

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 19 日
学 号	2021223087	姓 名	王玉杰
专业班级	计算机科学与技术 02 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

```
yujie@yujie-virtual-machine:~$ cd /etc
yujie@yujie-virtual-machine:/etc$ pwd
/etc
yujie@yujie-virtual-machine:/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。

```
yujie@yujie-virtual-machine:/etc$ ls -a /home/yujie
.          .cache          .gconf          .presage        公共的  文档
..         .config         .gnupg          .profile        模板
.bash_logout .dbus          .ICEauthority   .Xauthority     视频
.bashrc     examples.desktop .local          .xsession-errors 图片
yujie@yujie-virtual-machine:/etc$
```

- 3、使用命令创建目录/home/lyj/linux，然后删除该目录。

```
root@yujie-virtual-machine:~# mkdir linux
root@yujie-virtual-machine:~# ls
linux
root@yujie-virtual-machine:~# rmdir linux
未找到 'rmdir' 命令，您要输入的是否是：
命令 'vidir' 来自于包 'moreutils' (universe)
命令 'redir' 来自于包 'redir' (universe)
命令 'rfdir' 来自于包 'dpm' (universe)
命令 'rmdir' 来自于包 'coreutils' (main)
rmdir: 未找到命令
root@yujie-virtual-machine:~# rmdir linux
root@yujie-virtual-machine:~# ls
root@yujie-virtual-machine:~#
```

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

```
root@yujie-virtual-machine:~# cat >bac
hello linux!
^Z
[1]+  已停止                  cat > bac
root@yujie-virtual-machine:~# cat bac
hello linux!
root@yujie-virtual-machine:~#
```

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。

```
root@yujie-virtual-machine:~# mkdir wyj
root@yujie-virtual-machine:~# cp ./bac wyj
root@yujie-virtual-machine:~# cd wyj
root@yujie-virtual-machine:~/wyj# ls
bac
root@yujie-virtual-machine:~/wyj# rm -rf wyj
root@yujie-virtual-machine:~/wyj# ls
bac
root@yujie-virtual-machine:~/wyj# cd /root
root@yujie-virtual-machine:~# ls
bac  wyj
root@yujie-virtual-machine:~# rm -rf wyj
root@yujie-virtual-machine:~# ls
bac
root@yujie-virtual-machine:~#
```

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。

```
root@yujie-virtual-machine:~# cat -3 /etc/adduser.conf
cat: 无效选项 -- 3
Try 'cat --help' for more information.
root@yujie-virtual-machine:~# head -3 /etc/adduser.conf
# /etc/adduser.conf: `adduser' configuration.
# See adduser(8) and adduser.conf(5) for full documentation.

root@yujie-virtual-machine:~# taif -5 /etc/adduser.conf
未找到 'taif' 命令, 您要输入的是否是:
命令 'tail' 来自于包 'coreutils' (main)
命令 'tailf' 来自于包 'util-linux' (main)
命令 'tgif' 来自于包 'tgif' (universe)
taif: 未找到命令
root@yujie-virtual-machine:~# tail -5 /etc/adduser.conf
tail: 无法打开 '/etc/adduser.conf' 读取数据: 没有那个文件或目录
root@yujie-virtual-machine:~# tail -5 /etc/adduser.conf
# check user and group names also against this regular expression.
#NAME_REGEX="^[a-z][-a-z0-9_]*\ $"

# use extrausers by default
#USE_EXTRAUSERS=1
root@yujie-virtual-machine:~#
```

7、分屏查看文件/etc/adduser.conf的内容。


```

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-pass
wd
# package, may assume that UIDs less than 100 are unallocated.
FIRST_SYSTEM_UID=100
LAST_SYSTEM_UID=999

FIRST_SYSTEM_GID=100
LAST_SYSTEM_GID=999

# FIRST_[GU]ID to LAST_[GU]ID inclusive is the range of UIDs of dynamically
# allocated user accounts/groups.
FIRST_UID=1000
LAST_UID=29999

FIRST_GID=1000
LAST_GID=29999

# The USERGROUPS variable can be either "yes" or "no". If "yes" each
# created user will be given their own group to use as a default. If
--更多--(49%)

```

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

```

root@yujie-virtual-machine:~/wyj# cat >facebook.txt
google 110 5000
baidu 100 5000
guge 50 3000
sohu 100 4500
^Z

```

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

```

root@yujie-virtual-machine:~/wyj# sort facebook.txt
baidu 100 5000
google 110 5000
guge 50 3000
sohu 100 4500

```

(2) 按公司人数排序

```
root@yujie-virtual-machine:~/wyj# sort -t' ' -k 2 -n facebook.txt
guge 50 3000
baidu 100 5000
sohu 100 4500
google 110 5000
```

(3) 按公司人数排序，人数相同的按照员工平均工资升序排序

