

# 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	2021213619	姓 名	李小嵘
专业班级	计算机科学与技术 03 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) 云服务器腾讯云。

## 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

```
[cfy@VM-4-7-centos daily_practice]$ cd /etc
[cfy@VM-4-7-centos etc]$ pwd
/etc
[cfy@VM-4-7-centos 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。

```
[cfy@VM-4-7-centos etc]$ ls -al /home/cfy
total 156
drwx----- 12 cfy cfy 4096 May 4 21:41 .
drwxr-xr-x. 5 root root 4096 Apr 22 11:32 ..
drwxrwxr-x 5 cfy cfy 4096 Apr 3 11:49 aiXcoder
drwxr-xr-x 4 cfy cfy 4096 Apr 26 13:31 arduino15
-rw----- 1 cfy cfy 55638 May 6 21:20 .bash_history
-rw-r--r-- 1 cfy cfy 18 Apr 1 2020 .bash_logout
-rw-r--r-- 1 cfy cfy 193 Apr 1 2020 .bash_profile
-rw-r--r-- 1 cfy cfy 351 Mar 8 16:28 .bashrc
drwxrwxr-x 4 cfy cfy 4096 Apr 26 13:25 .cache
drwxrwxr-x 3 cfy cfy 4096 Mar 8 16:28 .config
lrwxrwxrwx 1 cfy cfy 48 Mar 8 16:28 .cquery -> /home/cfy/.VimForCpp/cquery/config/cquery.config
-rw-rw-r-- 1 cfy cfy 54 Mar 8 16:46 .gitconfig
-rw-rw-r-- 1 cfy cfy 827 Mar 8 16:27 install.sh
-rw----- 1 cfy cfy 35 Mar 14 15:02 .lessht
drwxrwxr-x 3 cfy cfy 4096 Mar 8 16:29 .lfCache
drwxrwxr-x 33 cfy cfy 4096 May 5 12:57 linux
drwx----- 3 cfy cfy 4096 Mar 8 16:29 .local
drwxrwxr-x 3 cfy cfy 4096 Mar 8 16:27 .pki
-rwxr-xr-x 1 cfy cfy 23360 May 4 21:40 udpClient
lrwxrwxrwx 1 cfy cfy 24 Mar 8 16:28 .vim -> /home/cfy/.VimForCpp/vim
drwxrwxr-x 8 cfy cfy 4096 Mar 8 16:28 .VimForCpp
-rw----- 1 cfy cfy 568 Mar 8 16:27 .viminfo
lrwxrwxrwx 1 cfy cfy 33 Mar 8 16:28 .vimrc -> /home/cfy/.VimForCpp/vim/init.vim
drwxrwxr-x 5 cfy cfy 4096 May 6 10:13 .vscode-server
lrwxrwxrwx 1 cfy cfy 38 Mar 8 16:28 .ycm_extra_conf.py -> /home/cfy/.VimForCpp/ycm_extra_conf.py
[cfy@VM-4-7-centos etc]$
```

3、使用命令创建目录/home/lyj/linux，然后删除该目录。

```
[cfy@VM-8-3-centos ~]$ ll
total 36
-rw-rw-r-- 1 cfy cfy 827 Oct 22 2022 install.sh
drwxrwxr-x 19 cfy cfy 4096 May 6 21:22 sbl
-rwxr-xr-x 1 cfy cfy 25040 Apr 27 14:12 udpClient
[cfy@VM-8-3-centos ~]$ mkdir linux
[cfy@VM-8-3-centos ~]$ ll
total 40
-rw-rw-r-- 1 cfy cfy 827 Oct 22 2022 install.sh
drwxrwxr-x 2 cfy cfy 4096 May 6 21:27 linux
drwxrwxr-x 19 cfy cfy 4096 May 6 21:22 sbl
-rwxr-xr-x 1 cfy cfy 25040 Apr 27 14:12 udpClient
[cfy@VM-8-3-centos ~]$ rmdir linux
[cfy@VM-8-3-centos ~]$ ll
total 36
-rw-rw-r-- 1 cfy cfy 827 Oct 22 2022 install.sh
drwxrwxr-x 19 cfy cfy 4096 May 6 21:22 sbl
-rwxr-xr-x 1 cfy cfy 25040 Apr 27 14:12 udpClient
[cfy@VM-8-3-centos ~]$
```

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

```
[cfy@VM-8-3-centos ~]$ echo "Hello, Linux" > abc
[cfy@VM-8-3-centos ~]$ cat abc
Hello, Linux
[cfy@VM-8-3-centos ~]$
```

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。

```
[cfy@VM-8-3-centos ~]$ ll
total 44
-rw-rw-r-- 1 cfy cfy 13 May 6 21:31 abc
drwxrwxr-x 2 cfy cfy 4096 May 6 21:33 ak
-rw-rw-r-- 1 cfy cfy 827 Oct 22 2022 install.sh
drwxrwxr-x 19 cfy cfy 4096 May 6 21:22 sbl
-rwxr-xr-x 1 cfy cfy 25040 Apr 27 14:12 udpClient
[cfy@VM-8-3-centos ~]$ cp abc ak/
[cfy@VM-8-3-centos ~]$ ll ak/
total 4
-rw-rw-r-- 1 cfy cfy 13 May 6 21:33 abc
[cfy@VM-8-3-centos ~]$
```

```
[cfy@VM-8-3-centos ~]$ rm -rf ak
[cfy@VM-8-3-centos ~]$ ll
total 40
-rw-rw-r-- 1 cfy cfy 13 May 6 21:31 abc
-rw-rw-r-- 1 cfy cfy 827 Oct 22 2022 install.sh
drwxrwxr-x 19 cfy cfy 4096 May 6 21:22 sbl
-rwxr-xr-x 1 cfy cfy 25040 Apr 27 14:12 udpClient
[cfy@VM-8-3-centos ~]$
```

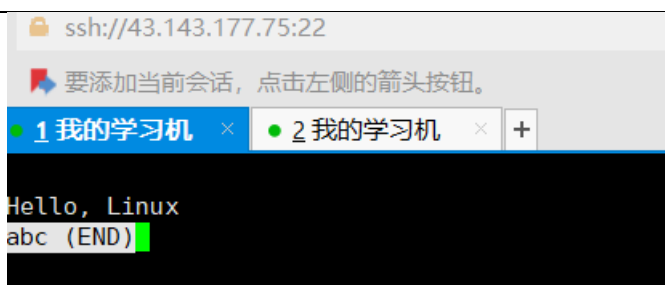
6、查看文件/etc/adduser.conf 的前 3 行内容， 查看文件/etc/adduser.conf 的最后 5 行内容。

```
[cfy@VM-8-3-centos ~]$ head -3 /etc/subgid
lighthouse:100000:65536
caixiaohuatongxue:165536:65536
thj:231072:65536
[cfy@VM-8-3-centos ~]$
-----
[cfy@VM-8-3-centos ~]$ tail -5 /etc/vimrc
endif

" Don't wake up system with blinking cursor:
" http://www.linuxpowertop.org/known.php
let &guicursor = &guicursor . ",a:blinkon0"
[cfy@VM-8-3-centos ~]$
```

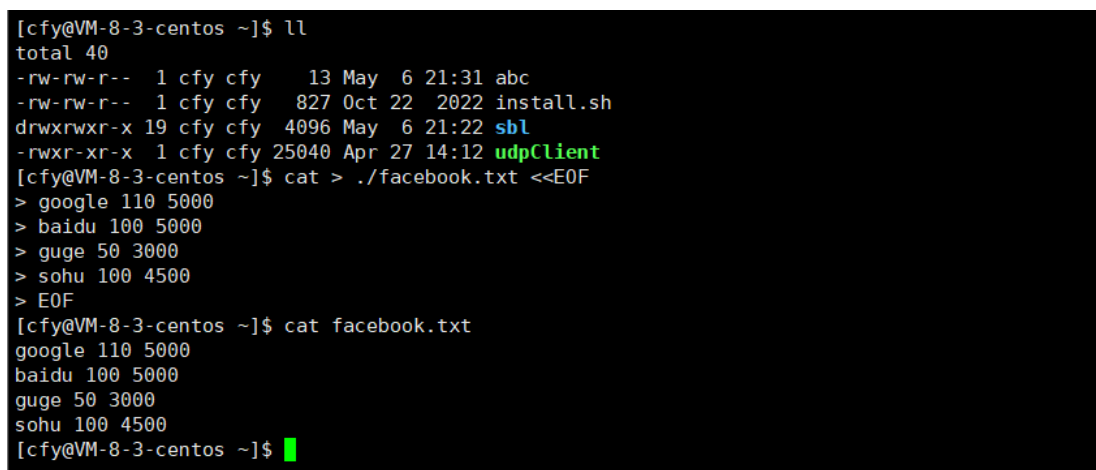
7、分屏查看文件/etc/adduser.conf 的内容。

