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	2021213162	姓　名	李忠豪
专业班级	计算机科学与技术 04 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

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

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。

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

```
lzh@lzh:~$ cd /etc
lzh@lzh:/etc$ pwd
/etc
lzh@lzh:/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。

```
lzh@lzh:/etc$ cd /home/lzh
lzh@lzh:~$ ls -a
.   bak      .bash_history  .bashrc  foo      .profile  .sudo_as_admin_successfu
..  bar.txt  .bash_logout   .cache   foo.bak  .ssh
lzh@lzh:~$ pwd
```

3、使用命令创建目录/home/lyj/linux，然后删除该目录。

```
lzh@lzh:~$ ls
bak  bar.txt  foo  foo.bak
lzh@lzh:~$ mkdir linux
lzh@lzh:~$ ls
bak  bar.txt  foo  foo.bak  li
lzh@lzh:~$ rmdir linux
lzh@lzh:~$ ls
bak  bar.txt  foo  foo.bak
lzh@lzh:~$
```

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

```
lzh@lzh:~$ cat > abc
Hello Linux!
lzh@lzh:~$ ls
abc  bak  bar.txt  foo  foo.bak
lzh@lzh:~$ cat abc
Hello Linux!
lzh@lzh:~$
```

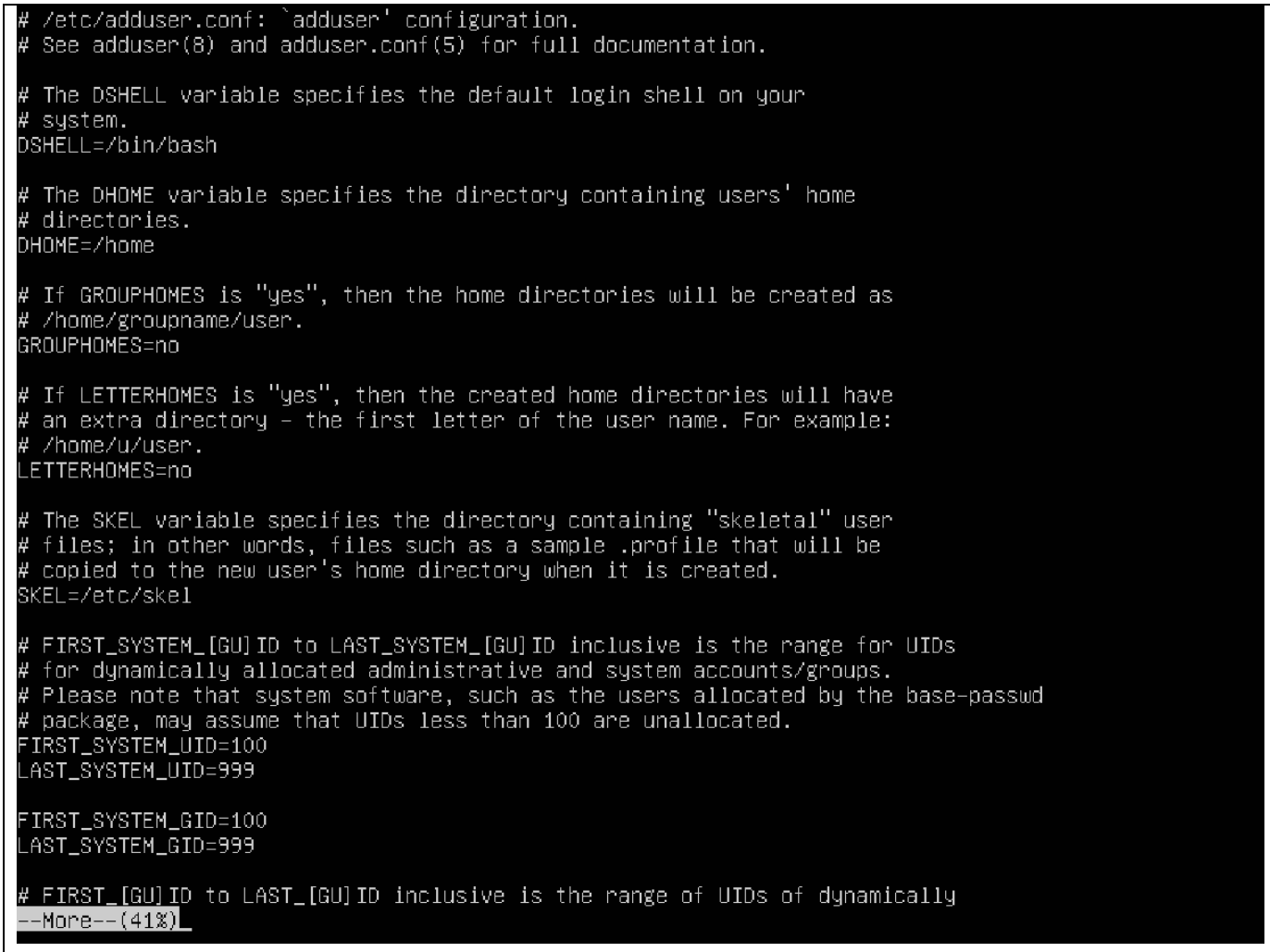
5、使用命令创建目录/home/lyj/ak，然后将/home/lyj/abc 文件复制到该目录下，最后将该目录及其目录下的文件一起删除。

