# 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 12 日 |
| 学　　号 | 2021223115 | 姓　　名 | 李梦婷 |
| 专业班级 | 计算机科学与技术 05 班 | | |
| 指导教师 | 卢洋 | | |

东北林业大学

信息与计算机科学技术实验中心

一、 实验目的

1、掌握Linux下文件和目录操作命令：cd、ls、mkdir、rmdir、rm

2、掌握Linux下文件信息显示命令：cat、more、head、tail

3、掌握Linux下文件复制、删除及移动命令：cp、mv

4、掌握 Linux 的文件排序命令：sort

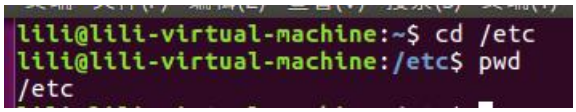
二、 实验环境

（1）计算机的硬件配置 PC 系列微机。

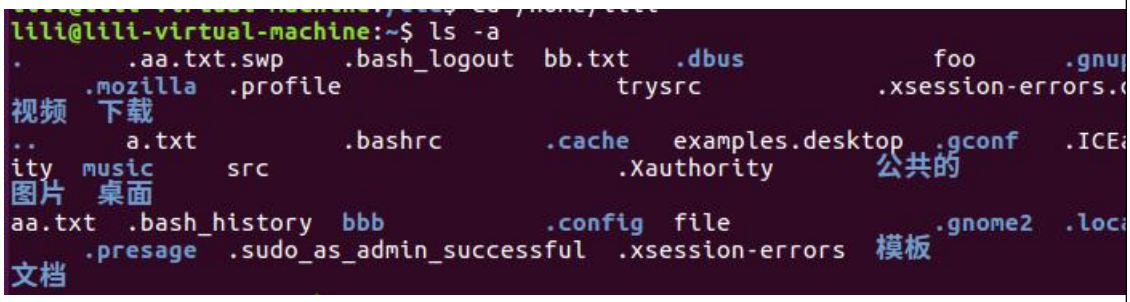
（2）计算机的软件配置 VMware 虚拟机软件及 Ubuntu 虚拟机。

三、 实验内容及结果

1. 使用命令切换到/etc 目录，并显示当前工作目录路径

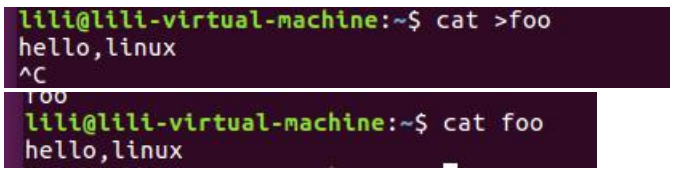



2、使用命令显示/home/用户名 目录下所有文件目录的详细信息，包括隐藏文件。




3、使用命令创建目录/home/用户名/linux，然后删除该目录。


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







5、使用命令创建目录/home/用户名/bak，然后将/home/用户名/foo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