```
[cfy@VM-8-3-centos ~]$ less abc
```



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
```



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

```
[cfy@VM-8-3-centos ~]$ sort -r facebook.txt
sohu 100 4500
guge 50 3000
google 110 5000
baidu 100 5000
[cfy@VM-8-3-centos ~]$
```

(2) 按公司人数排序

```
[cfy@VM-8-3-centos ~]$ sort -k2n facebook.txt
guge 50 3000
baidu 100 5000
sohu 100 4500
google 110 5000
[cfy@VM-8-3-centos ~]$
```

(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

```
[cfy@VM-8-3-centos ~]$ sort -k2n -k3nr facebook.txt
guge 50 3000
baidu 100 5000
sohu 100 4500
google 110 5000
[cfy@VM-8-3-centos ~]$
```

(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

```
[cfy@VM-8-3-centos ~]$ sort -k3nr -k2n facebook.txt
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。

```
[cfy@VM-8-3-centos ~]$ sort -k1.2 facebook.txt
baidu 100 5000
sohu 100 4500
google 110 5000
guge 50 3000
[cfy@VM-8-3-centos ~]$
```

#### 四、 实验过程分析与讨论

对于不知道的指令选项，用 ChatGPT 查询，比如排序的选项先后顺序。

#### 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	2021213619	姓 名	李小嵘
专业班级	计算机科学与技术 03 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) 云服务器腾讯云。

## 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；

```
[cfy@VM-8-3-centos ~]$ mkdir baz
[cfy@VM-8-3-centos ~]$ cd baz
[cfy@VM-8-3-centos baz]$ cat > qux << EOF
> 123
> 456
> abc
> EOF
[cfy@VM-8-3-centos baz]$ █
```

(2) 在**用户目录**下查找**文件** qux ，并显示该**文件**位置信息；

```
[cfy@VM-8-3-centos ~]$ find ~ -name qux
/home/cfy/baz/qux
[cfy@VM-8-3-centos ~]$
```

(3) 统计**文件** qux 中所包含内容的**行数**、**字数**和**字节数**；

```
[cfy@VM-8-3-centos baz]$ wc qux
 3  3 12 qux
[cfy@VM-8-3-centos baz]$
```

(4) 在**用户目录**下查找**文件** qux ，并删除该**文件**；

```
[cfy@VM-8-3-centos baz]$ find ~ -name qux -delete
[cfy@VM-8-3-centos baz]$ ll
total 0
[cfy@VM-8-3-centos baz]$
```

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

```
[cfy@VM-8-3-centos baz]$ find ~ -name qux -delete
[cfy@VM-8-3-centos baz]$ ll
total 0
[cfy@VM-8-3-centos 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 ；

```
[cfy@VM-8-3-centos ~]$ mkdir path1
[cfy@VM-8-3-centos ~]$ cd path1/
[cfy@VM-8-3-centos path1]$ touch file1 file2
[cfy@VM-8-3-centos path1]$ ll
total 0
-rw-rw-r-- 1 cfy cfy 0 May  7 14:12 file1
-rw-rw-r-- 1 cfy cfy 0 May  7 14:12 file2
[cfy@VM-8-3-centos path1]$
```

(2) 在**用户目录**下新建**文件夹** path2 ，在 path2 下新建**文件** file3 ；

```
[cfy@VM-8-3-centos ~]$ mkdir path2; cd path2; touch file3; ll
total 0
-rw-rw-r-- 1 cfy cfy 0 May  7 14:14 file3
[cfy@VM-8-3-centos path2]$
```

(3) 在**用户目录**下新建文件 file4 ；

```
[cfy@VM-8-3-centos ~]$ touch file4
[cfy@VM-8-3-centos ~]$ ll
total 56
-rw-rw-r-- 1 cfy cfy 13 May 6 21:31 abc
drwxrwxr-x 2 cfy cfy 4096 May 7 14:10 baz
-rw-rw-r-- 1 cfy cfy 58 May 7 13:15 facebook.txt
-rw-rw-r-- 1 cfy cfy 0 May 7 14:14 file4
-rw-rw-r-- 1 cfy cfy 827 Oct 22 2022 install.sh
drwxrwxr-x 2 cfy cfy 4096 May 7 14:12 path1
drwxrwxr-x 2 cfy cfy 4096 May 7 14:14 path2
drwxrwxr-x 19 cfy cfy 4096 May 6 21:22 sbl
-rwxr-xr-x 1 cfy cfy 25040 Apr 27 14:12 udpClient
[cfy@VM-8-3-centos ~]$
```

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

```
[cfy@VM-8-3-centos ~]$ tar -cvf package.tar path1 file4
path1/
path1/file2
path1/file1
file4
[cfy@VM-8-3-centos ~]$
```

(5) 查看包 package.tar 的内容；

```
[cfy@VM-8-3-centos ~]$ tar -tvf package.tar
drwxrwxr-x cfy/cfy 0 2023-05-07 14:12 path1/
-rw-rw-r-- cfy/cfy 0 2023-05-07 14:12 path1/file2
-rw-rw-r-- cfy/cfy 0 2023-05-07 14:12 path1/file1
-rw-rw-r-- cfy/cfy 0 2023-05-07 14:14 file4
[cfy@VM-8-3-centos ~]$
```

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

```
[cfy@VM-8-3-centos ~]$ tar -rvf package.tar path2
path2/
path2/file3
[cfy@VM-8-3-centos ~]$
```

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

```
[cfy@VM-8-3-centos ~]$ mkdir path3
[cfy@VM-8-3-centos ~]$ cp package.tar path3
[cfy@VM-8-3-centos ~]$
```

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

```
[cfy@VM-8-3-centos ~]$ cd path3
[cfy@VM-8-3-centos path3]$ tar -xvf package.tar
path1/
path1/file2
path1/file1
file4
path2/
path2/file3
[cfy@VM-8-3-centos path3]$
```

### 3. 符号链接内容

(1) 新建文件 foo.txt ，内容为 123 ；

```
[cfy@VM-8-3-centos ~]$ echo "123" > foo.txt
[cfy@VM-8-3-centos ~]$
```

(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；

```
[cfy@VM-8-3-centos ~]$ ln foo.txt bar.txt
[cfy@VM-8-3-centos ~]$ comm foo.txt bar.txt
123
[cfy@VM-8-3-centos ~]$ diff foo.txt bar.txt
[cfy@VM-8-3-centos ~]$
```

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

```
[cfy@VM-8-3-centos ~]$ ll -i foo.txt bar.txt
666499 -rw-rw-r-- 2 cfy cfy 4 May  7 14:23 bar.txt
666499 -rw-rw-r-- 2 cfy cfy 4 May  7 14:23 foo.txt
[cfy@VM-8-3-centos ~]$
```

(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；

```
[cfy@VM-8-3-centos ~]$ echo "abc" > bar.txt
[cfy@VM-8-3-centos ~]$ diff foo.txt bar.txt
[cfy@VM-8-3-centos ~]$
```

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

```
[cfy@VM-8-3-centos ~]$ rm foo.txt
[cfy@VM-8-3-centos ~]$ ll -i bar.txt
666499 -rw-rw-r-- 1 cfy cfy 4 May  7 14:33 bar.txt
[cfy@VM-8-3-centos ~]$ cat bar.txt
abc
[cfy@VM-8-3-centos ~]$
```