```
lzh@lzh:~$ mkdir foo.bak
lzh@lzh:~$ cp abc foo.bak
lzh@lzh:~$ ls
abc  bak  bar.txt  foo.bak
lzh@lzh:~$ rm abc
lzh@lzh:~$ rm foo.bak
rm: cannot remove 'foo.bak': Is a directory
lzh@lzh:~$ rm -fm foo.bak
rm: invalid option -- 'm'
Try 'rm --help' for more information.
lzh@lzh:~$ rm -rf foo.bak
lzh@lzh:~$ ls
bak  bar.txt
lzh@lzh:~$ _
```

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。

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

lzh@lzh:/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
lzh@lzh:/etc$
```

7、分屏查看文件/etc/adduser.conf 的内容。



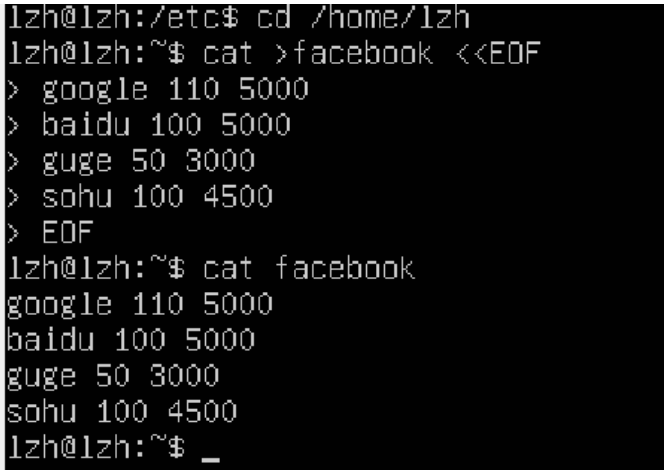
8、使用命令 cat 用输出重定向在/home/lyj 目录下创建文件 facebook.txt，文件内容为:

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

（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。

```
lzh@lzh:~$ sort facebook
baidu 100 5000
google 110 5000
guge 50 3000
sohu 100 4500
```

```
lzh@lzh:~$ sort facebook -k 2 -n
guge 50 3000
baidu 100 5000
sohu 100 4500
google 110 5000
lzh@lzh:~$ sort facebook -k 2 -k 3 -n
guge 50 3000
sohu 100 4500
baidu 100 5000
google 110 5000
lzh@lzh:~$
```

```
lzh@lzh:~$ sort facebook -k 3r -k 2 -n
baidu 100 5000
google 110 5000
sohu 100 4500
guge 50 3000
lzh@lzh:~$ sort facebook -k 1.2
baidu 100 5000
sohu 100 4500
google 110 5000
guge 50 3000
lzh@lzh:~$ _
```

实验二 Linux 常用命令（二）

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。

实验内容

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;

```
lzh@lzh:~$ mkdir baz
lzh@lzh:~$ cd baz
lzh@lzh:~/baz$ touch qux
lzh@lzh:~/baz$ echo hello world >qux
lzh@lzh:~/baz$ cat qux
hello world
lzh@lzh:~/baz$
```

（2）在用户目录下查找文件 qux，并显示该文件位置信息；

```
lzh@lzh:~$ find -name qux
./baz/qux
./qux
lzh@lzh:~$
```

（3）统计文件 qux 中所包含内容的行数、字数和字节数；

```
lzh@lzh:~$ wc qux
 1  2 12 qux
lzh@lzh:~$ _
```

（4）在用户目录下查找文件 qux，并删除该文件；

（5）查看文件夹 baz 内容，看一下是否删除了文件 qux。

```
lzh@lzh:~$ find -name qux -delete
lzh@lzh:~$ cd baz
lzh@lzh:~/baz$ ls
lzh@lzh:~/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；

```
lzh@lzh:~/baz$ cd ~
lzh@lzh:~$ pwd
/home/lzh
lzh@lzh:~$ mkdir path1
lzh@lzh:~$ cd path1
lzh@lzh:~/path1$ touch file1 file2
lzh@lzh:~/path1$ ls
file1  file2
lzh@lzh:~/path1$
```

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

```
lzh@lzh:~/path1$ cd ~
lzh@lzh:~$ mkdir path2
lzh@lzh:~$ cd path2
lzh@lzh:~/path2$ touch file3
lzh@lzh:~/path2$ ls
file3
lzh@lzh:~/path2$ _
```

（3）在用户目录下新建文件 file4；

```
lzh@lzh:~/path2$ cd ~
lzh@lzh:~$ mkdir file4
lzh@lzh:~$ ls
app     a.txt    baz     c.txt     file4        makefile  other1.o  other2.o  test
app.c   bak      bc      faceboof  libother1.a  other1.c  other2.c  path1     test.c
app.o   bar.txt  b.txt   facebook  libother2.a  other1.h  other2.h  path2     test.o
```

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