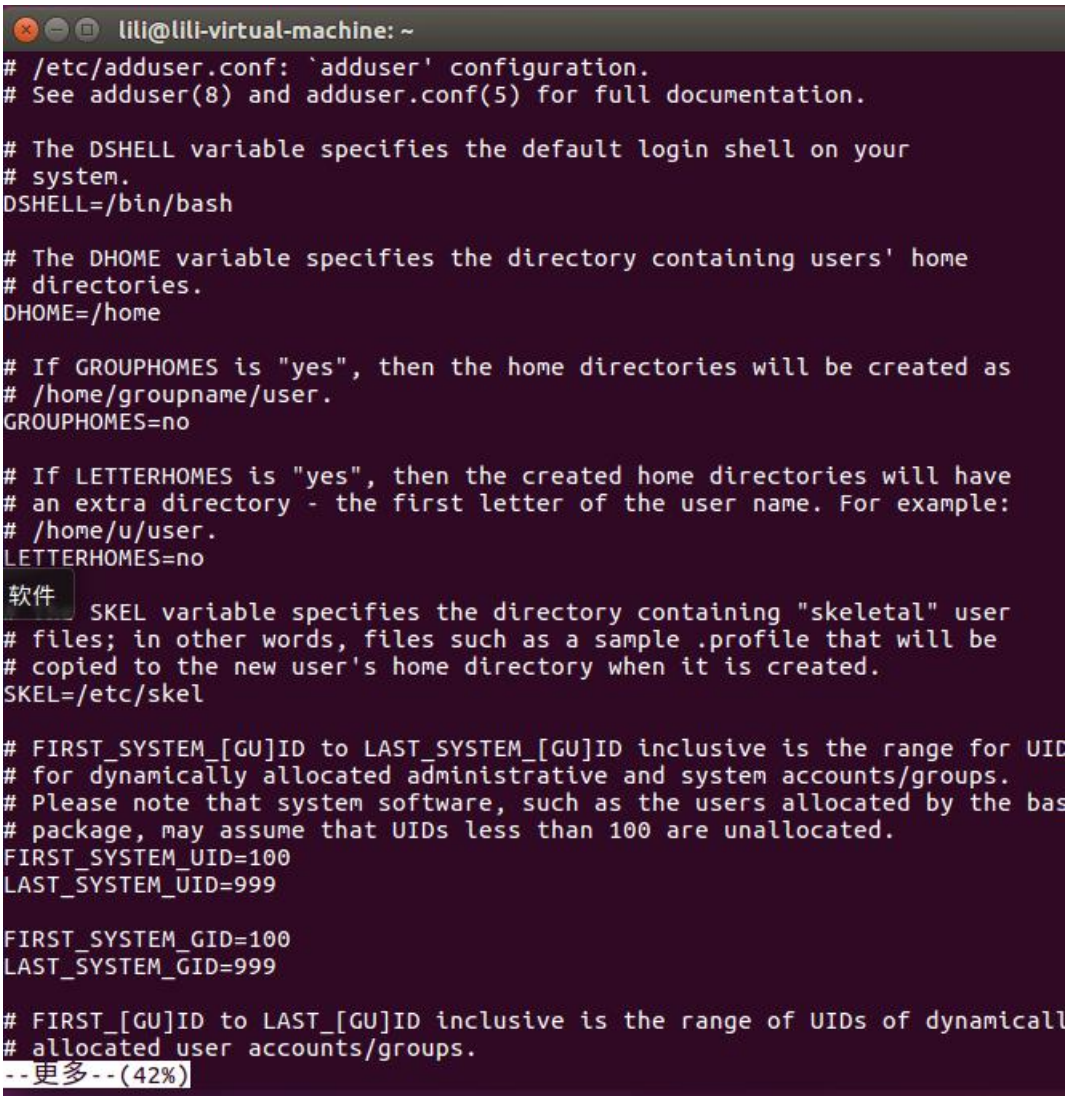
删除。

```
lili@lili-virtual-machine:~$ mkdir bak
lili@lili-virtual-machine:~$ cp -r foo bak
lili@lili-virtual-machine:~$ cd bak
lili@lili-virtual-machine:~/bak$ rm -i foo
rm: 是否删除普通文件 'foo'?  y
lili@lili-virtual-machine:~/bak$ cd ..
lili@lili-virtual-machine:~$ rmdir bak
```

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

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

lili@lili-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
```

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/用户名目录下创建文件bar.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：

```
lili@lili-virtual-machine:~$ cat >bar.txt
google 110 5000
baidu  100 5000
guge   50  3000
sohu   100 4500
^C
```

（1）按公司字母顺序排序

```
lili@lili-virtual-machine:~$ sort bar.txt
baidu  100 5000
google 110 5000
guge   50  3000
sohu   100 4500
```

（2）按公司人数排序

```
sohu   100 4500
lili@lili-virtual-machine:~$ sort -n -t' ' -k 2 bar.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

```
lili@lili-virtual-machine:~$ sort -t' ' -k2n -k3n bar.txt
guge   50  3000
baidu  100 5000
sohu   100 4500
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

```
lili@lili-virtual-machine:~$ sort -t' ' -k3nr -k2n bar.txt
google 110 5000
baidu  100 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。

```
lili@lili-virtual-machine:~$ sort -t' ' -k1,2 bar.txt
baidu  100 5000
google 110 5000
guge   50  3000
sohu   100 4500
```