```
root@yujie-virtual-machine:~/wyj# sort -t' ' -k2n -k3n facebook.txt
guge 50 3000
sohu 100 4500
baidu 100 5000
google 110 5000
```

(4) 按员工工资降序排序，如工资相同，则按公司人数升序排序

```
root@yujie-virtual-machine:~/wyj# sort -t' ' -k3r -k2n facebook.txt
baidu 100 5000
google 110 5000
sohu 100 4500
guge 50 3000
root@yujie-virtual-machine:~/wyj#
```

(5) 从公司英文名称的第2个字母开始进行排序。

```
root@yujie-virtual-machine:~/wyj# sort -t' ' -k1,2 facebook.txt
baidu 100 5000
google 110 5000
guge 50 3000
sohu 100 4500
```

四、 实验过程分析与讨论

- 功能说明：将文本文件内容加以排序,sort 可针对文本文件的内容，以行为单位来排序。
- 格式：sort [选项] filename

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 19 日
学 号	2021223087	姓 名	王玉杰
专业班级	计算机科学与技术 02 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

- 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、locate 、wc
- 2、掌握 Linux 下文件打包、压缩命令：tar gzip
- 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 chown

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) 在用户主目录下新建目录 locate，在 locate 下新建文件 newfile，内容随意写几行。

```
root@yujie-virtual-machine:~# cat >newfile
hello world
i love you
from china
^Z
[2]+ 已停止                  cat > newfile
root@yujie-virtual-machine:~#
```

- (2) 在用户主目录下查找文件 newfile，并显示该文件位置信息。

```
root@yujie-virtual-machine:~# find . -name new*
./locate/newfile
root@yujie-virtual-machine:~#
```

- (4) 统计 newfile 文件中所包含的行数、字数和字节数。

```
[3]+ 已停止                  cat > newfile
root@yujie-virtual-machine:~/locate# wc newfile
 3  7 33 newfile
```

- (5) 创建文件 newfile1，在用户主目录下查找比文件 newfile 更新的文件。

```
find: newfile: 没有那个文件或目录
root@yujie-virtual-machine:~# find . -newer ./locate/newfile
./locate
./locate/newfile1
```

(4) 在用户主目录下查找文件 newfile，并删除该文件。

(6) 查看文件夹 locate 内容，看一下是否删除了文件 newfile。

```
root@yujie-virtual-machine:~# find . -name newfile -exec rm {} \;
root@yujie-virtual-machine:~# ls
bac  locate  wyj
root@yujie-virtual-machine:~# cd locate
root@yujie-virtual-machine:~/locate# ls
root@yujie-virtual-machine:~/locate#
```

2、文件打包

(1) 在用户主目录下新建文件夹 m1，在 m1 下新建文件 f1 和 f2。

```
root@yujie-virtual-machine:~# mkdir m1
root@yujie-virtual-machine:~/m1# touch f1 f2
root@yujie-virtual-machine:~/m1# ls
f1  f2
```

(2) 在用户主目录下新建文件夹 m2，在 m2 下新建文件 f3。

```
root@yujie-virtual-machine:~/m2# touch f3
root@yujie-virtual-machine:~/m2# ls
f3
```

(3) 在用户主目录下新建文件 f4。

```
root@yujie-virtual-machine:~# touch f4
```

(4) 在用户主目录下对文件夹 m1 和 f4 进行打包，生成文件 bao1.tar。

```
root@yujie-virtual-machine:~# tar -cf bao1.tar m1 f4
root@yujie-virtual-machine:~# tar -tf bao1.tar
m1/
m1/f1
m1/f2
f4
```

(5) 查看包 bao1.tar 的内容。

向包 bao1.tar 里添加文件夹 m2 的内容。

```
root@yujie-virtual-machine:~# tar -rvf bao1.tar m2
m2/
m2/f3
```

(7) 将包 bao1.tar 复制到用户主目录下的新建文件夹 m3 中。

```

root@yujie-virtual-machine:~# mkdir m3
root@yujie-virtual-machine:~# cp bao1.tar m3
root@yujie-virtual-machine:~# cd m3
root@yujie-virtual-machine:~/m3# ls
bao1.tar
root@yujie-virtual-machine:~/m3#

```

(8) 进入 m3 文件夹，并还原包 bao1.tar 的内容。

```

root@yujie-virtual-machine:~/m3# tar -xvf bao1.tar
m1/
m1/f1
m1/f2
f4
m2/
m2/f3

```

3、符号链接内容

(1) 新建文件 a.txt, 内容为 12345。

```

root@yujie-virtual-machine:~# cat >a.txt

```

Ubuntu 软件

[4]+ 已停止 cat > a.txt

(2) 建立 a.txt 得硬链接文件 b.txt，并比较 b.txt 的内容和 a.txt 是否相同，要求用 comm 或 diff 命令。

(3) 查看 a.txt 和 b.txt 的 i 节点号(inode)是否相同。

