

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 8 日
学 号	2021223281	姓 名	任可欣
专业班级	计算机科学与技术 06 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

```
ren@DESKTOP-E80BM21:/mnt/c/Windows/system32$ cd /etc
ren@DESKTOP-E80BM21:/etc$ pwd
/etc
```

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。

```
ren@DESKTOP-E80BM21:~$ cd /home
ren@DESKTOP-E80BM21:/home$ cd ren
ren@DESKTOP-E80BM21:~$ ls -a
. .bash_history .bash_logout .bashrc .motd_shown .profile
```

3. 使用命令创建目录/home/lyj/linux，然后删除该目录。

```
ren@DESKTOP-E80BM21:~$ mkdir /home/ren/linux
ren@DESKTOP-E80BM21:~$ ls
linux
ren@DESKTOP-E80BM21:~$ rmdir /home/ren/linux
```

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

```
ren@DESKTOP-E80BM21:/home$ cd ren
ren@DESKTOP-E80BM21:~$ cat> abc
hello, linux!

ren@DESKTOP-E80BM21:~$ ls
abc  baz
ren@DESKTOP-E80BM21:~$ cat zbc
cat: zbc: No such file or directory
ren@DESKTOP-E80BM21:~$ cat abc
hello, linux!

ren@DESKTOP-E80BM21:~$
```

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。

```
ren@DESKTOP-E80BM21:~$ ls
abc  baz
ren@DESKTOP-E80BM21:~$ cat zbc
cat: zbc: No such file or directory
ren@DESKTOP-E80BM21:~$ cat abc
hello,linux!

ren@DESKTOP-E80BM21:~$ ls
abc  baz
ren@DESKTOP-E80BM21:~$ mkdir ak
ren@DESKTOP-E80BM21:~$ ls
abc  ak  baz
ren@DESKTOP-E80BM21:~$ cp -r abc ak
ren@DESKTOP-E80BM21:~$ cd ak
ren@DESKTOP-E80BM21:~/ak$ cat abc
hello,linux!

ren@DESKTOP-E80BM21:~/ak$ cd../
-bash: cd../: No such file or directory
ren@DESKTOP-E80BM21:~/ak$ cd ../
ren@DESKTOP-E80BM21:~$ rm -r ak
ren@DESKTOP-E80BM21:~$ ls
abc  baz
ren@DESKTOP-E80BM21:~$
```

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。

```

ren@DESKTOP-E8OBM21: /etc
70 # regular expression when creating a new home directory
71 SKEL_IGNORE_REGEX="dpkg-(old|new|dist|save)"
72
73 # Set this if you want the --add_extra_groups option to adduser to add
74 # new users to other groups.
75 # This is the list of groups that new non-system users will be added to
76 # Default:
77 #EXTRA_GROUPS="dialout cdrom floppy audio video plugdev users"
78
79 # If ADD_EXTRA_GROUPS is set to something non-zero, the EXTRA_GROUPS
80 # option above will be default behavior for adding new, non-system users
81 #ADD_EXTRA_GROUPS=1
82
83
84 # check user and group names also against this regular expression.
85 #NAME_REGEX="^[a-z][-a-z0-9_]*\$"
86
87 # use extrausers by default
88 #USE_EXTRAUSERS=1
ren@DESKTOP-E8OBM21:/etc$ head -n 3 adduser.conf
# /etc/adduser.conf: 'adduser' configuration.
# See adduser(8) and adduser.conf(5) for full documentation.
ren@DESKTOP-E8OBM21:/etc$ tail -n 5 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
ren@DESKTOP-E8OBM21:/etc$

```

7、分屏查看文件/etc/adduser.conf 的内容。

```

ren@DESKTOP-E8OBM21: /etc
# /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 GROUPHOMES is "yes", then the home directories will be created as
# /home/groupname/user.
GROUPHOMES=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.
adduser.conf

```

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

```
ren@DESKTOP-E80BM21:/mnt/c/Windows/system32$ cd /home
ren@DESKTOP-E80BM21:/home$ cd ren
ren@DESKTOP-E80BM21:~$ cat > facebook.txt
google 110 5000
baidu 100 5000
guge 50 3000
sohu 100 4500
```