四、 实验过程分析与讨论

Sort 命令

- 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
- -m 将已排序的输入文件，合并为一个排序后的输出数据流。
- -n 以整数类型比较字段
- -o outfile 将输入写到指定文件，而非标准输出。如果该文件为输入文件之一，则 sort 在进行排序写到输入文件之前，会先将它复制到一个临时文件
- -r 倒置排序的顺序为 由大至小（descending）,而非默认的由小至大（ascending）
- -t char 使用单个字符 char 作为默认的字段分割字符，取代默认的空白字符。
- -u 只有唯一的记录，丢弃所有具有相同键值的记录，只留其中的第一条。只有键值字段是重要的，也就是说：被丢弃的记录其他部分可能是不同值。
- 行为模式：sort 会读取指定的文件，如果未给定文件，则读取标准输入，在将排序好的数据写至标准输出。
- -b 忽略开头的空白
- -c 检查输入是否已正确排序，如输入未经排序，但退出码(exit code)为非零值，则不会有任何输出

- -d　字典顺序：仅文字数字与空白才有意义
- -g　一般数值：以浮点数字类型比较字段。这个选项的运作有点类似 -n.差别仅在于这个选项的数字可能有小数点及指数。（仅 GNU 版本提供此功能）
- -f　以不管字母大小写的方式排序
- -i　忽略无法打印的字符

五、指导教师意见

指导教师签字：卢洋

# 实验报告

| | | | |
|---|---|---|---|
| 实验名称 | **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 | 2021223115 | 姓　　名 | 李梦婷 |
| 专业班级 | 计算机科学与技术 5 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、wc；
2、掌握 Linux 下文件打包命令：tar ；
3、掌握 Linux 下符号链接命令和文件比较命令：ln、comm、diff；
4、掌握 Linux 的文件权限管理命令：chmod 。

## 六、 实验环境

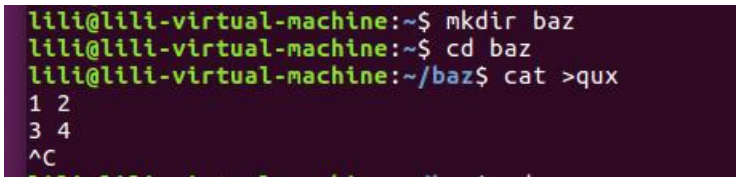
（1）计算机的硬件配置 PC 系列微机。

（2）计算机的软件配置 VMware 虚拟机软件及 Ubuntu 虚拟机。

## 七、 实验内容及结果

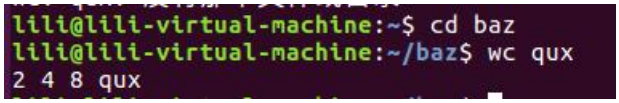
1、查找指定文件

（1）在用户目录下新建目录 baz，在 baz 下新建文件 qux，并写如任意几行内容；

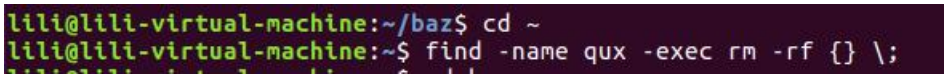



（2）在用户主目录下查找文件 qux，并显示该文件位置信息。

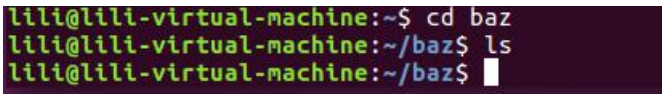

（3）统计 qux 文件中所包含的行数、字数和字节数。




（4）在用户主目录下查找文件 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。




2、文件打包

（1）在用户主目录下新建文件夹 path1，在 path1 下新建文件 file1 和 file2。

```
lili@lili-virtual-machine:~$ mkdir path1
lili@lili-virtual-machine:~$ cd path1
lili@lili-virtual-machine:~/path1$ touch file1
lili@lili-virtual-machine:~/path1$ touch file2
lili@lili-virtual-machine:~/path1$ ls
file1  file2
```