```

root@yujie-virtual-machine:~# ln a.txt b.txt
root@yujie-virtual-machine:~# ls
a.txt  bac  bao1.tar  b.txt  f4  locate  m1  m2  m3  wyj
root@yujie-virtual-machine:~# diff a.txt b.txt
root@yujie-virtual-machine:~# df a.txt b.txt
文件系统      1K-块   已用    可用  已用%  挂载点
/dev/sda1      19525500 4548268 13962348   25% /
/dev/sda1      19525500 4548268 13962348   25% /

```

(4) 修改 b.txt 的内容为 123456，然后通过命令判断 a.txt 与 b.txt 是否相同。

```

[5]+ 已停止 cat >> b.txt
root@yujie-virtual-machine:~# df a.txt b.txt
文件系统      1K-块   已用    可用  已用%  挂载点
/dev/sda1      19525500 4548268 13962348   25% /
/dev/sda1      19525500 4548268 13962348   25% /
root@yujie-virtual-machine:~#

```

(5) 删除 a.txt 文件，然后查看 b.txt 文件的 inode 及内容。


```

root@yujie-virtual-machine:~# stat b.txt
 文件: 'b.txt'
 大小: 6              块: 8              IO 块: 4096   普通文件
设备: 801h/2049d      Inode: 1046619    硬链接: 1
权限: (0644/-rw-r--r--) Uid: (   0/   root)  Gid: (   0/   root)
最近访问: 2023-05-23 14:13:06.305786577 +0800
最近更改: 2023-05-23 14:13:09.813750406 +0800
最近改动: 2023-05-23 14:18:22.743803619 +0800
创建时间: -

```

(6) 建立文件 b.txt 的符号链接文件 c.txt, 然后查看 b.txt 和 c.txt 的 inode 号, 观察两者是否相同, 比较 b.txt 和 c.txt 的文件内容是否相同。

```

root@yujie-virtual-machine:~# ln -s b.txt c.txt
root@yujie-virtual-machine:~# stat b.txt c.txt
 文件: 'b.txt'
 大小: 6              块: 8              IO 块: 4096   普通文件
设备: 801h/2049d      Inode: 1046619    硬链接: 1
权限: (0644/-rw-r--r--) Uid: (   0/   root)  Gid: (   0/   root)
最近访问: 2023-05-23 14:13:06.305786577 +0800
最近更改: 2023-05-23 14:13:09.813750406 +0800
最近改动: 2023-05-23 14:18:22.743803619 +0800
创建时间: -
 文件: 'c.txt' -> 'b.txt'
 大小: 5              块: 0              IO 块: 4096   符号链接
设备: 801h/2049d      Inode: 1046630    硬链接: 1
权限: (0777/lrwxrwxrwx) Uid: (   0/   root)  Gid: (   0/   root)
最近访问: 2023-05-23 14:19:49.167610549 +0800
最近更改: 2023-05-23 14:19:49.167610549 +0800
最近改动: 2023-05-23 14:19:49.167610549 +0800
创建时间: -
root@yujie-virtual-machine:~#

```

(7) 删除 b.txt 后查看 c.txt, 观察系统给出什么提示信息。

```

root@yujie-virtual-machine:~# rm b.txt
root@yujie-virtual-machine:~# cat c.txt
cat: c.txt: 没有那个文件或目录

```

4、权限管理

(1) 新建文件 tt.txt, 是否能创建。

```

root@yujie-virtual-machine:~# touch tt.txt
root@yujie-virtual-machine:~# ls
bac  bao1.tar  c.txt  f4  locate  m1  m2  m3  tt.txt  wyj
root@yujie-virtual-machine:~# touch tt.txt
root@yujie-virtual-machine:~# chmod a+x tt.txt
root@yujie-virtual-machine:~#

```

(2) 增加写权限, 创建文件 tt.txt 并为该文件增加执行权限 (所有用户都可以执行)。

```

root@yujie-virtual-machine:~# chmod go-x tt.txt

```

(4) 为文件 tt.txt 去除组和其它用户的执行权限。

(5) 更改文件的所有者

```

root@yujie-virtual-machine:~# chown yujie tt.txt

```

八、 实验过程分析与讨论

运行 `updatedb` 命令时，显示数据库被锁 `Updatedb updatedb` 命令用来创建或更新 `locate/slocate` 命令所必需的数据库 文件。`updatedb` 命令的执行过程较长，因为在执行时它会遍历整个 系统的目录树，并将所有的文件信息写入 `locate/slocate` 数据库文 件中。# 更新指定命令的 `slocate` 数据库 `updatedb -U /usr/local/`。

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 vi 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 19 日
学 号	2021223087	姓 名	王玉杰
专业班级	计算机科学与技术 02 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

掌握 vi 编辑器及 gcc 编译器

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、vi 编辑器和 gcc 编译器的简单使用

- (1) 在用户主目录下新建一个目录，命名为 vifile