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
- (2) 按公司人数排序
- (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
- (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
- (5) 从公司英文名称的第2个字母开始进行排序。

```
ren@DESKTOP-E8OBM21:~$ sort -r facebook.txt
sort: cannot read: facebook.txt: No such file or directory
ren@DESKTOP-E8OBM21:~$ cat > facebook.txt
google 110 5000
baidu 100 5000
guge 50 3000
sohu 100 4500
^C
ren@DESKTOP-E8OBM21:~$ sort -r facebook.txt
sohu 100 4500
guge 50 3000
google 110 5000
baidu 100 5000
^C
ren@DESKTOP-E8OBM21:~$ sort -n facebook.txt
^C
baidu 100 5000
google 110 5000
guge 50 3000
sohu 100 4500
ren@DESKTOP-E8OBM21:~$ sort -n -t ' ' -k 2 -k 3 facebook.txt
^C
guge 50 3000
sohu 100 4500
baidu 100 5000
google 110 5000
ren@DESKTOP-E8OBM21:~$ sort -n -t ' ' -k 3r -k 2 facebook.txt
baidu 100 5000
google 110 5000
sohu 100 4500
guge 50 3000
^C
ren@DESKTOP-E8OBM21:~$ sort -t ' ' -k 1.2 facebook.txt
^C
baidu 100 5000
sohu 100 4500
google 110 5000
guge 50 3000
ren@DESKTOP-E8OBM21:~$
```


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 3 月 15 日
学 号	2021223281	姓 名	任可欣
专业班级	计算机科学与技术 06 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` 。

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 ，并写如任意几行内容；

```
ren@DESKTOP-E80BM21:~/etc$ cd ~
ren@DESKTOP-E80BM21:~$ mkdir baz

ren@DESKTOP-E80BM21:~$ cd baz/
ren@DESKTOP-E80BM21:~/baz$ touch que
ren@DESKTOP-E80BM21:~/baz$ touch qux
ren@DESKTOP-E80BM21:~/baz$ echo 'hello,linux' >qux
```

(2) 在用户目录下查找文件 qux ，并显示该文件位置信息；

```
ren@DESKTOP-E80BM21:~/baz$ find ~ -name qux
/home/ren/baz/qux
ren@DESKTOP-E80BM21:~/baz$
```

(3) 统计文件 qux 中所包含内容的行数、字数和字节数；

```
ren@DESKTOP-E80BM21:~/baz$ wc que
1 1 12 que
ren@DESKTOP-E80BM21:~/baz$
```

(4) 在用户目录下查找文件 qux ，并删除该文件；

```
ren@DESKTOP-E80BM21:~/baz$ find ~ -name qux -delete
ren@DESKTOP-E80BM21:~/baz$
```

(5) 查看文件夹 baz 内容，看一下是否删除了文件 qux 。

```
ren@DESKTOP-E80BM21:~/baz$ ls
ren@DESKTOP-E80BM21:~/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 ；

```
ren@DESKTOP-E80BM21:~/baz$ mkdir path1
ren@DESKTOP-E80BM21:~/baz$ cd path1
ren@DESKTOP-E80BM21:~/baz/path1$ touch file1 file2
ren@DESKTOP-E80BM21:~/baz/path1$ ls
file1 file2
ren@DESKTOP-E80BM21:~/baz/path1$
```

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

```
ren@DESKTOP-E80BM21:~/baz/path1$ mkdir path2
ren@DESKTOP-E80BM21:~/baz/path1$ cd path2
ren@DESKTOP-E80BM21:~/baz/path1/path2$ touch file3
ren@DESKTOP-E80BM21:~/baz/path1/path2$ ls
file3
ren@DESKTOP-E80BM21:~/baz/path1/path2$
```

(3) 在用户目录下新建文件 file4 ；

```
ren@DESKTOP-E80BM21:~/baz/path1/path2$ touch file4
ren@DESKTOP-E80BM21:~/baz/path1/path2$ ls
file3 file4
ren@DESKTOP-E80BM21:~/baz/path1/path2$
```

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

```
ren@DESKTOP-E80BM21:~/baz/path1/path2$ tar -cvf package.tar path1 file4
tar: path1: Cannot stat: No such file or directory
file4
tar: Exiting with failure status due to previous errors
```

(5) 查看包 package.tar 的内容；

```
ren@DESKTOP-E80BM21:~/baz/path1/path2$ tar -tvf package.tar
-rw-r--r-- ren/ren      0 2023-05-23 21:01 file4
ren@DESKTOP-E80BM21:~/baz/path1/path2$
```

(6) 向包 package.tar 里添加文件夹 path2 的内容；

```
ren@DESKTOP-E80BM21:~/baz/path1/path2/path3/path2$ tar -rvf package.tar path2
```

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

```
ren@DESKTOP-E80BM21:~/baz/path1/path2$ mkdir path3
ren@DESKTOP-E80BM21:~/baz/path1/path2$ cp package.tar path3
ren@DESKTOP-E80BM21:~/baz/path1/path2$
```

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

```
ren@DESKTOP-E80BM21:~/baz/path1/path2/path3/path2$ tar -xvf package.tar
file4
```

3. 符号链接内容

(1) 新建文件 `foo.txt`，内容为 `123`；