```
lzh@lzh:~$ tar -cvf package.tar path1 file4
path1/
path1/file2
path1/file1
file4/
lzh@lzh:~$
```

（5）查看包 package.tar 的内容；

```
lzh@lzh:~$ tar -tvf package.tar
drwxrwxr-x lzh/lzh           0 2023-03-22 09:19 path1/
-rw-rw-r-- lzh/lzh           0 2023-03-22 09:19 path1/file2
-rw-rw-r-- lzh/lzh           0 2023-03-22 09:19 path1/file1
drwxrwxr-x lzh/lzh           0 2023-03-22 09:53 file4/
lzh@lzh:~$ _
```

（6）向包 package.tar里添加文件夹 path2 的内容；

```
lzh@lzh:~$ tar -rvf package.tar path2
path2/
path2/file3
lzh@lzh:~$ _
```

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

```
lzh@lzh:~$ mkdir path3
lzh@lzh:~$ cp package.tar path3
lzh@lzh:~$ cd path3
lzh@lzh:~/path3$ ls
package.tar
lzh@lzh:~/path3$
```

（8）进入path3文件夹，并还原包 package.tar 的内容。

```
lzh@lzh:~/path3$ ls
package.tar
lzh@lzh:~/path3$ tar -xvf package.tar
path1/
path1/file2
path1/file1
file4/
path2/
path2/file3
lzh@lzh:~/path3$ _
```

3. 符号链接内容

（1）新建文件 foo.txt，内容为 123；

```
lzh@lzh:~$ echo '123' >foo.txt
lzh@lzh:~$ cat foo.txt
123
lzh@lzh:~$
```

（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；

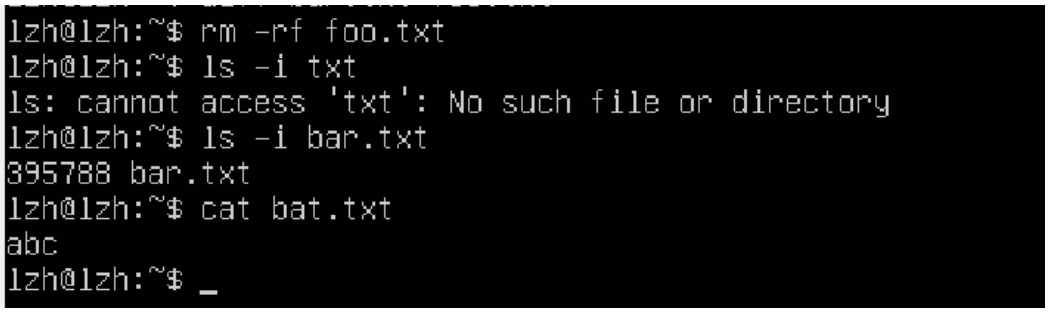
```
lzh@lzh:~$ ln foo.txt bar.txt
lzh@lzh:~$ diff foo.txt bar.txt
lzh@lzh:~$
```

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

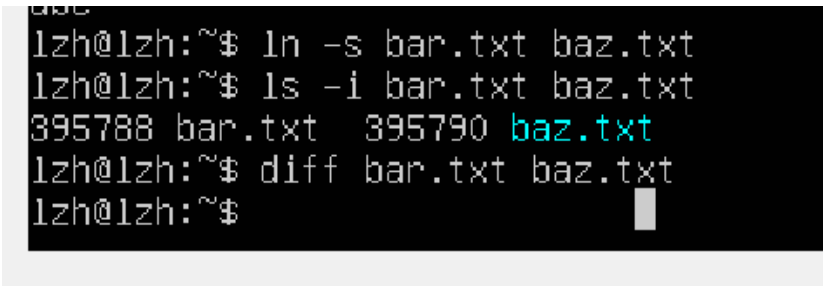
```
lzh@lzh:~$ ls -i foo.txt bar.txt
395788 bar.txt  395788 foo.txt
lzh@lzh:~$ _
```

（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；

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



（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；



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

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）。

```
lzh@lzh:~$ chmod o+x qux.txt
lzh@lzh:~$ ls -l qux.txt
-rw-rw-r-x 1 lzh lzh 1 Mar 22 11:52 qux.txt
lzh@lzh:~$ _
```

实验三 vim 编辑器及 gcc 编译器的使用

实验目的

掌握 vim 编辑器及 gcc 编译器的使用方法。

实验内容

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 ；

进入目录 workspace1 ；

```
lzh@lzh:~$ cd ~
lzh@lzh:~$ mkdir workspace1
lzh@lzh:~$ cd workspace1
lzh@lzh:~/workspace1$
```

2，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 e 为：

```c
#include <stdio.h>

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

保存 test.c 的内容，并退出；

```
~
"test.c" 6L, 73C written
lzh@lzh:~/workspace1$ cat test.c
#include <stdio.h>
int main()
{
        printf("hello world!\n");
        return  0;
}
lzh@lzh:~/workspace1$
```

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

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

4,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；

(2)进入workspace2 目录；

```
lzh@lzh:~/workspace1$ cd ~
lzh@lzh:~$ mkdir workspace2
```