```
root@yujie-virtual-machine:~# mkdir vifile
root@yujie-virtual-machine:~#
```

- (2) 进入目录 vifile

```
root@yujie-virtual-machine:~# cd vifile
root@yujie-virtual-machine:~/vifile#
```

- (3) 在 vifile 下用 vi 编辑器新建一个 c 语言程序文件，文件名为 test.c test.c 文件内容为： `int main() { printf("hello world!\n"); }`

```
root@yujie-virtual-machine:~/vifile# cat tes
int main()
{
printf("hello world\n");
}
```

- (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 test，查看执行结果。

```
root@yujie-virtual-machine:~/vifile# ls
a.out test test.c
root@yujie-virtual-machine:~/vifile# ./a.out
hello world
```

2、vi 编辑器的详细使用

(1) 在用户主目录下建一个名为 vi 的目录。

```
root@yujie-virtual-machine:~# cd vifile
root@yujie-virtual-machine:~/vifile#
```

(2) 进入 vi 目录。

(3) 将文件/etc/gai.conf 复制到当前目录下，并用命令 sudo 修改 gai.conf 的属性为所有用户可 以读写。

```
root@yujie-virtual-machine:~/vi# cp /etc/gai.conf ./
root@yujie-virtual-machine:~/vi# ls
gai.conf
root@yujie-virtual-machine:~/vi# sudo chmod a=wr gai.conf
root@yujie-virtual-machine:~/vi# ls
gai.conf
root@yujie-virtual-machine:~/vi# ll
总用量 12
drwxr-xr-x  2 root root 4096 5月 23 16:34 ./
drwx----- 10 root root 4096 5月 23 16:32 ../
-rw-rw-rw-  1 root root 2584 5月 23 16:34 gai.conf
```

(4) 使用 vi 编辑当前目录下的 gai.conf。

```
# 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 in
#
# 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
:
```

(5) 显示行号。

```
1 # Configuration for getaddrinfo(3).
2 #
3 # So far only configuration for the destination address sorting is needed.
4 # RFC 3484 governs the sorting. But the RFC also says that system
5 # administrators should be able to overwrite the defaults. This can be
6 # achieved here.
7 #
8 # All lines have an initial identifier specifying the option followed by
9 # up to two values. Information specified in this file replaces the
10 # default information. Complete absence of data of one kind causes the
11 # appropriate default information to be used. The supported commands include:
12 #
13 # reload <yes|no>
14 #     If set to yes, each getaddrinfo(3) call will check whether this file
15 #     has changed and if necessary reload. This option should not really be
16 #     used. There are possible runtime problems. The default is no.
17 #
18 # label <mask> <value>
19 #     Add another rule to the RFC 3484 label table. See section 2.1 in
20 #     RFC 3484. The default is:
21
22 :set nu
```

(6) 将光标移到第 18 行。 gg+17+↓

```
16 #     used. There are possible runtime problems. The default is no.
17 #
18 # label <mask> <value>
19 #     Add another rule to the RFC 3484 label table. See section 2.
20 #     RFC 3484. The default is:
E492: Not an editor command: gg+17
```


(7) 复制该行内容。 Yy

(8) 将光标移到最后一行行首。 G

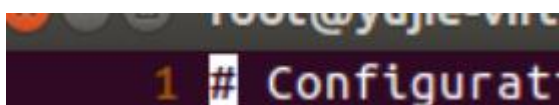
(9) 粘贴复制行的内容。 p

```
62 #
63 #scopev4 ::ffff:169.254.0.0/112 2
64 #scopev4 ::ffff:127.0.0.0/104 2
65 #scopev4 ::ffff:0.0.0.0/96 14
66 #     Add another rule to the RFC 3484 label table. See section 2.1 in
```

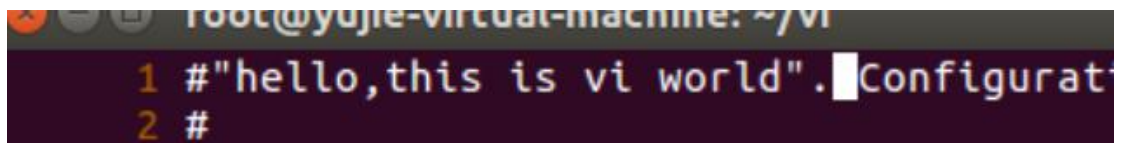
(9) 存盘但不退出。 :w



(10) 将光标移到首行。 gg

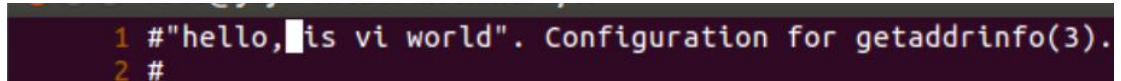


(11) 插入模式下输入“Hello, this is vi world!”。



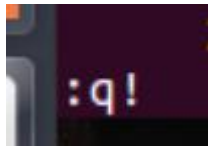
```
root@yujie-virtual-machine: ~/vi
1 #\"hello,this is vi world\". Configuration for getaddrinfo(3).
2 #
```

(12) 删除字符串“this”。 光标指向 this 首字母+dw