```
ren@DESKTOP-E8OBM21: ~/baz/path1/path2/path3/path2
ren@DESKTOP-E8OBM21:~/baz/path1/path2/path3/path2$ echo "123" >foo.txt
ren@DESKTOP-E8OBM21:~/baz/path1/path2/path3/path2$
```

(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；

```
ren@DESKTOP-E8OBM21:~/baz/path1/path2/path3/path2$ ln foo.txt bar.txt
ren@DESKTOP-E8OBM21:~/baz/path1/path2/path3/path2$ diff bar.txt foo.txt
ren@DESKTOP-E8OBM21:~/baz/path1/path2/path3/path2$
```

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

```
ren@DESKTOP-E8OBM21:~/baz/path1/path2/path3/path2$ ls -li foo.txt bar.txt
29381 bar.txt 29381 foo.txt
ren@DESKTOP-E8OBM21:~/baz/path1/path2/path3/path2$
```

(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；

```
ren@DESKTOP-E8OBM21:~/baz/path1/path2/path3/path2$ echo "abc" >bar.txt
ren@DESKTOP-E8OBM21:~/baz/path1/path2/path3/path2$ diff foo.txt bar.txt
ren@DESKTOP-E8OBM21:~/baz/path1/path2/path3/path2$
```

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

```
ren@DESKTOP-E8OBM21:~/baz/path1/path2/path3/path2$ ls -li bar.txt
29381 bar.txt
ren@DESKTOP-E8OBM21:~/baz/path1/path2/path3/path2$ cat bar.txt
abc
ren@DESKTOP-E8OBM21:~/baz/path1/path2/path3/path2$
```

(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；

```
ren@DESKTOP-E80BM21:~/baz/path1/path2/path3/path2$ ln -s bar.txt baz.txt
ren@DESKTOP-E80BM21:~/baz/path1/path2/path3/path2$ ls -li bar.txt baz.txt
29381 bar.txt 29382 baz.txt
ren@DESKTOP-E80BM21:~/baz/path1/path2/path3/path2$ diff baz.txt bar.txt
ren@DESKTOP-E80BM21:~/baz/path1/path2/path3/path2$
```

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

```
ren@DESKTOP-E80BM21:~/baz/path1/path2/path3/path2$ rm bar.txt
ren@DESKTOP-E80BM21:~/baz/path1/path2/path3/path2$ cat baz.txt
cat: baz.txt: No such file or directory
ren@DESKTOP-E80BM21:~/baz/path1/path2/path3/path2$
```

4. 权限管理

(1) 新建文件 qux.txt ；

```
ren@DESKTOP-E80BM21:~/baz/path1/path2/path3/path2$ touch que.txt
ren@DESKTOP-E80BM21:~/baz/path1/path2/path3/path2$
```

(2) 为文件 qux.txt 增加执行权限（所有用户都可以执行）

```
ren@DESKTOP-E80BM21:~/baz/path1/path2/path3/path2$ chmod +x que.txt
ren@DESKTOP-E80BM21:~/baz/path1/path2/path3/path2$
```


四、 实验过程分析与讨论

".tar"格式的打包和解打包都使用 tar 命令，区别只是选项不同。我们先看看 tar 命令的基本信息。

命令名称：tar。

```
[root@localhost ~]#tar [选项] [-f 压缩包名] 源文件或目录
```

选项：

-c: 打包

-f: 指定压缩包的文件名。压缩包的扩展名是用来给管理员识别格式的，所以一定要正确指定扩展名；

-v: 显示打包文件过程；

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 3 月 22 日
学 号	2021223281	姓 名	任可欣
专业班级	计算机科学与技术 06 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

掌握 vim 编辑器及 gcc 编译器的使用方法。

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. 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 ；

```
ren@DESKTOP-E8OBM21: ~/baz/path1/path2/path3/path2/workspace1  
ren@DESKTOP-E8OBM21: ~/baz/path1/path2/path3/path2$ mkdir workspace1
```

(2) 进入目录 workspace1 ；

```
ren@DESKTOP-E8OBM21: ~/baz/path1/path2/path3/path2$ cd workspace1  
ren@DESKTOP-E8OBM21: ~/baz/path1/path2/path3/path2/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：

```
test.c+
#include<stdio.h>
int main() {
    printf("hello world!\n");
    return 0;
}
```

(4) 保存 test.c 的内容, 并退出;


```
#include<stdio.h>
int main() {
    printf("hello world!\n");
    return 0;
}
```

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

```
[root@localhost workspace1]# gcc test.c -o test
[root@localhost workspace1]# ./test
hello,linux!
```

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;

```
[root@localhost workspace1]# cd ../
[root@localhost ~]# mkdir workspace2
```

(2) 进入 workspace2 目录;

```
[root@localhost ~]# cd workspace2  
[root@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;