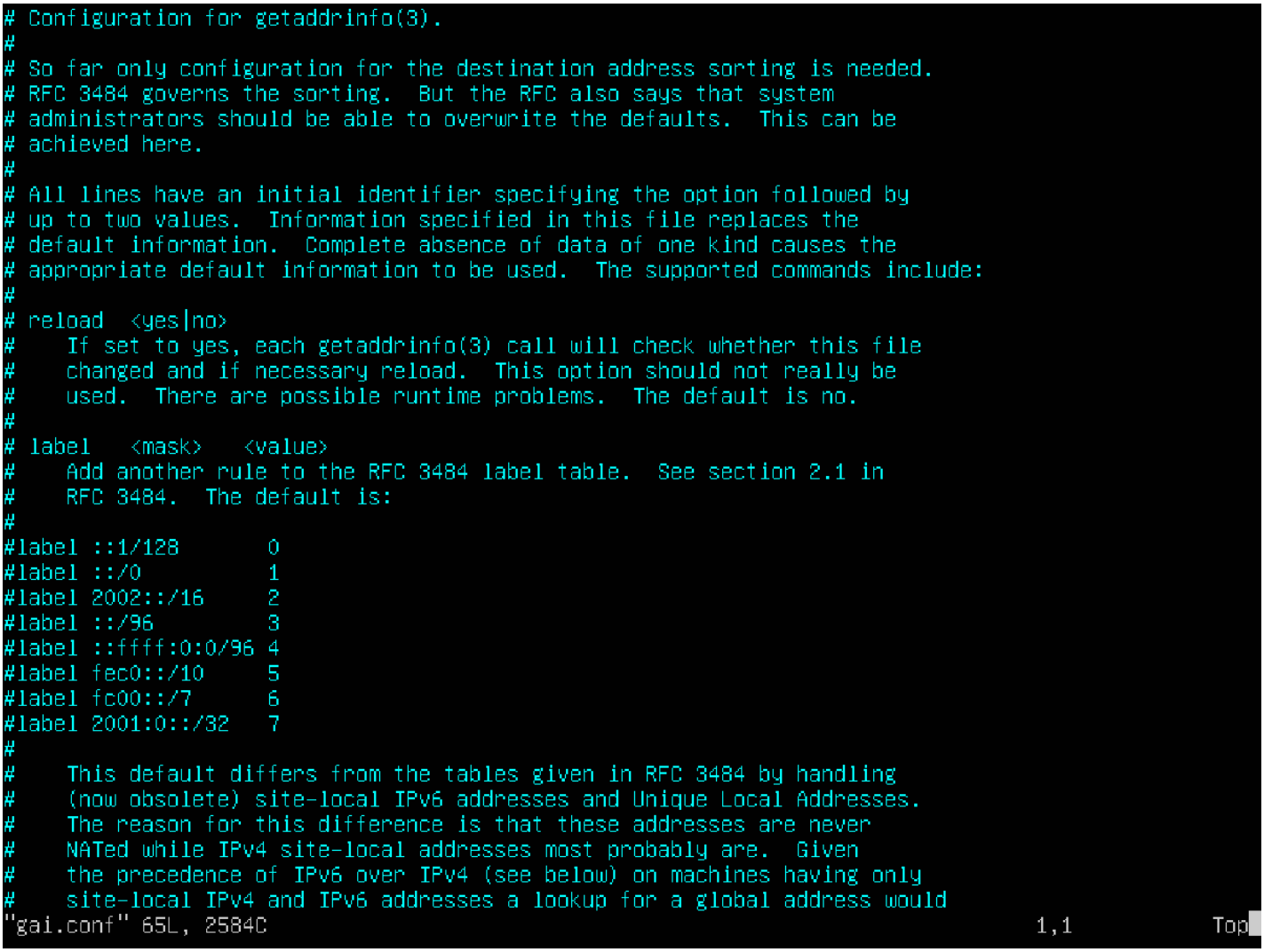
(3)使用以下命令：

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

(4)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；

```
lzh@lzh:~/workspace2$ cat /etc/gai.conf > ./gai.conf
lzh@lzh:~/workspace2$ ls
gai.conf
lzh@lzh:~/workspace2$
```

(5)使用vim 编辑当前目录下的 gai.conf ；

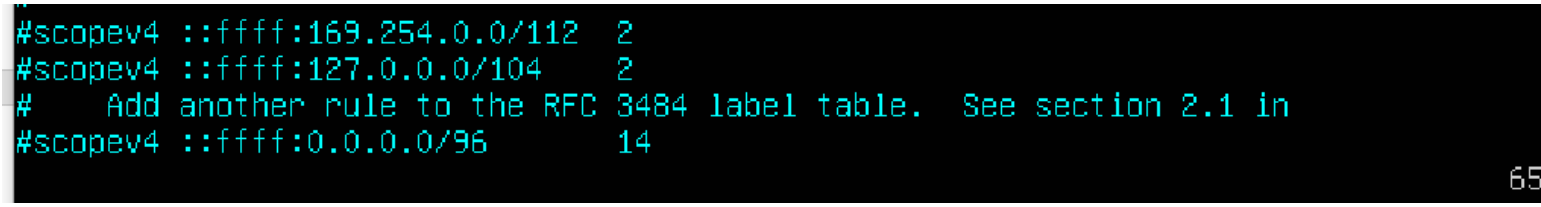


(6)将光标移到第 18 行；

(7)复制该行内容；

(8)将光标移到最后一行行首；

(8)粘贴复制行的内容；



（9）撤销第 8 步的动作;

```
#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  05:33:37
```