（2）在用户主目录下新建文件夹 path2，在 path2 下新建文件 file3。

```
lili@lili-virtual-machine:~$ mkdir path2
lili@lili-virtual-machine:~$ cd path2
lili@lili-virtual-machine:~/path2$ touch file3
```

（3）在用户主目录下新建文件 file4。

```
lili@lili-virtual-machine:~$ touch file4
lili@lili-virtual-machine:~$ ls
aa.txt   bar.txt  bb.txt            file4  path1  trysrc  视频  下载
abc      baz      examples.desktop  foo    path2  公共的  图片  桌面
a.txt    bbb      file              music  src    模板    文档
```

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

```
lili@lili-virtual-machine:~$ tar -cvf package.tar file4 path1
终端
path1/file2
path1/file1
```

（5）查看包 package.tar 的内容。

```
lili@lili-virtual-machine:~$ tar -tvf package.tar
-rw-rw-r-- lili/lili         0 2023-05-21 22:18 file4
drwxrwxr-x lili/lili         0 2023-05-21 22:10 path1/
-rw-rw-r-- lili/lili         0 2023-05-21 22:10 path1/file2
-rw-rw-r-- lili/lili         0 2023-05-21 22:10 path1/file1
lili@lili-virtual-machine:~$
```

（6）向包 package.tar 里添加文件夹 path2 的内容。

```
-rw-rw-r-- lili/lili         0 2023-05-21 22:10 path1/file1
lili@lili-virtual-machine:~$ tar -rf package.tar path2
lili@lili-virtual-machine:~$ tar -tf package.tar
file4
path1/
path1/file2
path1/file1
path2/
path2/file3
```

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

```
lili@lili-virtual-machine:~$ mkdir path3
lili@lili-virtual-machine:~$ cp package.tar path3
lili@lili-virtual-machine:~$ ls path3
package.tar
```

（8）进入 path3 文件夹，并还原包 package.tar 的内容。

```
lili@lili-virtual-machine:~$ cd path3
lili@lili-virtual-machine:~/path3$ tar -xvf package.tar
file4
path1/
path1/file2
path1/file1
path2/
path2/file3
lili@lili-virtual-machine:~/path3$ ls
file4  package.tar  path1  path2
```

3、符号链接内容

（1）新建文件 foo.txt,内容为 123。

```
zzc@zzc-virtual-machine:~$ cat> a.txt
123345
^Z
[7]+  已停止                 cat > a.txt
zzc@zzc-virtual-machine:~$
```

（2）建立 foo.txt 得硬链接文件 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。

```
lili@lili-virtual-machine:~$ ln foo.txt bar.txt
lili@lili-virtual-machine:~$ comm foo.txt bar.txt
                123
lili@lili-virtual-machine:~$
```

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

```
                123
lili@lili-virtual-machine:~$ ls -i -F foo.txt
667249 foo.txt
lili@lili-virtual-machine:~$ ls -i -F bar.txt
667249 bar.txt
```

（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。

```
lili@lili-virtual-machine:~$ cat >bar.txt
abc
^C
lili@lili-virtual-machine:~$ comm foo.txt bar.txt
                abc
```

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

```
lili@lili-virtual-machine:~$ rm foo.txt
lili@lili-virtual-machine:~$ ls -i -F bar.txt
667249 bar.txt
lili@lili-virtual-machine:~$ cat bar.txt
abc
```

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

```
lili@lili-virtual-machine:~$ ln -s bar.txt baz.txt
lili@lili-virtual-machine:~$ df -i bar.txt baz.txt
文件系统         Inode 已用(I) 可用(I) 已用(I)% 挂载点
/dev/sda1      1248480  232022 1016458      19% /
/dev/sda1      1248480  232022 1016458      19% /
lili@lili-virtual-machine:~$
```

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

```
lili@lili-virtual-machine:~$ rm bar.txt
lili@lili-virtual-machine:~$ cat baz.txt
cat: baz.txt: 没有那个文件或目录
lili@lili-virtual-machine:~$
```

4、权限管理

（1）新建文件 qux.txt;

```
zzc@zzc-virtual-machine:~$ cd test/
zzc@zzc-virtual-machine:~/test$ cat >tt.txt
bash: tt.txt: 权限不够
zzc@zzc-virtual-machine:~/test$
```