```
[root@localhost workspace2]# cat /etc/gai.conf > ./gai.conf  
-bash: ./: Is a directory  
[root@localhost workspace2]# ls  
gai.conf  
[root@localhost workspace2]#
```

(4) 使用 vim 编辑当前目录下的 gai.conf ;

```
ren@DESKTOP-E80BM21: ~/workspace2$ vim gai.conf  
ren@DESKTOP-E80BM21: ~/workspace2$
```

(5) 将光标移到第 18 行;

在命令行模式下按18G

```
# 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 include:  
# reload <yes|no>  
#   If set to yes, each getaddrinfo(3) call will check whether this file  
#   changed and if necessary reload. This option should not really be  
#   used. There are possible runtime problems. The default is no.  
#  
# label <mask> <value>  
#   Add another rule to the RFC 3484 label table. See section 2.1 in  
#   RFC 3484. The default is:  
#  
#label ::1/128 0  
#label ::/0 1  
#label 2002::/16 2  
#label ::/96 3  
#label ::ffff:0:0/96 4  
#label fec0::/10 5  
#label fc00::/7 6  
#label 2001:0::/32 7  
#
```

(6) 复制该行内容;

命令模式下按yy

(7) 将光标移到最后一行行首;

末行G

(8) 粘贴复制行的内容;

```
#include <stdio.h>
```

```
int main( )
```

```
{
```

```
    printf("hello world!\n");
```

```
    return 0;
```

```
}
```

```
cat /etc/gai.conf > ./gai.conf
```

```

# site-local IPv4 and IPv6 addresses a lookup for a global address would
# see the IPv6 be preferred. The result is a long delay because the
# site-local IPv6 addresses cannot be used while the IPv4 address is
# (at least for the foreseeable future) NATed. We also treat Teredo
# tunnels special.
#
# precedence <mask> <value>
#   Add another rule to the RFC 3484 precedence table. See section 2.1
#   and 10.3 in RFC 3484. The default is:
#
#precedence ::1/128          50
#precedence ::/0             40
#precedence 2002::/16        30
#precedence ::/96            20
#precedence ::ffff:0:0/96    10
#
#   For sites which prefer IPv4 connections change the last line to
#
#precedence ::ffff:0:0/96    100
#
#
# scopev4 <mask> <value>
#   Add another rule to the RFC 6724 scope table for IPv4 addresses.
#   By default the scope IDs described in section 3.2 in RFC 6724 are
#   used. Changing these defaults should hardly ever be necessary.
#   The defaults are equivalent to:
#
#scopev4 ::ffff:169.254.0.0/112 2
#scopev4 ::ffff:127.0.0.0/104    2
#
# label <mask> <value>
#scopev4 ::ffff:0.0.0.0/96       14

```

(9) 撤销第 8 步的动作;

(u)

```

# site-local IPv4 and IPv6 addresses a lookup for a global address would
# see the IPv6 be preferred. The result is a long delay because the
# site-local IPv6 addresses cannot be used while the IPv4 address is
# (at least for the foreseeable future) NATed. We also treat Teredo
# tunnels special.
#
# precedence <mask> <value>
#   Add another rule to the RFC 3484 precedence table. See section 2.1
#   and 10.3 in RFC 3484. The default is:
#
#precedence ::1/128          50
#precedence ::/0             40
#precedence 2002::/16        30
#precedence ::/96            20
#precedence ::ffff:0:0/96    10
#
#   For sites which prefer IPv4 connections change the last line to
#
#precedence ::ffff:0:0/96    100
#
# scopev4 <mask> <value>
#   Add another rule to the RFC 6724 scope table for IPv4 addresses.
#   By default the scope IDs described in section 3.2 in RFC 6724 are
#   used. Changing these defaults should hardly ever be necessary.
#   The defaults are equivalent to:
#
#scopev4 ::ffff:169.254.0.0/112 2
#scopev4 ::ffff:127.0.0.0/104    2
#scopev4 ::ffff:0.0.0.0/96       14
1 line less; before #1 17:30:41

```

(10) 存盘但不退出;

:w


```

# site-local IPv4 and IPv6 addresses a lookup for a global address would
# see the IPv6 be preferred. The result is a long delay because the
# site-local IPv6 addresses cannot be used while the IPv4 address is
# (at least for the foreseeable future) NATed. We also treat Teredo
# tunnels special.
#
# precedence <mask> <value>
#   Add another rule to the RFC 3484 precedence table. See section 2.1
#   and 10.3 in RFC 3484. The default is:
#
#precedence ::1/128          50
#precedence :::/0            40
#precedence 2002::/16        30
#precedence ::/96            20
#precedence ::ffff:0:0/96    10
#
#   For sites which prefer IPv4 connections change the last line to
#
#precedence ::ffff:0:0/96    100
#
# scopev4 <mask> <value>
#   Add another rule to the RFC 6724 scope table for IPv4 addresses.
#   By default the scope IDs described in section 3.2 in RFC 6724 are
#   used. Changing these defaults should hardly ever be necessary.
#   The defaults are equivalent to:
#
#scopev4 ::ffff:169.254.0.0/112 2
#scopev4 ::ffff:127.0.0.0/104    2
#scopev4 ::ffff:0.0.0.0/96       14
"gai.conf" 65L, 2584B written

```

(11) 将光标移到首行;

(gg)