（10）存盘但不退出；

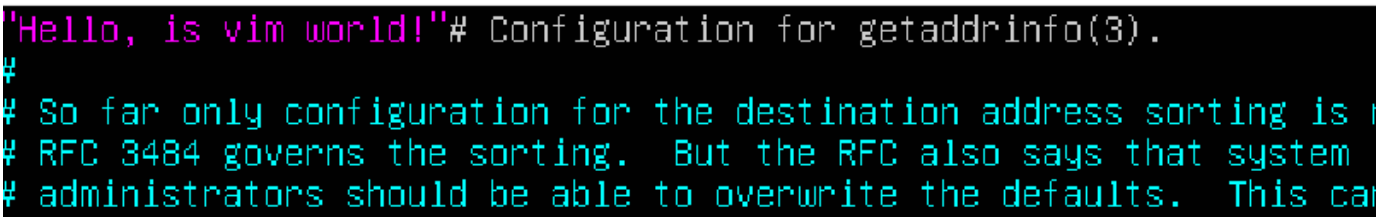
```
#    This default differs from the tables given in RFC 3484 by handling
#    (now obsolete) site-local IPv6 addresses and Unique Local Addresses.
#    The reason for this difference is that these addresses are never
#    NATed while IPv4 site-local addresses most probably are.  Given
#    the precedence of IPv6 over IPv4 (see below) on machines having only
#    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, 2584C written                                 65,1
```

（11）将光标移到首行；

（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
```

（13）删除字符串"this" ；



（14）强制退出 vim ， 不存盘。

实验四 用户和用户组管理

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

实验内容

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

```
root@lzh:/home/lzh# useradd -c "bar" -m -d /home/foo -s /bin/sh foo
```

```
root@lzh:~# passwd foo
New password:
Retype new password:
passwd: password updated successfully
root@lzh:~#
```

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@lzh:~# usermod -u 2000 -s /bin/csh foo
root@lzh:~# su foo
```

3 ，从用户 foo 切换到 root；

```
foo@lzh:~$ su root
Password:
root@lzh:/home/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@lzh:~# userdel -r foo
userdel: foo mail spool (/var/mail/foo) not found
root@lzh:~# cd /var/mail
root@lzh:/var/mail# ls -l
total 0
```

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@lzh:~/workpl1# vim newusers.txt
```

```
user1:x:1001:1001:User One:/home/user1:/bin/bash
user2:x:1002:1002:User Two:/home/user2:/bin/bash
user3:x:1003:1003:User Three:/home/user3:/bin/bash
```

```
root@lzh:~/workpl1# newusers < newusers.txt
root@lzh:~/workpl1# _
```

```
root@lzh:~/workpl1# vim passwords.txt_
```

```
root@lzh:~/workpl1# chpasswd <passwords.txt
root@lzh:~/workpl1#
```

```
user1:x:1001:1001:User One:/home/user1:/bin/bash
user2:x:1002:1002:User Two:/home/user2:/bin/bash
user3:x:1003:1003:User Three:/home/user3:/bin/bash
root@lzh:~/workpl1#
```

6，创建用户组 group1，并在创建时设置其 GID 为 3000；

```
root@lzh:~/workpl1# groupadd -g 3000 group1
root@lzh:~/workpl1# _
```

7，在用户组 group1 中添加两个之前批量创建的用户；

```
root@lzh:~/workpl1# sudo usermod -a -G group1 user1
root@lzh:~/workpl1# sudo usermod -a -G group1 user2
root@lzh:~/workpl1# sudo usermod -a -G group1 user2
root@lzh:~/workpl1# sudo usermod -a -G group1 user3
root@lzh:~/workpl1#
```

8，切换到 group1 组中的任一用户，在该用户下使用sudo 命令查看/etc/shadow文件，

检查上述操作是否可以执行；若不能执行，修改 sudoers文件使得该用户可以查看文件

/etc/shadow 的内容。

```
root@lzh:~/workpl1# su user1
user1@lzh:/root/workpl1$ sudo cat /etc/shadow
[sudo] password for user1:
Sorry, try again.
[sudo] password for user1:
user1 is not in the sudoers file.  This incident will be reported.
user1@lzh:/root/workpl1$
```

```
root@lzh:~/workpl1# visudo_
```

```
  GNU nano 4.8                  /etc/sudoers.tmp                        Modified
#
# 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:/sbin:/bin:/snap/bin"

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root ALL=(ALL:ALL) ALL
user1 ALL=(ALL:ALL) ALL
# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d



^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos   M-U Undo
^X Exit      ^R Read File  ^\ Replace   ^U Paste Text ^T To Spell  ^_ Go To Line M-E Redo
```