(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；

```
[cfy@VM-8-3-centos ~]$ ln -s bar.txt baz.txt
[cfy@VM-8-3-centos ~]$ ls -i bar.txt baz
666499 bar.txt

baz:
[cfy@VM-8-3-centos ~]$ diff bar.txt baz
baz/      baz.txt
[cfy@VM-8-3-centos ~]$ diff bar.txt baz.txt
[cfy@VM-8-3-centos ~]$
```

## 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。

### 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）

```
[cfy@VM-8-3-centos ~]$ touch qux.txt
[cfy@VM-8-3-centos ~]$ chmod +x qux.txt
[cfy@VM-8-3-centos ~]$ ll
total 88
-rw-rw-r-- 1 cfy cfy 13 May 6 21:31 abc
-rw-rw-r-- 1 cfy cfy 4 May 7 14:33 bar.txt
drwxrwxr-x 2 cfy cfy 4096 May 7 14:10 baz
lrwxrwxrwx 1 cfy cfy 7 May 7 14:49 baz.txt -> bar.txt
-rw-rw-r-- 1 cfy cfy 58 May 7 13:15 facebook.txt
-rw-rw-r-- 1 cfy cfy 0 May 7 14:14 file4
-rw-rw-r-- 1 cfy cfy 827 Oct 22 2022 install.sh
-rw-rw-r-- 1 cfy cfy 10240 May 7 14:20 package.tar
-rw-rw-r-- 1 cfy cfy 10240 May 7 14:20 paclear
drwxrwxr-x 2 cfy cfy 4096 May 7 14:12 path1
drwxrwxr-x 2 cfy cfy 4096 May 7 14:14 path2
drwxrwxr-x 4 cfy cfy 4096 May 7 14:22 path3
-rwxrwxr-x 1 cfy cfy 0 May 7 14:50 qux.txt
drwxrwxr-x 19 cfy cfy 4096 May 6 21:22 sbl
-rwxr-xr-x 1 cfy cfy 25040 Apr 27 14:12 udpClient
[cfy@VM-8-3-centos ~]$
```

#### 四、 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。

## 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	2021213619	姓 名	李小嵘
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

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) 云服务器腾讯云

## 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 ；

```
[cfy@VM-8-3-centos ~]$ mkdir workspace1  
[cfy@VM-8-3-centos ~]$ cd workspace1/  
[cfy@VM-8-3-centos workspace1]$
```

- (2) 进入目录 workspace1 ；

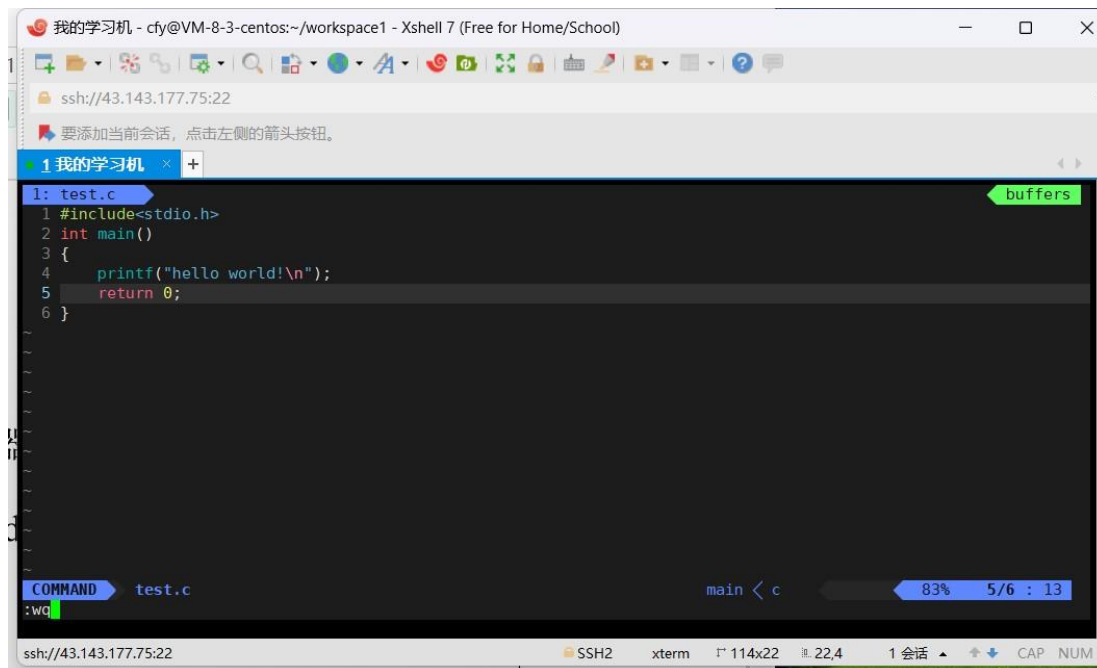
```
[cfy@VM-8-3-centos ~]$ mkdir workspace1  
[cfy@VM-8-3-centos ~]$ cd workspace1/  
[cfy@VM-8-3-centos workspace1]$
```

- (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; }
```

```
[cfy@VM-8-3-centos ~]$ mkdir workspace1
[cfy@VM-8-3-centos ~]$ cd workspace1/
[cfy@VM-8-3-centos workspace1]$ vim test.c
```

(4) 保存 test.c 的内容，并退出；



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

```
[cfy@VM-8-3-centos workspace1]$ gcc test.c
[cfy@VM-8-3-centos workspace1]$ ./a.out
hello world!
[cfy@VM-8-3-centos workspace1]$
```

## 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；

```
[cfy@VM-8-3-centos ~]$ mkdir workspace2
```

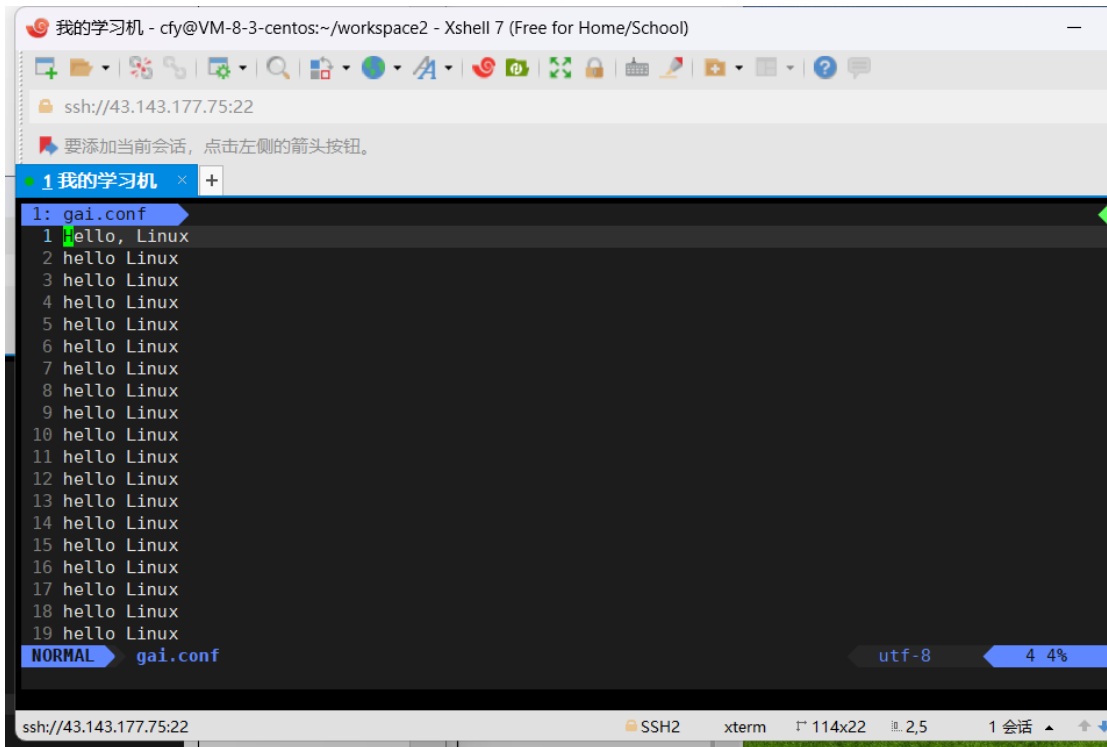
(2) 进入 workspace2 目录；