```

# 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.

```

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

```

Hello, this is vim world!
# 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 include:
#
# reload <yes|no>
#   If set to yes, each getaddrinfo(3) call will check whether this file
#   changed and if necessary reload. This option should not really be
#   used. There are possible runtime problems. The default is no.
#
# label <mask> <value>
#   Add another rule to the RFC 3484 label table. See section 2.1 in
#   RFC 3484. The default is:
#
#label ::1/128      0
#label ::/0         1
#label 2002::/16    2
#label ::/96        3
#label ::ffff:0:0/96 4
#label fec0::/10    5
#label fc00::/7     6
#label 2001:0::/32  7
#
-- INSERT --

```

(13) 删除字符串 "this" ;

```
:%s/this
```

(14) 强制退出 vim , 不存盘

```
:q!
```

八、 实验过程分析与讨论

我对 vim 编译器有了更深的了解

- vi 编辑器是所有 Unix 及 Linux 系统下标准的编辑器,他就相当于 windows 系统中的记事本一样,它的强大不逊色于任何最新的文本编辑器。他是我们使用 Linux 系统不能缺少的工具。由于对 Unix 及 Linux 系统的任何版本, vi 编辑器是完全相同的, 学会它后, 我们将会 Linux 的世界里畅行无阻。

- vim 具有程序编辑的能力, 可以以字体颜色辨别语法的正确性, 方便程序设计; 因为程序简单, 编辑速度相当快速。

- vim 可以当作 vi 的升级版本, 他可以用多种颜色的方式来显示一些特殊的信息。

- vim 会依据文件扩展名或者是文件内的开头信息, 判断该文件的内容而自动的执行该程序的语法判断式, 再以颜色来显示程序代码与一般信息。

- vim 里面加入了很多额外的功能, 例如支持正则表达式的搜索、多文件编辑、块复制等等。

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 3 月 29 日
学 号	2021223281	姓 名	任可欣
专业班级	计算机科学与技术 06 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、 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 ；

```
# useradd -c 'bar' -s /bin/sh -d /home/foo -m
root@LAPTOP-CGNJ5JQJ: # passwd foo
New password:
Retype new password:
passwd: password updated successfully
```

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-CGNJ5JQJ: ~
root@LAPTOP-CGNJ5JQJ:~# useradd -c 'bar' -s /bin/sh -d /home/foo -m
foo
root@LAPTOP-CGNJ5JQJ:~# passwd foo
New password:
Retype new password:
passwd: password updated successfully
root@LAPTOP-CGNJ5JQJ:~# su foo
$ usermod -u 2000 -s /bin/csh foo
usermod: user foo is currently used by process 94
$ sudo usermod -u 2000 -s /bin/csh foo
[sudo] password for foo:
foo is not in the sudoers file. This incident will be reported.
$ 123456
sh: 3: 123456: not found
$ sudo usermod -u 2000 -s /bin/sh foo
[sudo] password for foo:
foo is not in the sudoers file. This incident will be reported.
$
```

3. 从用户 foo 切换到 root ；

```
ren@DESKTOP-E80BM21:~$ su foo
```

4. 删除 foo 用户，并在删除该用户的同时一并删除其家目录；

```
ren@DESKTOP-E80BM21:~$ userdel foo -r
```

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；

```
ren@DESKTOP-E80BM21:~$ 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
> EOF
```

```
ren@DESKTOP-E80BM21:~$ 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
ren@DESKTOP-E80BM21:~$
```

```
ren@DESKTOP-E80BM21:~$ newusers < user.txt
```

```
ren@DESKTOP-E80BM21:~$ pwunconv
```

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

```
ren@DESKTOP-E80BM21:~$ groupadd group -g 3000
```

7. 在用户组 `group1` 中添加两个之前批量创建的用户；

```
ren@DESKTOP-E80BM21:~$ usermod -g group user1
```

```
ren@DESKTOP-E80BM21:~$ usermod -g group user2
```

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

十二、 实验过程分析与讨论

遇到的困难是转换用户 `su userone` 无法正常实现。

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 5 日
学 号	2021223281	姓 名	任可欣
专业班级	计算机科学与技术 06		
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）；

```
ren@DESKTOP-E80BM21:~$ foo=200
ren@DESKTOP-E80BM21:~$ echo $foo
200
ren@DESKTOP-E80BM21:~$
```