```
1 #\"hello, is vi world\". Configuration for getaddrinfo(3).
2 #
```

(13) 强制退出vi，不存盘



十二、 实验过程分析与讨论

各种选项有点记不住，需要多次联系及时查阅。

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 5 月 19 日
学 号	2021223087	姓 名	王玉杰
专业班级	计算机科学与技术 02 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`
- 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、实验内容及结果

- 1、建立一个用户名为 jone，描述信息为 jone，登录 shell 为 /bin/sh，登录主目录为 /home/jone 的用户，并设置口令为 123456。

```
root@yujie-virtual-machine:~/vi# useradd jpne -s /bin/sh -b /home/jone -c jone  
p 123456  
root@yujie-virtual-machine:~/vi#
```

- 2、使用命令从用户 root 切换到用户 jone，修改 jone 的 UID 为 2000，其 shell 类型为 /bin/csh。

```
root@yujie-virtual-machine:~/vi# usermod jpne -u 2000 -s /bin/csh
```

- 3、使用命令从用户 jone 切换到 root。

```
root@yujie-virtual-machine:~/vi# home/yujie#
```

- 4、使用命令删除 jone 用户，并且在删除该用户的同时一起删除其主目录。

```
root@yujie-virtual-machine:~/vi# userdel jpne -r  
userdel: jpne 邮件池 (/var/mail/jpne) 未找到  
userdel: 未找到 jpne 的主目录 "/home/jone/jpne"  
root@yujie-virtual-machine:~/vi# userdel jone -r  
userdel: 用户“jone”不存在  
root@yujie-virtual-machine:~/vi#
```

- 5、使用命令 newusers 批量创建用户，并使用命令 chpasswd 为这个批量用户创建密码（密码也是批量创建的），查看 /etc/passwd 文件确认是否创建成功。

```
user1:x:2000:2000:user1,,,/home/user1:bin/bash  
user2:x:2000:2000:user2,,,/home/user1:bin/bash
```

```
root@zzc-virtual-machine:/home/zzc# newusers <user  
newusers: 第 3 行: 无效行  
newusers: 发现错误, 忽略改动  
root@zzc-virtual-machine:/home/zzc# newusers <user  
root@zzc-virtual-machine:/home/zzc# cat userpasswd  
root@zzc-virtual-machine:/home/zzc# su user1  
user1@zzc-virtual-machine:/home/zzc$ su user2  
密码:  
user2@zzc-virtual-machine:/home/zzc$
```

- 6、使用命令创建用户组 group1，并在创建时设置其 GID 为 3000。

```
root@yujie-virtual-machine:~/vi# groupadd group1 -g 3000
root@yujie-virtual-machine:~/vi#
```

- 7、在用户组 group1 中添加两个之前批量创建的用户。

```
root@yujie-virtual-machine:~/vi# usermod -g group1 user1
```

- 8、切换到 group1 组中的某个用户，在该用户下使用 sudo 命令查看/etc/shadow 文件，看一下是否可以执行。若不能执行，修改 sudoers 文件使得该用户可以查看/etc/shadow 文件内容（尝试两种方法）

```
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults                env_reset
Defaults                mail_badpass
Defaults                secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/usr/sbin:/bin:/snap/bin"

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges
"/etc/sudoers" [readonly] 30 lines, 755 characters
```