```
[cfy@VM-8-3-centos ~]$ cd workspace2
[cfy@VM-8-3-centos workspace2]$
```

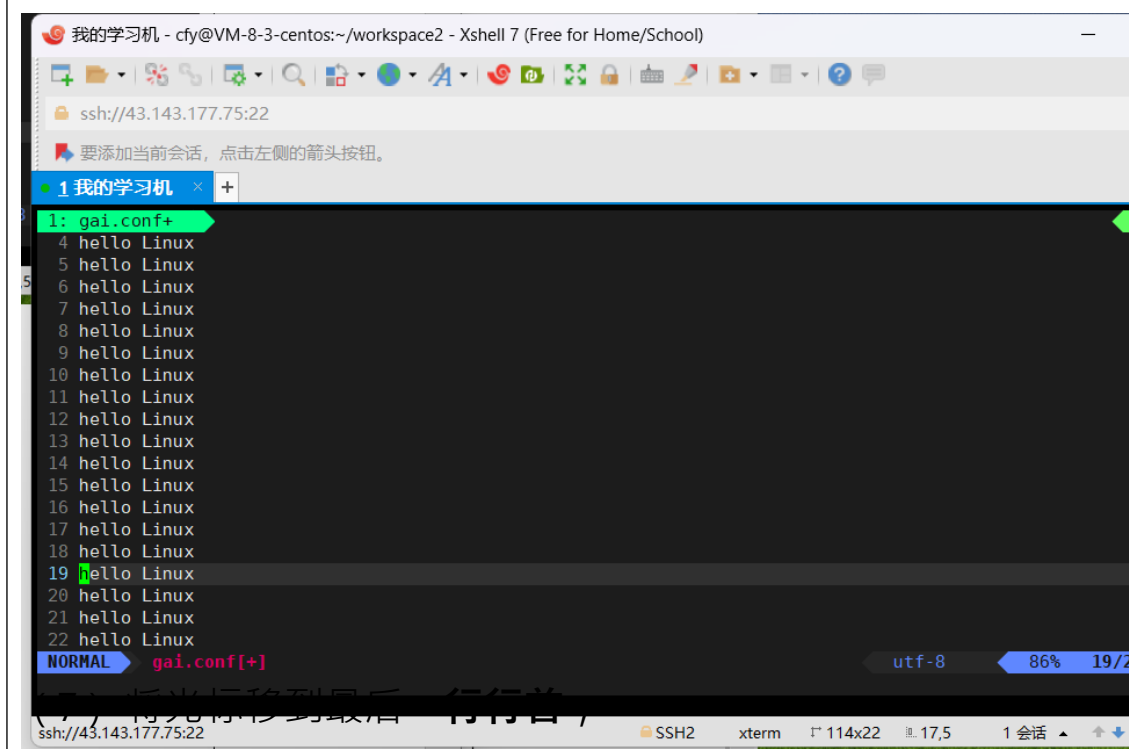
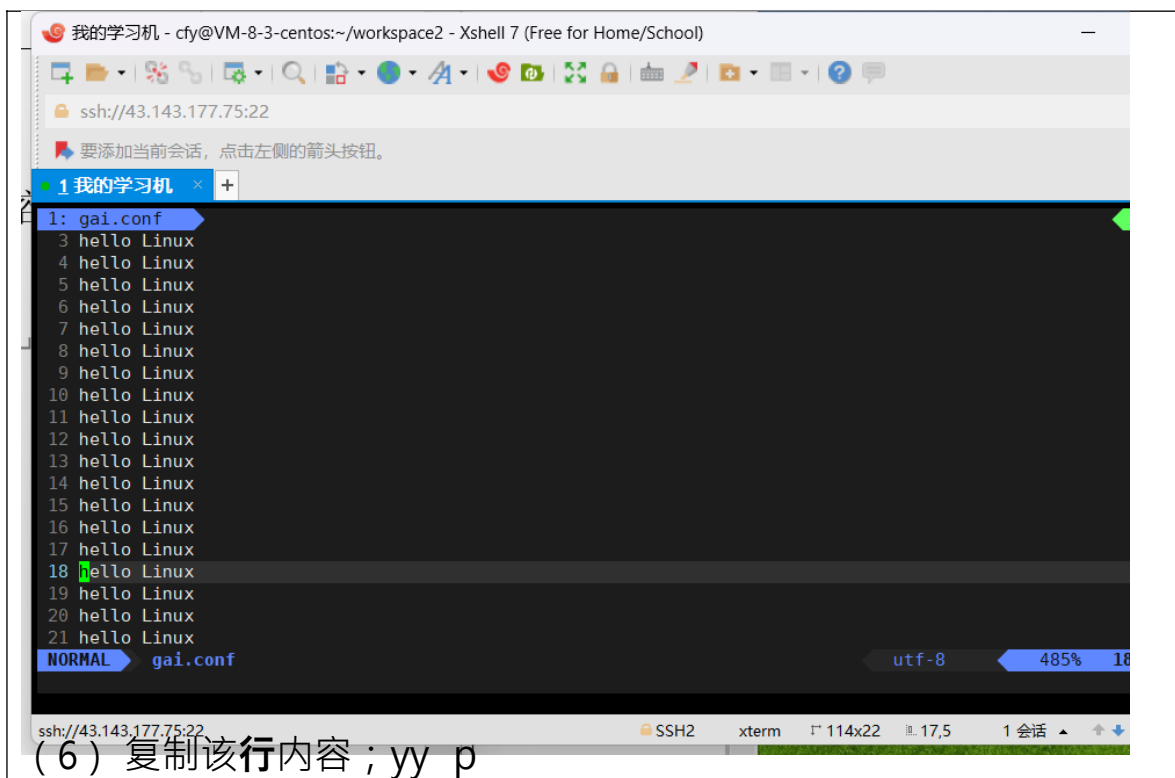
(3) 使用以下命令：将文件 `/etc/gai.conf` 的内容复制到当前目录下的新建文件 `gai.conf` 中；

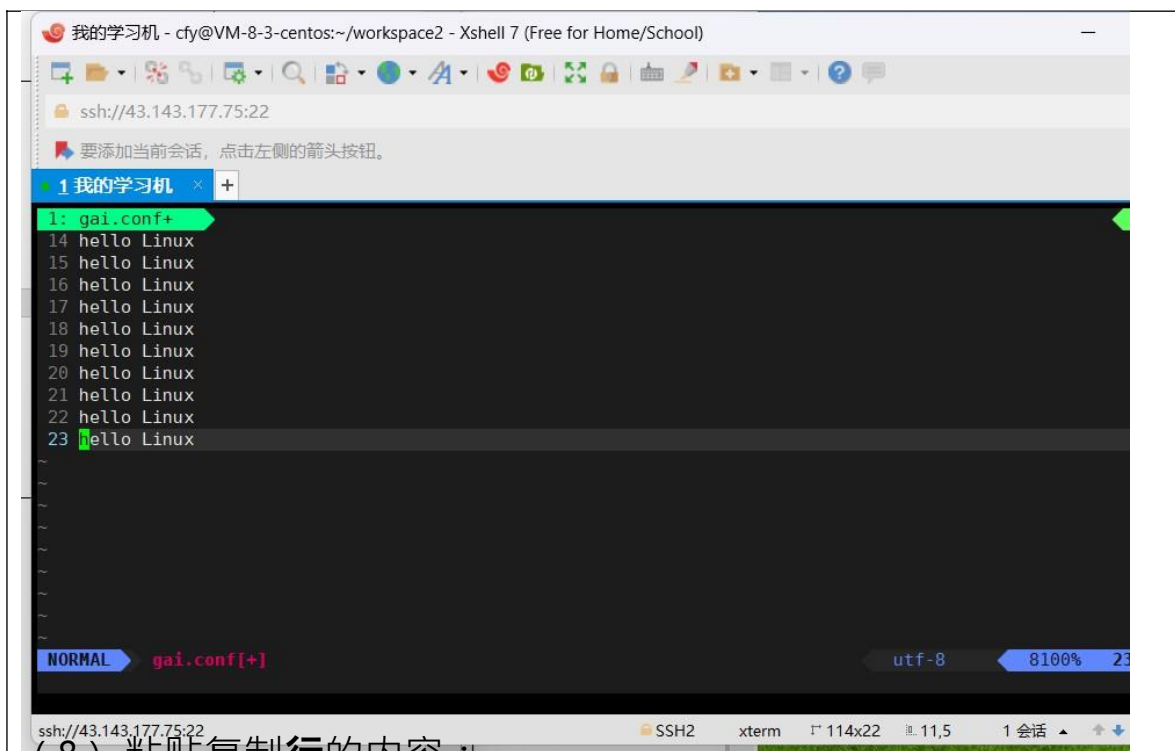
```
[cfy@VM-8-3-centos workspace2]$ cat ../etc/gai.conf > gai.conf
```

(4) 使用 `vim` 编辑当前目录下的 `gai.conf` ；

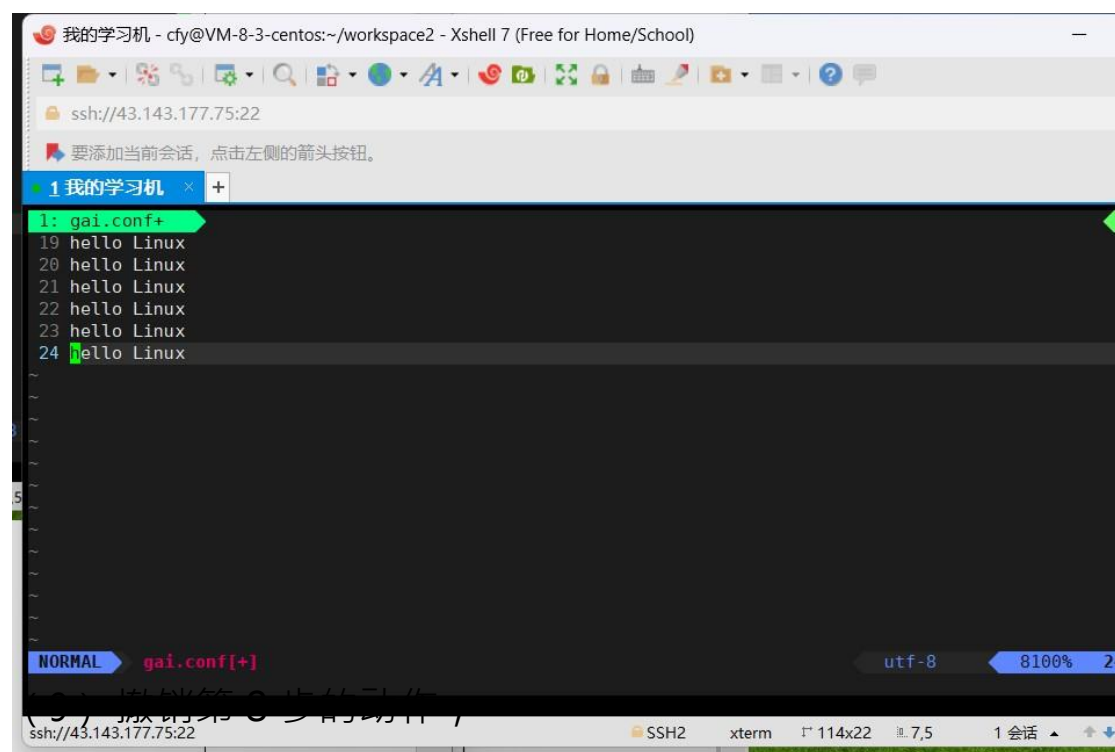


(5) 将光标移到第 18 行；





(8) 粘贴复制行的内容；



(9) 撤销第⑧步的动作；

我的学习机 - cfy@VM-8-3-centos:~/workspace2 - Xshell 7 (Free for Home/School)

ssh://43.143.177.75:22

要添加当前会话，点击左侧的箭头按钮。

1 我的学习机