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）；

```
ren@DESKTOP-E80BM21:~$ bar=100
ren@DESKTOP-E80BM21:~$ test $bar -gt 150
ren@DESKTOP-E80BM21:~$ echo $?
1
ren@DESKTOP-E80BM21:~$
```

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

```
ren@DESKTOP-E80BM21:~$ cd scripts
ren@DESKTOP-E80BM21:~/scripts$ cat > rkx.sh
#!/bin/bash
hn=$(hostname)
da=$(date)
echo $hn
echo $da
ren@DESKTOP-E80BM21:~/scripts$ chmod u+x rkx.sh
ren@DESKTOP-E80BM21:~/scripts$ bash rkx.sh
bash: rkx.sh: No such file or directory
ren@DESKTOP-E80BM21:~/scripts$ bash rkx.sh
DESKTOP-E80BM21
Wed May 24 09:56:17 CST 2023
```

4. 创建一个Shell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；

```
ren@DESKTOP-E80BM21:~/scripts$ cat > rkx2.sh
#!/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 == $num ))
then
    echo "true"
else
    echo "false"
fi
ren@DESKTOP-E80BM21:~/scripts$ chmod u+x rkx2.sh
ren@DESKTOP-E80BM21:~/scripts$ bash rkx2.sh

ren@DESKTOP-E80BM21:~/scripts$ bash rkx2.sh
Input number:124
rkx2.sh: line 13: ((: expr $num13 + $num23 + $num33 == 124 : syntax error: invalid
$num13 + $num23 + $num33 == 124 ")
false
```

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;

```
ren@DESKTOP-E80BM21:~/scripts$ cat >rkx3.sh
#!/bin/bash
echo 'sum = ' `expr $1 + $2 + $3`

ren@DESKTOP-E80BM21:~/scripts$ chmod u+x rkx3.sh
ren@DESKTOP-E80BM21:~/scripts$ bash rkx3.sh
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 。要求使用

实现。

```
ren@DESKTOP-E80BM21:~$ cd scripts
ren@DESKTOP-E80BM21:~/scripts$ cat >rkx5.sh
#!/bin/bash
#Sat Apr 22 10:50:20 CST 2023
read -p "Please input your grades:" 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
ren@DESKTOP-E80BM21:~/scripts$ chmod u+x rkx5.sh
ren@DESKTOP-E80BM21:~/scripts$ bash rkx5.sh
Please input your grades:99
level is A
ren@DESKTOP-E80BM21:~/scripts$
```

十六、 实验过程分析与讨论

通过本次实验我了解到了：

“# ”开头的就是注释，被编译器忽略

单行注释： #

多行注释： :<

运行 shell 时，会同时存在三种变量：

局部变量：局部变量脚本或命令中定义，仅在当前 shell 实例中有效，其他 shell 启动的程序不能访问局部变量。

环境变量：所有的程序，包括 shell 启动的程序，都能访问环境变量，有些程序需要环境变量来保证其正常运行。必要的时候 shell 脚本也可以定义环境变量。

shell 变量：shell 变量是由 shell 程序设置的特殊变量。shell 变量中有一部分是环境变量，有一部分是局部变量，这些变量保证了 shell 的正常运行

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 12 日
学 号	2021223281	姓 名	任可欣
专业班级	计算机科学与技术 06 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. 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 。

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. 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。

```
ren@DESKTOP-E80BM21: $ bash shell1
input:workspace1
shell1: line 4: unexpected EOF while looking for matching
shell1: line 10: syntax error: unexpected end of file
ren@DESKTOP-E80BM21:~$ cat >shell2
#!/bin/bash
echo -n "input:"
read dir
for i in `ls | grep -E "\.c"`
do
    mv $i $dir
done
ls -IS $dir
ren@DESKTOP-E80BM21:~$ chmod u+x shell2
ren@DESKTOP-E80BM21:~$ bash shell2
input:workspace1
mv: invalid option -- 'E'
Try 'mv --help' for more information.
test.c
```

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