（2）为文件 qux.txt 增加执行权限（所有用户都可以执行）。

```
lili@lili-virtual-machine:~$ chmod a+x qux.txt
lili@lili-virtual-machine:~$ ls qux.txt
qux.txt
lili@lili-virtual-machine:~$ ll qux.txt
-rwxrwxr-x 1 lili lili 0 5月  21 23:20 qux.txt*
```

八、 实验过程分析与讨论

一个命令拥有很多的参数，能够实现各种各样的功能。其中 ln 命令生成硬链接时是使当前文件指向该文件的索引号，这样删除另一个文件，通过硬链接仍然可以访问到以前的数据。而生成软连接时实际上是生成了一个包含另一文件的位置信息的文本文件，当源文件删除，该文件便不可访问内容。

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 17 日 |

| | | | |
|---|---|---|---|
| 学　　号 | 2021223115 | 姓　　名 | 李梦婷 |
| 专业班级 | 计算机科学与技术 5 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 编译器的使用方法

## 十、 实验环境

（1）计算机的硬件配置 PC 系列微机。

（2）计算机的软件配置 VMware 虚拟机软件及 Ubuntu 虚拟机。

## 十一、 实验内容及结果

1、vi 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；

```
lili@lili-virtual-machine:~$ mkdir workspace1
lili@lili-virtual-machine:~$ ls
aa.txt    bbb               foo          path3       公共的   下载
abc       bb.txt            music        qux.txt     模板     桌面
a.txt     examples.desktop  package.tar  src         视频
baz       file              path1        trysrc      图片
baz.txt   file4             path2        workspace1  文档
```

（2）进入目录 workspace1

```
lili@lili-virtual-machine:~$ cd workspace1
```

vi

（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

test.c 文件内容为：

```
int main( )
{
printf(“hello world!\n”);
}
```

```
lili@lili-virtual-machine:~/workspace1$ vi test.c
```

```
int main( )
{
        printf(“hello world!\n”);
}
```

（4）保存 test.c 的内容，并退出

```
~
:wq
```

（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。

```
hadoop2@hadoop102:~$ gcc test.c -o test
hadoop2@hadoop102:~$ ./test
Hello world!
hadoop2@hadoop102:~$
```

2、vi 编辑器的详细使用

（1）在用户主目录下建一个名为 workspace2 的目录。

```
lili@lili-virtual-machine:~$ mkdir workspace2
lili@lili-virtual-machine:~$ ls
aa.txt   baz.txt           file   package.tar  qux.txt  workspace1  视频  桌
abc      bbb               file4  path1        src      workspace2  图片
a.txt    bb.txt            foo    path2        test.c   公共的      文档
baz      examples.desktop  music  path3        trysrc   模板        下载
```

（2）进入 workspace2 目录。

```
lili@lili-virtual-machine:~$ cd workspace2
lili@lili-virtual-machine:~/workspace2$
```