```

root!!:19498:0:99999:7:::
daemon*:18480:0:99999:7:::
bin*:18480:0:99999:7:::
sys*:18480:0:99999:7:::
sync*:18480:0:99999:7:::
games*:18480:0:99999:7:::
man*:18480:0:99999:7:::
lp*:18480:0:99999:7:::
mail*:18480:0:99999:7:::
news*:18480:0:99999:7:::
uucp*:18480:0:99999:7:::
proxy*:18480:0:99999:7:::
www-data*:18480:0:99999:7:::
backup*:18480:0:99999:7:::
list*:18480:0:99999:7:::
irc*:18480:0:99999:7:::
gnats*:18480:0:99999:7:::
nobody*:18480:0:99999:7:::
systemd-timesync*:18480:0:99999:7:::
systemd-network*:18480:0:99999:7:::
systemd-resolve*:18480:0:99999:7:::
systemd-bus-proxy*:18480:0:99999:7:::
syslog*:18480:0:99999:7:::
"/etc/shadow" 40 lines, 1246 characters

```

十六、 实验过程分析与讨论

创建组群 china [root@localhost ~]# groupadd china 创建组群 ou, 并且设置该组群 GID 为 800 [root@localhost ~]# groupadd -g 800 ou 创建系统组群 chinese [root@localhost ~]# groupadd -r

主要概念:

- 基本上, 一个组就是一个整数组 ID (gid)
- 每个在系统上运行的进程都是属于一个组的集合 (gids)
- /etc/group 文件把组 ID 映射到组名称和组成员身上 /etc/group 文件存储格式 (组名称: 组密码: 组 ID: 组成员)

root:x:0:root lzgonline:x:500: 字段解释:

- 组名称: 每个组都有一个组名称
- 组密码: 可以给组提供一个密码, 一般很少这么做
- 组 ID: 像用户 ID 一样, linux 内核使用 ID 来识别
- 组成员: 定义组成员用户名列表, 用半角逗号隔开

4、文件系统中的每个文件有唯一的组 ID, 就像拥有唯一的所有者 ID 一样

drwxrwxr-x. 2 lzgonline lzgonline 4096 6 月 23 23:47 coding drwxr-xr-x. 2 lzgonline lzgonline 4096 6 月 23 22:03 公共的

5、用户有一个在 /etc/passwd 文件中定义的主要组 (第 4 个字段定义) root:x:0:0:root:/root:/bin/bash

6、用户可以在 /etc/group 文件中定义多个次要组 (例从下面可以看到 root 用户属于多个组)

root:x:0:root bin:x:1:root,bin,daemon daemon:x:2:root,bin,daemon sys:x:3:root,bin,adm adm:x:4:root,adm,daemon disk:x:6:root wheel:x:10:root

7、在 redhat 企业版中, 用户的主要组几乎总是与用户名相同

/etc/passwd 文件: lzgonline:x:500:500:liuzhigong:/home/lzgonline:/bin/bash

/etc/group 文件: lzgonline:x:500:

8、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

指导教师签字：卢洋

实验报告

实验名称	实验五 Shell 程序的创建一级条件判断语句		
实验教室	丹青 922	实验日期	2023 年 5 月 19 日
学 号	2021223087	姓 名	王玉杰
专业班级	计算机科学与技术 02 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 AK 的值为 200，并将其显示在屏幕上。（终端上执行）

```
root@yujie-virtual-machine:~/vi# ak=200
root@yujie-virtual-machine:~/vi# echo $ak
200
root@yujie-virtual-machine:~/vi#
```

- 2、定义变量 AM 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）

```
root@yujie-virtual-machine:~/vi# am=100
root@yujie-virtual-machine:~/vi# test $am -gt 150
root@yujie-virtual-machine:~/vi# test $am -gt 150 && echo yes
root@yujie-virtual-machine:~/vi#
```

- 3、创建一个简单的 Shell 程序，其功能为显示计算机主机名（hostname）和系统时间（date）
- 4、创建一个简单的 Shell 程序，要求带一个参数，判断该参数是否是水仙花数。所谓水仙花数是指一个 3 位数，它的每个位上的数字的 3 次幂之和等于它本身。例如 $153=1^3+3^3+5^3$ ，153 是水仙花数。编写程序时要求首先进行参数个数判断，判断是否带了一个参数，如果没有参数则给出提示信息，否则给出该数是否是水仙花数。要求对 153，124，370 分别进行测试判断。

```
echo "Total parameter are: $#"
```

```
test $# -eq 0 && echo "You don't give one parameter at least" && exit 0
```

```
for var in $@
do
    echo "The num is: $var"
    var0=$var
    var1=$((var0/100))
    var0=$((var0%100))
    var2=$((var0/10))
    var3=$((var0%10))
    if [ $((var1*var1*var1+var2*var2*var2+var3*var3*var3)) -eq $var ];then
        echo "$var is shuixianhua num!"
    else
        echo "$var is not a shuixianhua num."
    fi
done
```

- 4、创建一个简单的 Shell 程序，实现输入目录名，查看当前文件夹下有没有这个目录。如果没有则创建该目录，若已存在则输出“exist”。


```
read -p "Please input a file name: " file
if [ -e $file ];then
    echo "exist"
    exit 0
else
    echo "$file is not exist,now mkdir $file"
    eval "mkdir $file"
fi
```

6、创建一个简单的 shell 程序，输入学生的成绩，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