```
1: gai.conf+
18 hello Linux
19 hello Linux
20 hello Linux
21 hello Linux
22 hello Linux
23 hello Linux
```

NORMAL gai.conf[+] utf-8 8100% 23

1 line less; before #3 15 seconds ago

ssh://43.143.177.75:22 SSH2 xterm 114x22 7,5 1 会话

U (10) 存盘但不退出；

我的学习机 - cfy@VM-8-3-centos:~/workspace2 - Xshell 7 (Free for Home/School)

ssh://43.143.177.75:22

要添加当前会话，点击左侧的箭头按钮。

1 我的学习机

```
1: gai.conf+
18 hello Linux
19 hello Linux
20 hello Linux
21 hello Linux
22 hello Linux
23 hello Linux
```

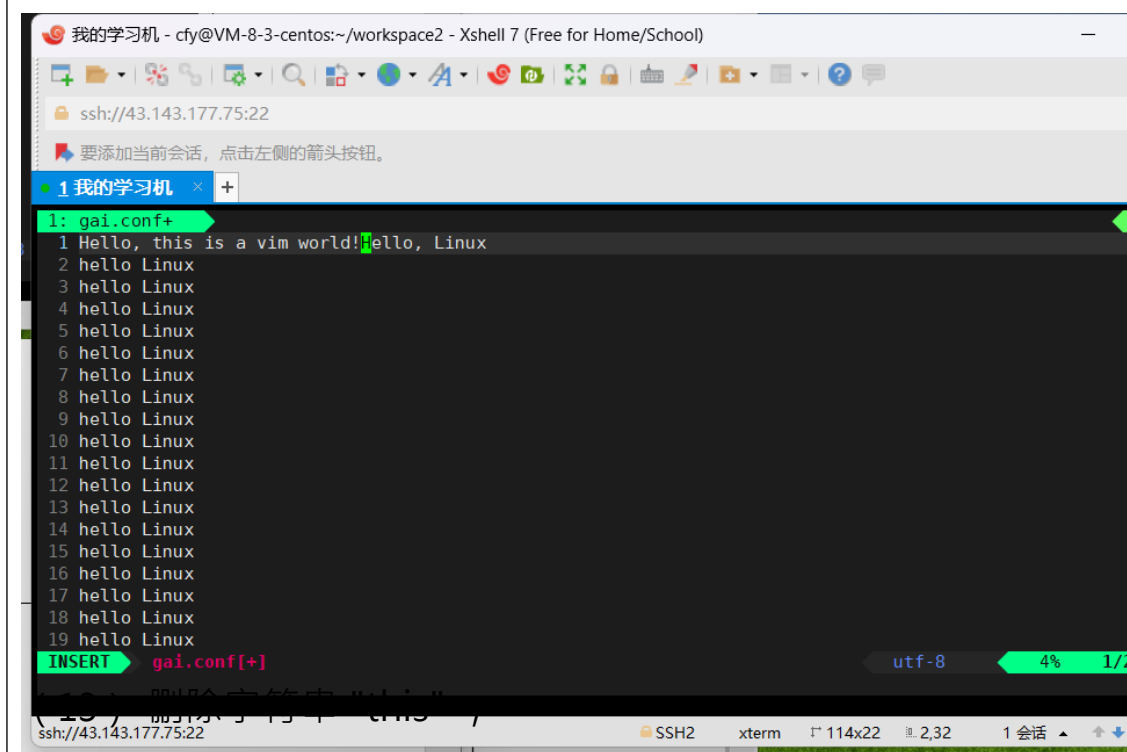
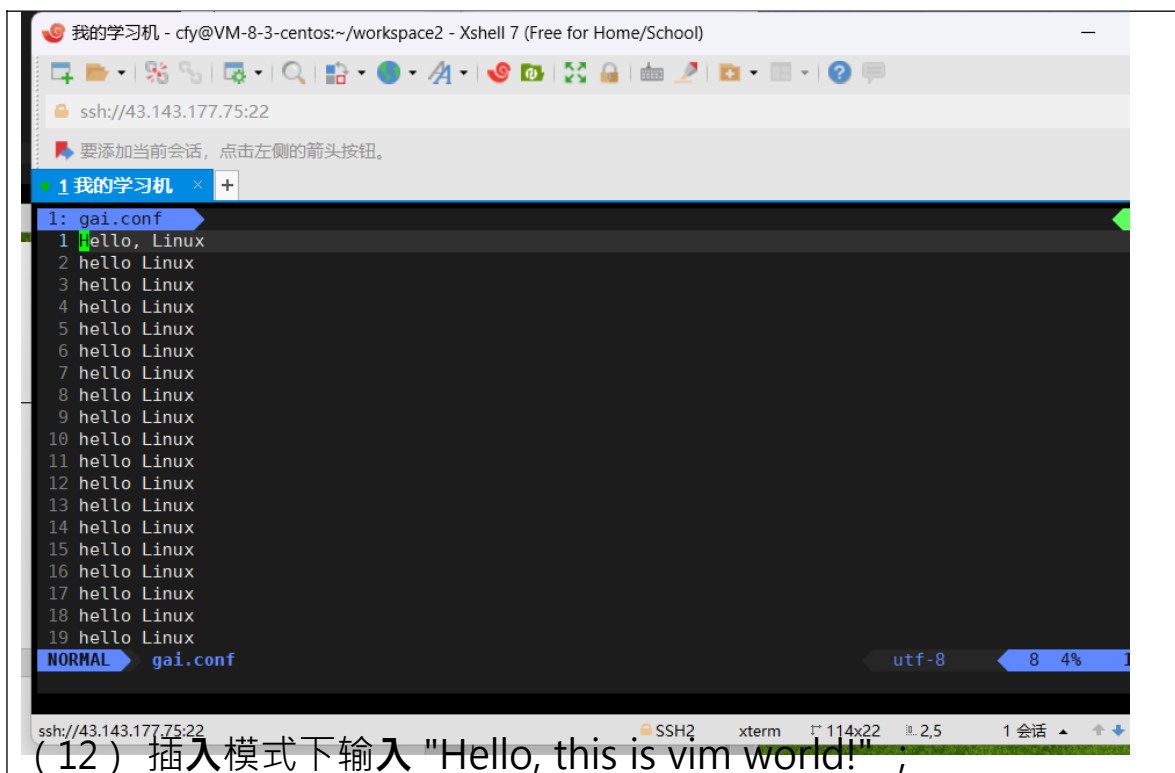
COMMAND gai.conf[+] utf-8 8100% 23

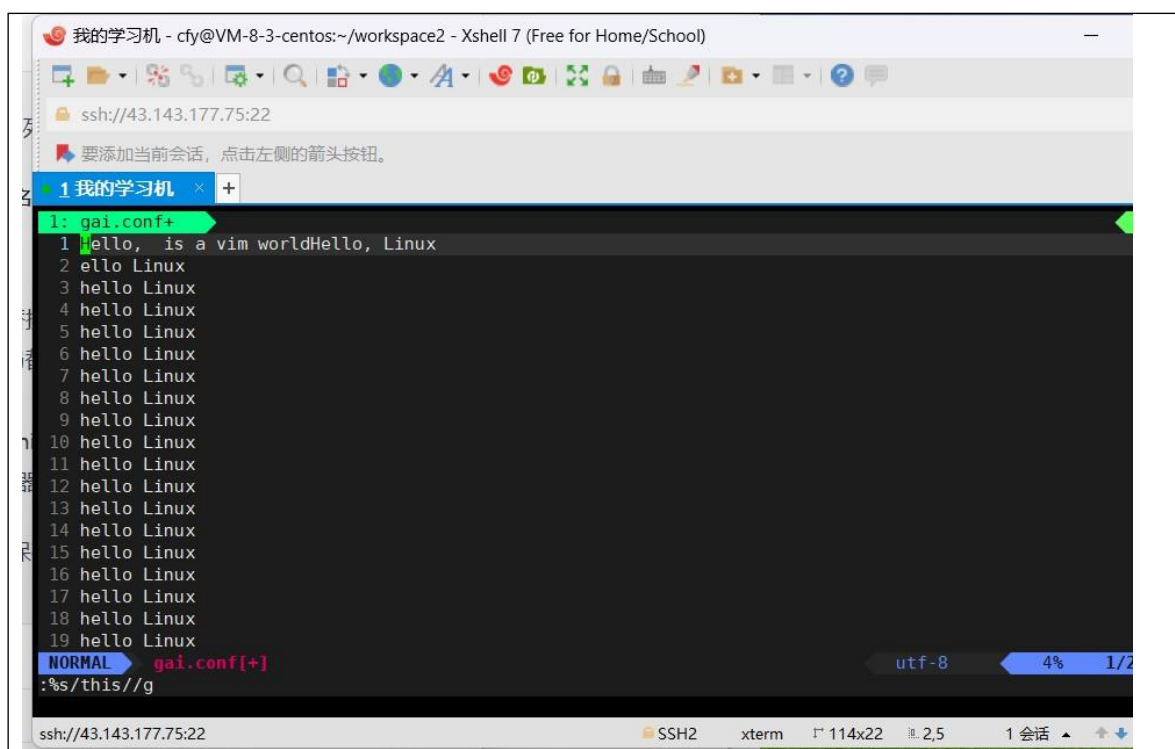
1 line less; before #3 15 seconds ago

ssh://43.143.177.75:22 SSH2 xterm 114x22 22,3 1 会话

(11) 将光标移到首行；

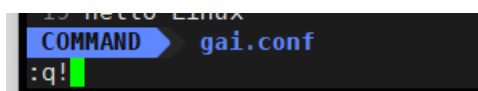
99





```
我的学习机 - cfy@VM-8-3-centos:~/workspace2 - Xshell 7 (Free for Home/School)
ssh://43.143.177.75:22
要添加当前会话，点击左侧的箭头按钮。
1 我的学习机
1: gai.conf+
1 Hello, is a vim worldHello, Linux
2 ello Linux
3 hello Linux
4 hello Linux
5 hello Linux
6 hello Linux
7 hello Linux
8 hello Linux
9 hello Linux
10 hello Linux
11 hello Linux
12 hello Linux
13 hello Linux
14 hello Linux
15 hello Linux
16 hello Linux
17 hello Linux
18 hello Linux
19 hello Linux
NORMAL gai.conf[+]
:s/this//g
ssh://43.143.177.75:22 SSH2 xterm 114x22 2,5 1 会话
```

( 14 ) 强制退出 vim ，不存盘。



```
19 hello Linux
COMMAND gai.conf
:q!
