/p'
The honeysuckle band played all night long for only $90.
It was an evening of splendid music and company.
Too bad the disco floor fell through at 23:10.
The local nurse Miss P.Neave was in attendance[752135642@localhost ~]$
```

- (3) 在第 4 行后面追加内容： "hello world!" ；

```

The local nurse Miss P.Neave was in attendance[752135642@localhost ~]$
[752135642@localhost ~]$ cat quote.txt | sed '4a hello world!'
The honeysuckle band played all night long for only $90.
It was an evening of splendid music and company.
Too bad the disco floor fell through at 23:10.
The local nurse Miss P.Neave was in attendance
hello world!
[752135642@localhost ~]$ █

```

(4) 将文本 "The" 替换为 "Quod" ；

```

hello world!
[752135642@localhost ~]$ cat quote.txt | sed 's/The/Quod/g'
Quod honeysuckle band played all night long for only $90.
It was an evening of splendid music and company.
Too bad the disco floor fell through at 23:10.
Quod local nurse Miss P.Neave was in attendance[752135642@localhost ~]$ █

```

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

```

Too bad the disco floor fell through at 23:10.
Quod local nurse Miss P.Neave was in attendance[752135642@localhost ~]$
[752135642@localhost ~]$ cat quote.txt | sed '3c This is the third line'
The honeysuckle band played all night long for only $90.
It was an evening of splendid music and company.
This is the third line
The local nurse Miss P.Neave was in attendance[752135642@localhost ~]$ █

```

(6) 删除第 2 行内容；

```

[752135642@localhost ~]$
[752135642@localhost ~]$ cat quote.txt | sed '2d'
The honeysuckle band played all night long for only $90.
Too bad the disco floor fell through at 23:10.
The local nurse Miss P.Neave was in attendance[752135642@localhost ~]

```

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

```

The local nurse Miss P.Neave was in attendance[752135642@localhost ~]$
[752135642@localhost ~]$
[752135642@localhost ~]$ var=evening
[752135642@localhost ~]$ cat quote.txt | sed -n "/$var/p"
It was an evening of splendid music and company.
[752135642@localhost ~]$ █

```

2.文件 numbers.txt 的内容如下所示: 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;

```

cat: number.txt: No such file or directory
[752135642@localhost ~]$ vim number.txt
[752135642@localhost ~]$ cat number.txt | awk '{FS=":"}{print $2}'
:
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。 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. one : two : three four : five : six 例如： foo.txt 内容为： 则输出为：

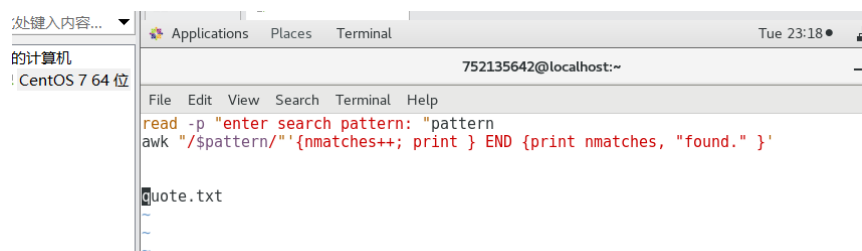
```
five
[752135642@localhost ~]$ vim foo.txt
[752135642@localhost ~]$ awk '{for(i=1;i<NF;i++) if ($i%2==0){print $i;count++}} END{print "The total number of even numbers is : " count}' foo.txt
2
4
46
The total number of even numbers is : 3
[752135642@localhost ~]$
```

4. 脚本的内容如下所示： 试运行该脚本，并理解该脚本实现的功能。

awk 中"/\$pattern/"这一部分用双引号括起来，是为了允许引号内的 S hell 变量进行替换此脚本的作用用于匹配字符串首先输入你要匹配的 字符串，脚本中指定的文件为 info.txt 并在 info.txt 文件中查找相应的 字符串，如果能匹配到，则 nmatches 变量就加一，并在最后输出要 匹配字符串出现的位置，以及出现的次数

```
[752135642@localhost ~]$ cat scripts.sh
cat: scripts.sh: No such file or directory
[752135642@localhost ~]$ vim scripts.sh
[752135642@localhost ~]$ sh scripts.sh
enter search pattern: patternname

1 found.
scripts.sh: line 5: quote.txt: command not found
[752135642@localhost ~]$
```



二十八、 实验过程分析与讨论

sed 和 awk 的用法:

1. sed 命令的作用是利用脚本来处理文本文件。
2. 使用方法: sed [参数] [n1][n2]function n1,n2 不一定存在,一般表示进行动作的行。如果动作在 10-20 行进行,则 为 10,20[function]
3. 参数说明: • -e 或--expression= 以选项中指定的 script 来处理输入 的文本文件,这个 -e 可以省略,直接写表达式。 • -f 或--file=以选项中指定的 script 文件来处理输入的文 本文件。 • -h 或--help 显示帮助。 • -n 或 --quiet 或 --silent 仅显示 script 处理后的结 果。 • -V 或 --version 显示版本信息。 • -i 直接在源文件里修改内容 动作说明[function]: • a: 追加, a 的后面可以接字符串,而这些字符串会在目标行末尾 追加~ • c: 取代, c 的后面可以接字符串,这些字符串可以取代 n1,n2 之 间的行! • d: 删除,因为是删除啊,所以 d 后面通常不接任何咚咚; • i: 插入, i 的后面可以接字符串,而 这些字符串会在新的一行出现(目 前的上一行); • p: 打印,亦即将某个选择的数据印 出。通常 p 会与参数 sed -n 一起运行~ s: 取代,通常这个 s 的动作可以搭配正规 表示法,例如 1,20s/old/new/g

五、指导教师意见

指导教师签字: 卢洋