```
user1@lzh:/root/workpl1$ sudo cat /etc/shadow_
```

```
man:*:19235:0:99999:7:::
lp:*:19235:0:99999:7:::
mail:*:19235:0:99999:7:::
news:*:19235:0:99999:7:::
uucp:*:19235:0:99999:7:::
proxy:*:19235:0:99999:7:::
www-data:*:19235:0:99999:7:::
backup:*:19235:0:99999:7:::
list:*:19235:0:99999:7:::
irc:*:19235:0:99999:7:::
gnats:*:19235:0:99999:7:::
nobody:*:19235:0:99999:7:::
systemd-network:*:19235:0:99999:7:::
systemd-resolve:*:19235:0:99999:7:::
systemd-timesync:*:19235:0:99999:7:::
messagebus:*:19235:0:99999:7:::
syslog:*:19235:0:99999:7:::
_apt:*:19235:0:99999:7:::
tss:*:19235:0:99999:7:::
uuidd:*:19235:0:99999:7:::
tcpdump:*:19235:0:99999:7:::
landscape:*:19235:0:99999:7:::
pollinate:*:19235:0:99999:7:::
usbmux:*:19417:0:99999:7:::
sshd:*:19417:0:99999:7:::
systemd-coredump:!!:19417::::::
lzh:$6$3pvWFlkaRnaBnNO5$qfG4dmQutQD/sH6FOXN5gKP8VyCcjj.QYZ7aHC2SsEsQK1TgjDTr7LpqNQRwcrpgRtN7.jLRMkfm
4HEk5D8vX0:19417:0:99999:7:::
lxd:!:19417::::::
fwupd-refresh:*:19441:0:99999:7:::
user1:$6$RxQaSPjzi7FuDpXu$185LgpgGcuITpepwZpDSOwlbZIHBTL7LhRObd/GjhBUqpHmnw1gmt1SESuRajSZ7ORdOf9bwaT
pTtfdU9k5iQ/:19465:0:99999:7:::
user2:$6$.yzjTwbzVbU52c4u$e32zpcjtGx3JMVvexP.wwoJd/xOvQ/TG4zYLCzqC4zPNlk42MSYmFpiNX7gIwyry5vndhb7Woz
6ncSQ5RJBqH1:19465:0:99999:7:::
user3:$6$tO2a4gDTWlYfP.G4$heWof4A.9lkAK5M2TOEdFFK7NjAa.ryy1nzbMyucMYdrbJ5Wu3W89FGUyireICnn.lJ1u4Luqa
VFP9xdd2VWk1:19465:0:99999:7:::
user1@lzh:/root/workpl1$
```

实验五 Shell 程序的创建及条件判断语句

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。

实验内容

定义变量 foo 的值为 200 ，并将其显示在屏幕上（终端上执行）；

```
user1@lzh:/root/workpl1$ foo=100
user1@lzh:/root/workpl1$ echo $foo
100
user1@lzh:/root/workpl1$
```

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）；

```
user1@lzh:/root/workpl1$ bar=100
user1@lzh:/root/workpl1$ test $bar -gt 150
user1@lzh:/root/workpl1$ echo $?
1
user1@lzh:/root/workpl1$ _
```

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 ）；

```
#!/bin/bash
echo "Hostname: $(hostname)"
echo "System time: $(date)"
~
~
```

```
"show_info.sh" 3L, 69C written
root@lzh:~/workpl1# bash show_info.sh
Hostname: lzh
System time: Tue 18 Apr 2023 03:19:07 PM UTC
root@lzh:~/workpl1#
```

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；

所谓水仙花数是指一个 3 位数，该数字每位数字的 3 次幂之和等于其本身，例如：

```
153 == 1^3+3^3+5^3
```

根据上述定义 153 是水仙花数。编写程序时要求首先进行输入参数个数判断，判断是否有输入参数存　在：如果没有则给出提示信息；否则给出该数是否是水仙花数。要求对 153 、 124 和 370 进行测试判 断。

```
#!/bin/bash
if [ $# -eq 0 ]; then
        echo "Please enter a parameter"
        exit 1
fi
input=$1
sum=0
for((i=0; i<${#input}; i++));do
        digit=${input:$i:1}
        sum=$((sum + digit ** ${#input}))
done
if [ $sum -eq $input ]; then
        echo "is the number of daffodlils"
else
        echo "is not a narcussistic number"
fi
~
~
~
```

```
"shuixianhuan.sh" 16L, 296C written
root@lzh:~/workpl1# bash shuixianhuan.sh 153
is the number of daffodlils
root@lzh:~/workpl1# bash shuixianhuan.sh 124
is not a narcussistic number
root@lzh:~/workpl1# bash shuixianhuan.sh 370
is the number of daffodlils
root@lzh:~/workpl1#
```

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；

```
#!/bin/bash
a=$1
b=$2
c=$3
echo $((a+b+c))
~
~
~
~
```

```
"sum.sh" 5L, 43C written
root@lzh:~/workpl1# bash sum.sh 1 2 3
6
root@lzh:~/workpl1# _
```

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,为 C ,60-70 为 D ，小于 60 分为 E 。要求使用 If elif else fi 实现。