```



#### 四、实验过程分析与讨论

完成了 vim 的基本操作。

#### 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	2021213619	姓 名	李小嵘
专业班级	计算机科学与技术 03 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) 云服务器腾讯云

## 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，并设置登录口令为 `123456`；

```
[cfy@VM-8-3-centos ~]$ sudo useradd -m -s /bin/sh -p $(openssl passwd -1 123456) -c "bar" foo
[sudo] password for cfy:
[cfy@VM-8-3-centos ~]$
```

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`；

```
[cfy@VM-8-3-centos ~]$ sudo su - foo
-sh-4.2$ sudo usermod -u 2000 -s /bin/csh foo
-sh-4.2$ sudo usermod -u 2000 -s /bin/csh foo
[sudo] password for foo:
usermod: user foo is currently used by process 9619
-sh-4.2$
```

### 3. 从用户 foo 切换到 root

```
-sh-4.2$ su
Password:
[root@VM-8-3-centos 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

```
[root@VM-8-3-centos ~]# userdel -r foo
userdel: user foo is currently used by process 9619
[root@VM-8-3-centos ~]#
```

### 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@VM-8-3-centos ~]# ll
total 0
[root@VM-8-3-centos ~]# vim users.txt
[root@VM-8-3-centos ~]# newusers < user.txt
bash: user.txt: No such file or directory
[root@VM-8-3-centos ~]# newusers < users.txt
[root@VM-8-3-centos ~]# pwunconv
[root@VM-8-3-centos ~]# vim passwds.txt
[root@VM-8-3-centos ~]# chpasswd < passwds.txt
[root@VM-8-3-centos ~]# pwconv
[root@VM-8-3-centos ~]#
```