```
ren@DESKTOP-E80BM21:~$ cd scripts
ren@DESKTOP-E80BM21:~/scripts$ cat>12.sh
#!/bin/bash
sum=0;
i=2;
while(($i<=20)); do
sum=$((sum+i));
((i=i+2));
done;
echo $sum;
ren@DESKTOP-E80BM21:~/scripts$ chmod u+x 12.sh
ren@DESKTOP-E80BM21:~/scripts$ bash 12.sh
110
ren@DESKTOP-E80BM21:~/scripts$
```

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
#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

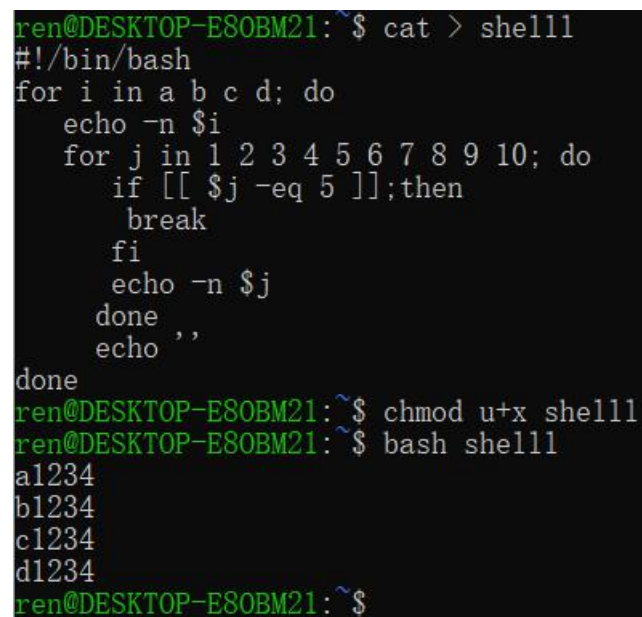
echo -n $j

done

echo ''

done
```

用break时结果为:



```
ren@DESKTOP-ESOBM21:~$ cat > shell11
#!/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
            break
        fi
        echo -n $j
    done
    echo ''
done
ren@DESKTOP-ESOBM21:~$ chmod u+x shell11
ren@DESKTOP-ESOBM21:~$ bash shell11
a1234
b1234
c1234
d1234
ren@DESKTOP-ESOBM21:~$
```

用break2结果为:

```

ren@DESKTOP-E80BM21: ~$ cat >5
#!/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
            break2
        fi
        echo -n $j
    done
    echo ' '
done
ren@DESKTOP-E80BM21: ~$ chmod u+x 5
ren@DESKTOP-E80BM21: ~$ bash 5
a12345: line 6: break2: command not found
5678910
b12345: line 6: break2: command not found
5678910
c12345: line 6: break2: command not found
5678910
d12345: line 6: break2: command not found
5678910

```

用continue结果为

```

ren@DESKTOP-E80BM21: ~$ cat >6
#!/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
            continue
        fi
        echo -n $j
    done
    echo ' '
done
ren@DESKTOP-E80BM21: ~$ chmod u+x 6
ren@DESKTOP-E80BM21: ~$ bash 6
a1234678910
b1234678910
c1234678910
d1234678910

```

用continue2结果为:

```
ren@DESKTOP-E80BM21:~$ cat >7
#!/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
            continue2
        fi
        echo -n $j
    done
    echo ', '
done
ren@DESKTOP-E80BM21:~$ chmod u+x 7
ren@DESKTOP-E80BM21:~$ bash 7
a12347: line 6: continue2: command not found
5678910
b12347: line 6: continue2: command not found
5678910
c12347: line 6: continue2: command not found
5678910
d12347: line 6: continue2: command not found
5678910
ren@DESKTOP-E80BM21:~$
```

二十、 实验过程分析与讨论

通过这次实验我学到了:

(1)for 语句的用法

for 变量名 in 取值列表

do

命令序列（命令行）

done

（2）until 语句的用法

until 条件测试操作

do

命令序列

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 4 月 19 日
学 号	2021223281	姓 名	任可欣
专业班级	计算机科学与技术 06 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；

```
ren@DESKTOP-E80BM21: $ cd scripts
ren@DESKTOP-E80BM21:~/scripts$ cat > rkxx
#!/bin/bash
funWithReturn() {
    echo "相加运算函数"
    echo "输入第一个数字:"
    read num1
    echo "输入第二个数字:"
    read num2
    val=`expr ${num1} + ${num2}`
    return ${var}
}
funWuthReturn
echo "输入的两数之和为: val!"
ren@DESKTOP-E80BM21:~/scripts$ chmod u+x rkxx
ren@DESKTOP-E80BM21:~/scripts$ bash rkxx
```

输入第一个数字:
1
输入第二个数字:
2
输入的两数之和为:3!