```
#!/bin/bash
if [ $1 -ge 90 ]; then
        echo "A"
elif [ $1 -ge 80 ];then
        echo "B"
elif [ $1 -ge 70 ];then
        echo "C"
elif [ $1 -ge 60 ];then
        echo "D"
else
        echo "E"
fi
~
~
```

```
root@lzh:~/workpl1# bash gread.sh 99
A
root@lzh:~/workpl1# bash gread.sh 80
B
root@lzh:~/workpl1# bash gread.sh 60
D
root@lzh:~/workpl1# bash gread.sh 30
E
root@lzh:~/workpl1#
```

实验六 Shell 循环控制语句

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 。

实验内容

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。

```
#!/bin/bash
for file in *.c; do
        cp "$file" ~/workpl2/
done
~
~
```

```
"copy_c_file.sh" 4L, 60C written
root@lzh:~/workpl1# bash copy_c_file.sh _
```

```
root@lzh:~/workpl1# cd ~/workpl2
root@lzh:~/workpl2# ls
a.c  b.c
root@lzh:~/workpl2# _
```

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

```
#!/bin/bash
sum=0
i=0
while [ $i -lt 10 ]; do
        sum=$((sum + i *2))
        i=$((i + 1))
done
echo "sum=$sum"
~
~
~
```

```
"sum_even_number.sh" 8L, 102C written
root@lzh:~/workpl2# bash sum_even_number.sh
sum=90
root@lzh:~/workpl2#
```

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

```
"sum_squares.sh" 8L, 97C written
root@lzh:~/workp12# bash sum_squares.sh
sum=385
root@lzh:~/workp12#
```

4. 运行下列程序，并观察程序的运行结果。将程序中的 --- 分别替换为 break 、 break 2 、 continue 、 continue 2 ，并观察四种情况下的实验结果。

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

```
#!/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
~
~
```

```
"test1.sh" 11L, 157C written
root@lzh:~/workpl2# bash test1.sh
a1234
b1234
c1234
d1234
root@lzh:~/workpl2#
```

```
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  2
                fi
                echo -n $j
        done
        echo ''
done
~
```

```
"test1.sh" 11L, 159C written
root@lzh:~/workpl2# bash test1.sh
a1234root@lzh:~/workpl2#
```

```
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
~
~
```

```
"test1.sh" 11L, 160C written
root@lzh:~/workpl2# bash test1.sh
a1234678910
b1234678910
c1234678910
d1234678910
root@lzh:~/workpl2#
```

```
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 2
                fi
                echo -n $j
        done
        echo ''
done
~
~
```

```
"test1.sh" 11L, 162C written
root@lzh:~/workpl2# bash test1.sh
a1234b1234c1234d1234root@lzh:~/workpl2# _
```

实验七 Shell 函数

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。

实验内容

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；

```
#!/bin/bash
add(){
        sum=$(($1 + $2))
        echo $sum
}
result=$(add $1 $2)
echo "sum=$result"
~
~
```

```
~
"add.sh" 7L, 89C written
root@lzh:~/workpl2# bash add.sh 3 5
sum=8
root@lzh:~/workpl2#
```

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；

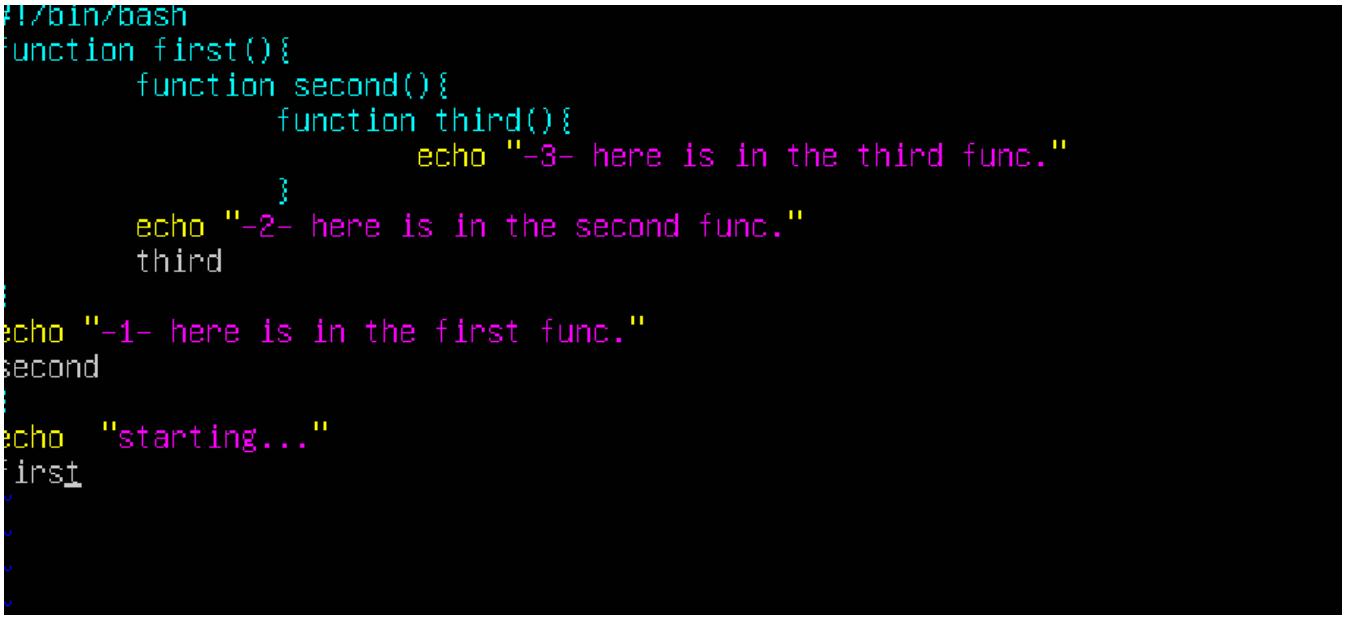
```
#!/bin/bash
factorial(){
        if [ $1 -le 1 ];then
                echo 1
        else
                local prev=$(factorial $(($1-1)))
                local result=$(($1*$prev))
                echo $result
        fi
}
result=$(factorial $1)
echo "The factorial is :$result"
~
~
~
~
~
```