```
[root@VM-8-3-centos ~]# ll /home
total 72
drwx----- 17 cfy          cfy          4096 May  7 16:17 cfy
drwx-----  2 foo          foo          4096 May  7 16:02 foo
drwx----- 13 hwc          hwc          4096 Apr 12 11:21 hwc
drwx----- 10 hzh          hzh          4096 Apr 22 14:26 hzh
drwx-----  2 liaodacong  liaodacong  4096 Sep  7 2022 liaodacong
drwx-----  5 LiaoDaCong  LiaoDaCong  4096 Sep 25 2022 LiaoDaCong
drwx-----  3 lighthouse  lighthouse  4096 Sep  6 2022 lighthouse
drwx-----  4 LZxianfeng  LZxianfeng  4096 Sep 30 2022 LZxianfeng
drwx-----  2 Test        Test        4096 Apr  4 20:54 Test
drwx----- 13 thj          thj          4096 Apr 20 20:17 thj
drwx-----  2 user001      users       4096 May  7 16:34 user001
drwx-----  2 user002      users       4096 May  7 16:34 user002
drwx-----  2 user003      users       4096 May  7 16:34 user003
drwx-----  2 user004      users       4096 May  7 16:34 user004
drwx-----  2 user005      users       4096 May  7 16:34 user005
drwx-----  2 user006      users       4096 May  7 16:34 user006
drwx-----  2 caixiaohuatongxue caixiaohuatongxue 4096 May  7 16:28 username
drwx----- 12 yaozheng     yaozheng    4096 Mar 22 10:03 yaozheng
[root@VM-8-3-centos ~]#
```

6. 创建用户组 group1，并在创建时设置其 GID 为 3000.

```
[cfy@VM-8-3-centos ~]$ sudo groupadd -g 3000 group1
[cfy@VM-8-3-centos ~]$ cat /etc/group | grep group1
group1:x:3000:
[cfy@VM-8-3-centos ~]$
```

7. 在用户组 group1 中添加两个之前批量创建的用户

```
[cfy@VM-8-3-centos ~]$ sudo usermod -a -G group1 user001
[cfy@VM-8-3-centos ~]$ sudo usermod -a -G group1 user002
[cfy@VM-8-3-centos ~]$
```

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

```
[cfy@VM-8-3-centos ~]$ sudo -u user001 -i
-bash-4.2$ sudo cat /etc/shadow

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for user001:
Sorry, try again.
[sudo] password for user001:
user001 is not in the sudoers file. This incident will be reported.
-bash-4.2$
```

#### 四、 实验过程分析与讨论

对于这个创建用户组，一开始创建无效，于是我用 ChatGPT 进行搜索并解决了这个问题。

#### 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	2021213619	姓 名	李小嵘
专业班级	计算机科学与技术 03 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) 云服务器腾讯云

## 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**）；

```
[cfy@VM-8-3-centos ~]$ foo=200
[cfy@VM-8-3-centos ~]$ echo $foo
200
[cfy@VM-8-3-centos ~]$ █
```

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**）；

```
[cfy@VM-8-3-centos ~]$ bar=100
[cfy@VM-8-3-centos ~]$ test $bar -gt 150
[cfy@VM-8-3-centos ~]$ echo $?
1
[cfy@VM-8-3-centos ~]$ █
```



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）；

```
1: myscript.sh+
1 echo "Hostname: $(hostname)"
2 echo "Date:$(date)"

[cfy@VM-8-3-centos ~]$ vim myscript.sh
[cfy@VM-8-3-centos ~]$ sh myscript.sh
Hostname: VM-8-3-centos
Date:Sun May 7 16:58:32 CST 2023
[cfy@VM-8-3-centos ~]$
```

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；

```
1: myscript.sh buffers
1 #!/bin/bash
2
3 # 获取输入参数
4 num=$1
5
6 #判断是否为水仙花数
7 n=num
8 sum=0
9 while[ $n -gt 0 ]
10 do
11     m=$((n%10))
12     sum=$((sum + m*m*m))
13     n=$((n/10))
14 done
15
16 if[ $sum -eq $num ]
17 then
18     echo "$num 是水仙花数"
19 else
20     echo "$num 不是水仙花数"
21 fi

[cfy@VM-8-3-centos ~]$ vim myscript.sh
[cfy@VM-8-3-centos ~]$ chmod +x myscript.sh

[cfy@VM-8-3-centos ~]$ ./myscript.sh 153
153 是水仙花数
[cfy@VM-8-3-centos ~]$
```

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；

```
1: sum.sh
1 #!/bin/bash
2
3 # 获取输入参数
4 num1=$1
5 num2=$2
6 num3=$3
7
8 # 计算和并输出
9 sum=$((num1 + num2 + num3))
10 echo "The sum of the three numbers is: $sum"
11
```

```
[cfy@VM-8-3-centos ~]$ vim sum.sh
[cfy@VM-8-3-centos ~]$ chmod +x sum.sh
[cfy@VM-8-3-centos ~]$ ./sum.sh 1 2 3
The sum of the three numbers is: 6
[cfy@VM-8-3-centos ~]$
```

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

```
if
elif
else
fi 实现
```

```
1: grade.sh+
1 #!/bin/bash
2
3 echo "Please enter your score:"
4 read score
5
6 if [ $score -ge 90 ]; then
7     echo "Your grade is A."
8 elif [ $score -ge 80 ]; then
9     echo "Your grade is B."
10 elif [ $score -ge 70 ]; then
11     echo "Your grade is C."
12 elif [ $score -ge 60 ]; then
13     echo "Your grade is D."
14 else
15     echo "Your grade is E."
16 fi
17
```

```
[cfy@VM-8-3-centos ~]$ vim grade.sh
[cfy@VM-8-3-centos ~]$ chmod +x grade.sh
[cfy@VM-8-3-centos ~]$ ./grade.sh
Please enter your score:
65
Your grade is D.
[cfy@VM-8-3-centos ~]$
```

#### 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。

[ -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 3 月 8 日
学 号	2021213619	姓 名	李小嵘
专业班级	计算机科学与技术 03 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) 云服务器腾讯云

## 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 ) ;

```
1: myscript.sh  ↵
1 target_dir=~/.workspace/
2 for file in *.c;do
3     cp "$file" "$target_dir"
4 done
5
6 echo "所有.c文件已经成功复制到$target_dir"
~
~

[cfy@VM-8-3-centos ~]$ mkdir workspace
[cfy@VM-8-3-centos ~]$ sh myscript.sh
所有.c文件已经成功复制到/home/cfy/workspace/
[cfy@VM-8-3-centos ~]$
```

2. 编写 Shell 脚本, 利用 while 循环求前十个偶数之和, 并输

出结果

```
1: myscript4.sh
1 n=1
2 sum=0
3
4 while((n<=20))
5 do
6     if((n%2 == 0))
7     then
8         sum=$((sum+n))
9     fi
10
11     n=$((n+1))
12 done
13
14 echo "前10个偶数之和为: $sum"

[cfy@VM-8-3-centos ~]$ vim myscript4.sh
[cfy@VM-8-3-centos ~]$ sh myscript4.sh
前10个偶数之和为: 110
[cfy@VM-8-3-centos ~]$
```

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

出结果

```
1: myscript5.sh
1 n=1
2 sum=0
3
4 until((n>10))
5 do
6     square=$((n*n))
7     sum=$((sum+square))
8
9     n=$((n+1))
10 done
11
12 echo "1到10的平方和为: $sum"

[cfy@VM-8-3-centos ~]$ vim myscript5.sh
[cfy@VM-8-3-centos ~]$ sh myscript5.sh
1到10的平方和为: 385
[cfy@VM-8-3-centos ~]$
```

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、continue2，并观察四种情况下的实验



## 结果

```
[cfy@VM-8-3-centos ~]$ sh myscript6.sh
a1234
b1234
c1234
d1234
[cfy@VM-8-3-centos ~]$
```

```
[cfy@VM-8-3-centos ~]$ sh myscript6.sh
a1234[cfy@VM-8-3-centos ~]$
```

```
a1234[cfy@VM-8-3-centos ~]$ vim myscript6.sh
[cfy@VM-8-3-centos ~]$ sh myscript6.sh
a1234678910
b1234678910
c1234678910
d1234678910
[cfy@VM-8-3-centos ~]$
```