```
echo "The parameter num are: $#"
```

```
for var in $@
do
    if [ $var -ge 90 ];then
        echo "A"
    elif [ "$var" -ge 80 -a "$var" -lt 90 ];then
        echo "B"
    elif [ "$var" -ge 70 -a "$var" -lt 80 ];then
        echo "C"
    elif [ "$var" -ge 60 -a "$var" -lt 70 ];then
        echo "D"
    else
        echo "E"
    fi
done
```

二十、 实验过程分析与讨论

$=$ or $=$: 等于 \neq : 不等于 \neq : 不等于 $>$: 大于 \geq : 大于等于 $<$: 小于 \leq : 小于等于这几个有点记不住

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 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 19 日
学 号	2021223087	姓 名	王玉杰
专业班级	计算机科学与技术 02 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) 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) 编写一个 shell 脚本，利用 for 循环把当前目录下的所有 *.sh 文件复制到指定的目录中，并为没有执行权限的文件添加执行权限。（可以在当前目录下先建立几个 *.sh 文件，用来测试，复制到的指定目录可以自己建立一个）

```
#!/bin/bash
read -p "please writd own name" file
filelist=$(ls $file/*.sh)
echo "$filelist"
for filename in $filelist
do
    echo "$filename"
    if [ ! -x "$filename" ];then
        eval "chmod a+x $filename"
    eval "cp $filename movedir"
    fi
done
```

~~~~~

- (2) 编写 shell 脚本，利用 while 循环求前 10 个偶数之和。

```
i=0
sum=0
while [ "$i" -lt 20 ]
do
    if [ $((($i%2)) -eq 0 ] ;then
        sum=$((sum+$i))
    fi
    i=$((i+1))
done
echo "$sum"
```

- (3) 编写 shell 脚本，利用 until 循环求 1 到 10 的平方和。

```
i=1
sum=0
until [ "$i" -gt 10 ]
do
    sum=$((sum+$i*$i))
    i=$((i+1))
done
echo "$sum"
```

- (4) 运行下列程序，观察程序的运行结果。红色的语句分别为 break，break 2，continue，continue2，观察四种情况下的实验结果。#!/bin/sh 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 或 continue fi echo -n " \$j" done echo \$j done 1.当为 break 时，外层for循环只有在内层for循环中的j==5时跳出当前循环，j只输出5 2.当为break 2 时，内层for循环中j==5时，直接跳出内外两层循环，因此只输出外层循环的第一个值a 3.当为continue 时，内层for循环中的j==5时，直接跳过当前循环体内剩余的语句，直接进行下一次循环，因为只有当整个内层循环全部执行完之后才输出j，因为最后一次循环j=10，故输出j=10 4.当为continue 2 时，内层for循环中j==5时，直接跳过两层循环体剩余的语句，因此不会执行第一层for循环内的 echo \$j，而只会执行echo

种情况下的实验结果。

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

种情况下的实验结果。

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

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



## 二十四、 实验过程分析与讨论

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 `#!/bin/bash for variable1 in {1..5} #for variable1 in 1 2 3 4 5 do echo "Hello, Welcome $variable1 times " done 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。Sheel 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）。Sheel 中还支持按规定的步数进行跳跃的方式实现列表 for 循环，例如计算 1~100 内所有的奇数之和

## 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 19 日 |
| 学 号  | 2021223087    | 姓 名  | 王玉杰             |
| 专业班级 | 计算机科学与技术 02 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

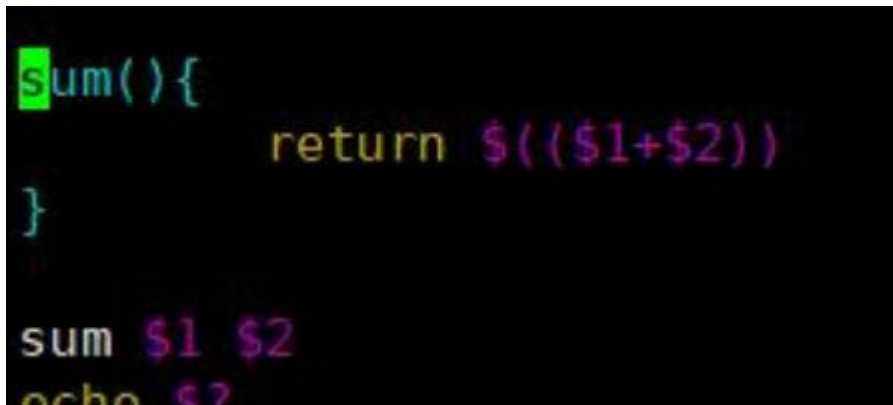
- 1、掌握 Shell 函数的定义方法
- 2、掌握 shell 函数的参数传递、调用和返回值
- 3、掌握 shell 函数的递归调用方法
- 4、理解 shell 函数的嵌套。