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；

```
ren@DESKTOP-E80BM21:~/scripts$ cat > rk
#!/bin/bash
echo "请输入"
read n
echo
func()
{
    local i="$1"
    if [ "$i" -eq 0 ]; then
        result=1
    else
        let "m=i-1"
        func "$m"
        let "result=$i * $?"
    fi
    return $result
}
func "$n"
echo "$n的阶乘为: $?"
ren@DESKTOP-E80BM21:~/scripts$ chmod u+x rk
ren@DESKTOP-E80BM21:~/scripts$ bash rk
请输入
3
9
```

3. 一个Shell脚本的内容如下所示：

```
#!/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
```

试运行该程序，并观察程序运行结果，理解函数嵌套的含义。

```
ren@DESKTOP-E80BM21:~$ cd scripts
ren@DESKTOP-E80BM21:~/scripts$ cat >rkxxx
#!/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
ren@DESKTOP-E80BM21:~/scripts$ chmod u+x rkxxx
ren@DESKTOP-E80BM21:~/scripts$ bash rkxxx
starting...
-1- here is in the first func.
-2- here is in the second func.
-3- here is in the third func.
ren@DESKTOP-E80BM21:~/scripts$
```

二十四、 实验过程分析与讨论

通过这次实验我了解到了：

shell 脚本就是由 Shell 命令组成的执行文件，将一些命令整合到一个文件中，进行处理业务逻辑，脚本不用编译即可运行。它通过解释器解释运行，所以速度相对来说比较慢。

shell 脚本中最重要的就是对 shell 命令的使用与组合，再使用 shell 脚本支持的一些语言特性，完成想要的功能。

+ 、 -、 *、 \ ： 乘号前必须加\进行转义才可以进行乘法运算

加法运算

val=`expr 2 + 2` （使用 linux 命令 expr 进行辅助运算）

val=\${2+2} （4 个空格不是必要的，不同于条件判断）

val=\$((2+2))

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 26 日
学 号	2021223281	姓 名	任可欣
专业班级	计算机科学与技术 06 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:

```
ren@DESKTOP-E80BM21: $ 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;

```
ren@DESKTOP-E80BM21: $ cat quote.txt | sed 's/\$/g'
The honeysuckle band played all night long for only 0.
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.
ren@DESKTOP-E80BM21: $
```

- (2) 显示包含 music 文字的行内容及行号;

```
ren@DESKTOP-E80BM21:~$ nl quote.txt | sed -n '/music/p'
2 It was an evening of splendid music and company.
ren@DESKTOP-E80BM21:~$
```

(3) 在第 4 行后面追加内容: "hello world!";

```
ren@DESKTOP-E80BM21:~$ sed -e 4a'hello world!' quote.txt
The honeysuckle band played all night long for only 0.
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";

```
ren@DESKTOP-E80BM21:~$ sed 's/The/Quod/g' quote.txt
Quod honeysuckle band played all night long for only 0.
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.";

```
ren@DESKTOP-E80BM21:~$ nl quote.txt | sed '3c This is the third line\.'
1 The honeysuckle band played all night long for only 0.
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.
ren@DESKTOP-E80BM21:~$
```

(6) 删除第 2 行内容;

```
ren@DESKTOP-E80BM21:~$ nl quote.txt | sed '2d'
1 The honeysuckle band played all night long for only 0.
3 Too bad the disco floor fell through at 23:10.
4 The local nurse Miss P.Neave was in attendance.
ren@DESKTOP-E80BM21:~$
```

(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。

```
ren@DESKTOP-E80BM21:~$ var='evening'
ren@DESKTOP-E80BM21:~$ echo $var
evening
ren@DESKTOP-E80BM21:~$
```

2. 文件 numbers.txt 的内容如下所示:

注: 每个冒号前后都有空格。


```
ren@DESKTOP-E80BM21: $ cat >numbers.txt
one : two : three
four : five : six
```

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；

```
ren@DESKTOP-E80BM21: $ awk -F ' ' '{print $2}' numbers.txt
:
:
:
ren@DESKTOP-E80BM21: ~ $
```

```
ren@DESKTOP-E80BM21: ~ $ 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。

```
ren@DESKTOP-E80BM21: $ cat >foo.txt
2 5 6
15 46 79
1 2 3
```

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。

```
ren@DESKTOP-E80BM21: $ cat >test
test i test a test
you test
ren@DESKTOP-E80BM21: $ cat >122
#!/bin/bash
read -p "enter search pattern: " pattern
awk "/$pattern/" '{ nmatches++; print } END { print nmatches, "found." }' info.txt
ren@DESKTOP-E80BM21: $ chmod u+x 122
ren@DESKTOP-E80BM21: $ bash 122
enter search pattern: test
awk: cmd. line:1: /test/{ nmatches++; print } END { print nmatches, "found." }
awk: cmd. line:1: syntax error
awk: cmd. line:1: /test/{ nmatches++; print } END { print nmatches, "found." }
awk: cmd. line:1:
svr
```

二十八、 实验过程分析与讨论

通过这次实验我学习到：

（1）sed

简介：利用脚本来处理文本文件

语法：sed [-hnV][[-e<script>][[-f<script 文件>]][文本文件]

（2）awk

简介：强大的文本分析工具

语法：awk [选项参数] 'script' var=value file(s) 或 awk [选项参数] -f scriptfile var=value file(s)

五、指导教师意见

指导教师签字：卢洋