```
[cfy@VM-8-3-centos ~]$ vim myscript6.sh
[cfy@VM-8-3-centos ~]$ sh myscript6.sh
a1234b1234c1234d1234[cfy@VM-8-3-centos ~]$
```

## 四、 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 4 月 19 日
学 号	2021213619	姓 名	李小嵘
专业班级	计算机科学与技术 03 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

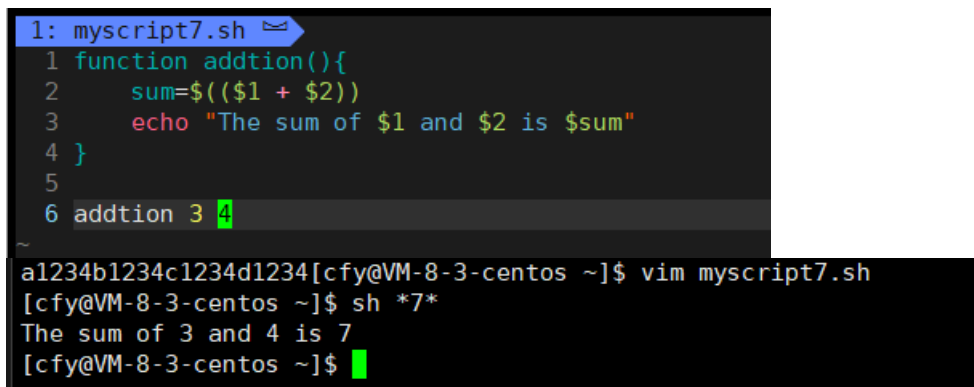
1. 掌握 Shell 函数的定义**方法**；
2. 掌握 Shell 函数的参数传递、调**用**和返回值；
3. 掌握 Shell 函数的递归调**用方法**；
4. 理解 Shell 函数的嵌套。

## 二、 实验环境

- (1) 计算机的硬件配置 PC 系列微机。
- (2) 云服务器腾讯云

## 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



```
1: myscrip7.sh ↵
1 function addition(){
2     sum=$(( $1 + $2 ))
3     echo "The sum of $1 and $2 is $sum"
4 }
5
6 addition 3 4
~
a1234b1234c1234d1234[cfy@VM-8-3-centos ~]$ vim myscrip7.sh
[cfy@VM-8-3-centos ~]$ sh *7*
The sum of 3 and 4 is 7
[cfy@VM-8-3-centos ~]$
```

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

```

1: myscript8.sh
1 function factorial(){
2     if [ $1 -eq 1 ]; then
3         # 如果n等于1, 则直接返回1
4         echo 1
5     else # 否则, 将n与n-1的阶乘相乘
6         n_minus_one=$(( $1 - 1 ))
7         n_minus_one_factorial=$(factorial $n_minus_one)
8         echo $(( $1 * $n_minus_one_factorial ))
9     fi
10 }
11
12 n=$1
13
14 echo "$n的阶乘为: $(factorial $n)"

```

```

[cfy@VM-8-3-centos ~]$ sh myscript8.sh 6
6的阶乘为: 720
[cfy@VM-8-3-centos ~]$

```

3.运行该程序，并观察程序运行结果， 函数嵌套的含义

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

```
function first() {
```

```
    function second()
```

```
        { function third()
```

```
        {
```

```
            echo "-----this is third"
```

```
        }
```

```
        echo "this is the second"
```

```
        third
```

```
    }
```

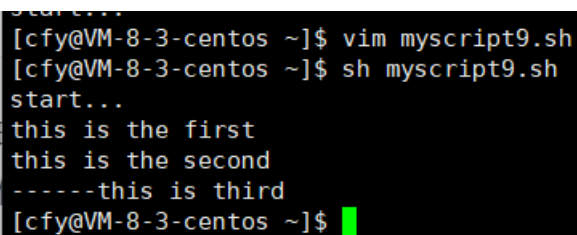
```
    echo "this is the first"
```

```
second

}

echo "start..."

first
```

A terminal window screenshot showing the execution of a script. The prompt is [cfy@VM-8-3-centos ~]\$. The user runs vim myscript9.sh, then sh myscript9.sh. The output is: start..., this is the first, this is the second, -----this is third. The prompt returns to [cfy@VM-8-3-centos ~]\$.

```
[cfy@VM-8-3-centos ~]$ vim myscript9.sh
[cfy@VM-8-3-centos ~]$ sh myscript9.sh
start...
this is the first
this is the second
-----this is third
[cfy@VM-8-3-centos ~]$
```

#### 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 4 月 26 日
学 号	2021213619	姓 名	李小嵘
专业班级	计算机科学与技术 03 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) 云服务器腾讯云

## 三、 实验内容及结果

1、已知 quote.txt 文件内容如下

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.

试编写 sed 命令实现如下功能：

- (1.) 删除\$符号；

```
> cat  
[cfy@VM-8-3-centos ~]$ cat quote.txt | set -i 's/\$/g' quote.txt  
[cfy@VM-8-3-centos ~]$ cat 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.  
[cfy@VM-8-3-centos ~]$
```

(2.) 显示包含 music 文字的行内容及行号

```
[cfy@VM-8-3-centos ~]$ cat quote.txt | sed -n '/music/p'  
It was an evening of splendid music and company.  
[cfy@VM-8-3-centos ~]$
```

(3.) 在第 4 行后面追加内容: “hello world!”;

```
[cfy@VM-8-3-centos ~]$ cat quote.txt | sed '4a hello world!'  
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!  
[cfy@VM-8-3-centos ~]$
```

(3.) 将文本 “The” 替换为 “Quod”;

```
[cfy@VM-8-3-centos ~]$ cat quote.txt | sed 's/The/Quod/g'  
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.  
[cfy@VM-8-3-centos ~]$
```

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

```
[cfy@VM-8-3-centos ~]$ cat quote.txt | sed '3c This is the third line.'  
The honeysuckle band played all night long for only 0.  
This is the third line.  
The local nurse Miss P.Neave was in attendance.  
[cfy@VM-8-3-centos ~]$
```

(5.) 删除第 2 行内容;

```
[cfy@VM-8-3-centos ~]$ cat quote.txt | sed '2d'
The honeysuckle band played all night long for only 0.
Too bad the disco floor fell through at 23:10.
The local nurse Miss P.Neave was in attendance.
[cfy@VM-8-3-centos ~]$
```

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

```
[cfy@VM-8-3-centos ~]$ var=evening
[cfy@VM-8-3-centos ~]$ cat quote.txt | sed -n "/$var/p"
It was an evening of splendid music and company.
[cfy@VM-8-3-centos ~]$
```

2. 文件 `number.txt` 的内容如下所示：

```
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 行的内容，观察两者的区别；

```
[cfy@VM-8-3-centos ~]$ cat << EOF >number.txt
> one : two : three
> four : five : six
> EOF
[cfy@VM-8-3-centos ~]$ cat number.txt | awk '{FS=":"}{print $2}'
:
five
[cfy@VM-8-3-centos ~]$ cat number.txt | awk '{FS=" "}{print $2}'
:
:
[cfy@VM-8-3-centos ~]$
```

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。

例如: `foo.txt` 内容为:

```
2 4 3
15 46 79
```

则输出为:

```
even:
2
4
46
numbers:
3
```

```
[cfy@VM-8-3-centos ~]$ awk '{for(i=1;i<=NF;i++) if($i%2==0) {print $i; count++;}} END{print "The total number of even numbers is: "count}' foo.txt
2
4
46
The total number of even numbers is: 3
[cfy@VM-8-3-centos ~]$
```

4. 脚本的内容如下所示, 试运行该脚本吧, 并理解该脚本实现的功能

```
#!/bin/bash

read -p "enter search pattern: " pattern

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

```
[cfy@VM-8-3-centos ~]$ cat scripts.sh
#!/bin/bash

read -p "enter search pattern: "
awk "/$pattern/" '{ nmatches++; print } END { print nmatches, "found." }' info.txt

[cfy@VM-8-3-centos ~]$ cat info.txt
1 2 3
123
abc
[cfy@VM-8-3-centos ~]$ sh scripts.sh
enter search pattern: abc
1 2 3
123
abc
3 found.
[cfy@VM-8-3-centos ~]$
```

`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

#### 四、 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。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]

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