```
~
"factorial.sh" 12L, 204C written
root@lzh:~/workpl2# bash factorial.sh  4
The factorial is :24
root@lzh:~/workpl2# _
```

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..."
```

试运行该程序，并观察程序运行结果，理解函数嵌套的含义。



实验八 sed 和 awk

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。

实验内容

1. 文件 quote.txt 的内容如下所示:

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

试使用 sed 命令实现如下功能:

(1) 删除 $ 符号;

(2) 显示包含 music 文字的行内容及行号;

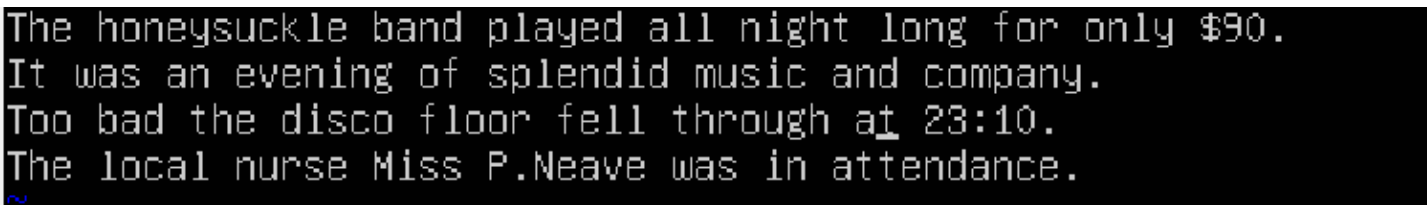
(3) 在第 4 行后面追加内容: "hello world!";

(4) 将文本 "The" 替换为 "Quod" ;

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

(6) 删除第 2 行内容;

(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。



```
"quote.txt" 4L, 201C written
root@lzh:~/workpl2# sed 's/\$//g' 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.
root@lzh:~/workpl2# sed -n '/music/{=;p}' quote.txt
2
It was an evening of splendid music and company.
root@lzh:~/workpl2# sed '4a hello world!' quote.txt
The honeysuckle band played all night long for only $90.
It was an evening of splendid music and company.
Too bad the disco floor fell through at 23:10.
The local nurse Miss P.Neave was in attendance.
hello world!
root@lzh:~/workpl2# sed 's/The/Quod/g' quote.txt
Quod honeysuckle band played all night long for only $90.
It was an evening of splendid music and company.
Too bad the disco floor fell through at 23:10.
Quod local nurse Miss P.Neave was in attendance.
root@lzh:~/workpl2# sed '2d' quote.txt
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.
```

```
root@lzh:~/workpl2# var="evening"
root@lzh:~/workpl2# sed -n "/$var/p" quote.txt
It was an evening of splendid music and company.
root@lzh:~/workpl2#
```

2. 文件 numbers.txt 的内容如下所示：

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

注：每个冒号前后都有空格。

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

```
"numbers.txt" 2L, 36C written
root@lzh:~/workpl2# awk '{print $2}' numbers.txt
:
:
root@lzh:~/workpl2# awk -F ':'  '{print $2}' numbers.txt
 two
 five
root@lzh:~/workpl2# _
```

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。

例如 foo.txt 的内容为：

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

则输出为：

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

```
#!/bin/bash
count=0
while read line; do
        for num in $line; do
                if (( num % 2 == 0 )); then
                        echo $num
                        count=$((count_+ 1))
                fi
        done
done < foo.txt
echo $count
~
```

```
2 4 3
15 45 78
~
~
```

```
"foo.sh" 12L, 168C written
root@lzh:~/workp12# bash foo.sh
2
4
78
3
root@lzh:~/workp12#
```

4. 脚本的内容如下所示：

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

试运行该脚本，并理解该脚本实现的功能。

这个脚本使用 awk 命令在 info.txt 文件中搜索用户输入的模式。脚本首先使用 read 命令提示用户输入要搜索的模式，然后将该模式存储在变量 pattern 中。接下来，脚本使用 awk 命令在 info.txt 文件中搜索与模式匹配的行。对于每个匹配的行，awk 会将匹配计数器 nmatches 加 1 并打印该行。最后，脚本会输出匹配的行数。