## 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) 编写 shell 脚本，定义一个函数返回两个数的和。
- (2) 编写 shell 脚本，该脚本中定义一个递归函数，求 n 的阶乘。



```
sum(){  
    return $((($1+$2))  
}  
  
sum $1 $2  
echo $?
```

```
#!/bin/bash
#function 阶乘

jiecheng(){
    local i=1
    local mul=1
    while [ $i -le $n ]
    do
        mul=$((i*$mul))
        i=$((i+1))
    done
    return $mul
}

n=$1

jiecheng $n
echo "$?"
```

(3)已知 shell 脚本 test.sh 内容如下所示，试运行下列程序，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 "-----this is third"
        }
        echo "this is the second"
    }
    echo "this is the first"
}
```

```
start...
this is the first
this is the second
-----this is third
kpl@hadoop100:~/linuxlearn$
```

## 二十八、 实验过程分析与讨论

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 19 日 |
| 学 号  | 2021223087    | 姓 名  | 王玉杰             |
| 专业班级 | 计算机科学与技术 02 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

### 二十九、 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

### 三十、 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。

### 三十一、 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

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

(2) 显示包含 music 文字的的行内容及行号

```
zxc@zxc-virtual-machine:~$ cat quote.txt | sed -n '/music/p'
It was an evening of splendid music and company.
zxc@zxc-virtual-machine:~$
```

(3) 在第 4 行后面追加文件“hello world! ”

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

(5) 将文本“The”修改为“Ok”

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

(6) 将第 3 行内容修改为“This is the third line.”

```

zxc@zxc-virtual-machine:~$ cat quote.txt | sed '2c This is the third line'
The honeysuckle band played all night long for only $90.
This is the third line
Too bad the disco floor fell through at 23:10.
The local nurse Miss P.Neave was in attendance.
zxc@zxc-virtual-machine:~$ cat quote.txt
The honeysuckle band played all night long for only $90.
It was an evening of splendid music and company.
Too bad the disco floor fell through at 23:10.
The local nurse Miss P.Neave was in attendance.

```

(7) 删除第 2 行内容。

```

zxc@zxc-virtual-machine:~$ cat quote.txt | sed '2d'
The honeysuckle band played all night long for only $90.
Too bad the disco floor fell through at 23:10.
The local nurse Miss P.Neave was in attendance.
zxc@zxc-virtual-machine:~$ cat quote.txt
The honeysuckle band played all night long for only $90.
It was an evening of splendid music and company.
Too bad the disco floor fell through at 23:10.
The local nurse Miss P.Neave was in attendance.
zxc@zxc-virtual-machine:~$

```

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

```

zxc@zxc-virtual-machine:~$ cat quote.txt | sed -n "/$var/p"
It was an evening of splendid music and company.

```

2、已知文件 aaa.txt 内容如下“one : two : three four : five : six (注：每个冒号前后都有空格) 试编写 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 如果以一个空格作为分隔符，则冒号会被视为单独的一列 如果以一个冒号作为分隔符，则会将字段分为 5 组，且第一组的冒号：会被保留，且对角线上的元素会被分为一列

```

zxc@zxc-virtual-machine:~$ cat aaa.txt | awk '{FS=":"}{print $2}'
:
five
zxc@zxc-virtual-machine:~$ cat aaa.txt | awk '{FS=" "}{print $2}'
:
:
:
zxc@zxc-virtual-machine:~$

```

3、已知文件 b.txt 里面都是数字，且每行包含 3 个数字，数字之前以空格作为分隔符，试将 b.txt 里的所有偶数输出，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。例如：b.txt 内容为： 2 4 3 15 46 79 则输出为： 2 4 46

```

c%2==0){printf "%t";sum+=1}}END{printfsum}
zxc@zxc-virtual-machine:~$ cat b.txt | awk 'BEGIN{sum=0}{for(i=1;i<=NF;i++){if($i%2==0){printf "%t";sum+=1}}END{printfsum}'
2446zxc@zxc-virtual-machine:~$

```

4、已知脚本 t.sh 的内容如下，试通过运行该脚本，理解该脚本实现的功能。

```

#!/bin/bash
read -p "enter search pattern: " pattern
awk "/$pattern/" '{ nmatches++; print } END { print nmatches "found." }' info.txt
awk 中"/$pattern/"这一部分用双引号括起来，是为了允许引号内的 Shell 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

```
zxc@zxc-virtual-machine:~$ cat info.txt
hux - Sysadmin
Database - Oracle,MySQL etc.
Security - Firewall,Network, Online Security etc.
Cool - Websites
zxc@zxc-virtual-machine:~$ touch t.sh
zxc@zxc-virtual-machine:~$ cat t.sh
#!/bin/bash
read -p "enter search pattern: " pattern
awk "/$pattern/" '{ nmatches++; print } END { print nmatches "found." }' info.t
xt
```

## 三十二、 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