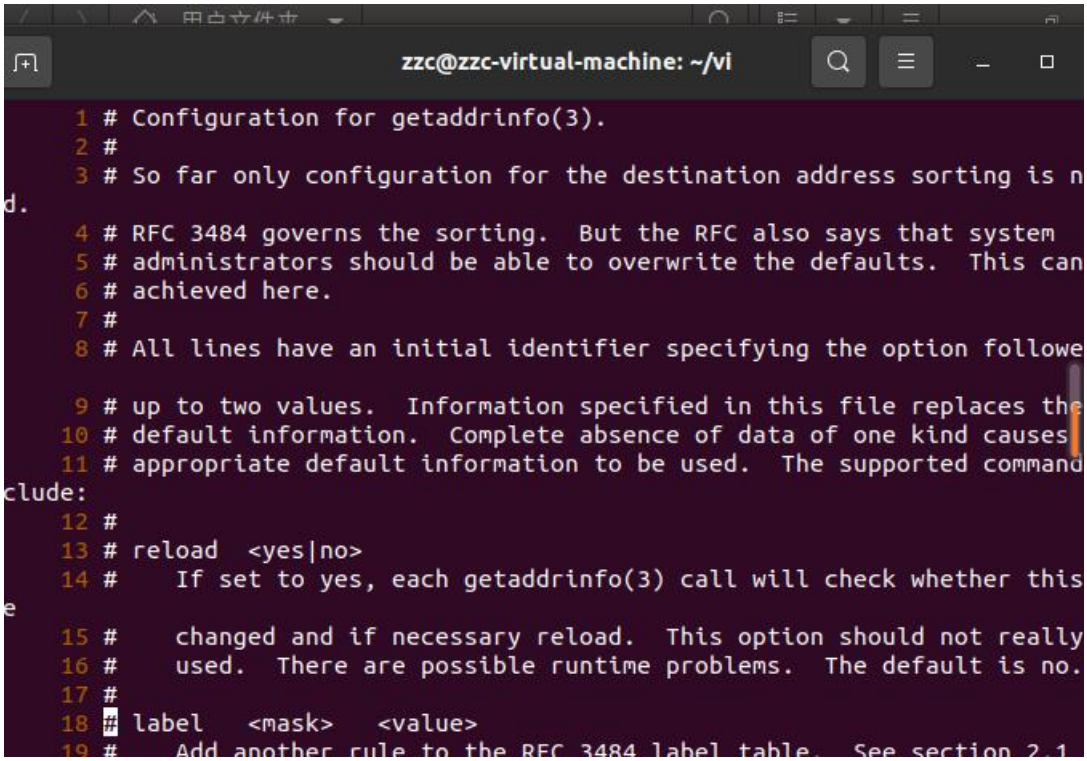
（3）使用以下命令：将文件/etc/gai.conf 的内容复制到当前目录下的新建文件 gai.conf 中；

```
lili@lili-virtual-machine:~/workspace2$ cat /etc/gai.conf > ./gai.conf
lili@lili-virtual-machine:~/workspace2$ ls
gai.conf
```

（4）使用 vim 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 inclu
#
# 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
"gai.conf" 65 lines, 2584 characters
```

（5)将光标移到第 18 行。




（6）复制该行内容。

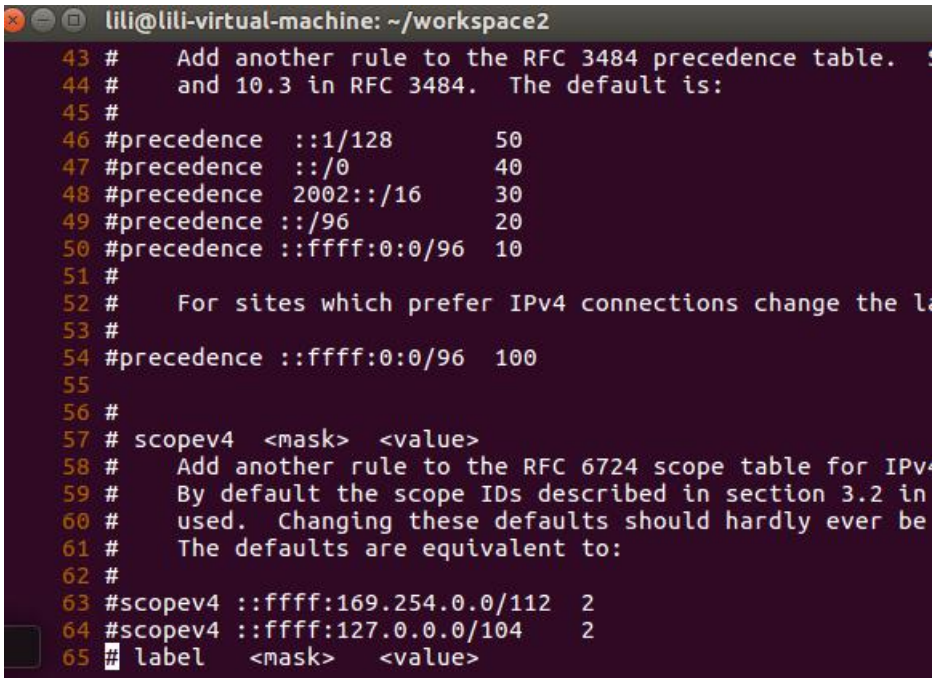
yy

（7）将光标移到最后一行行首。

G

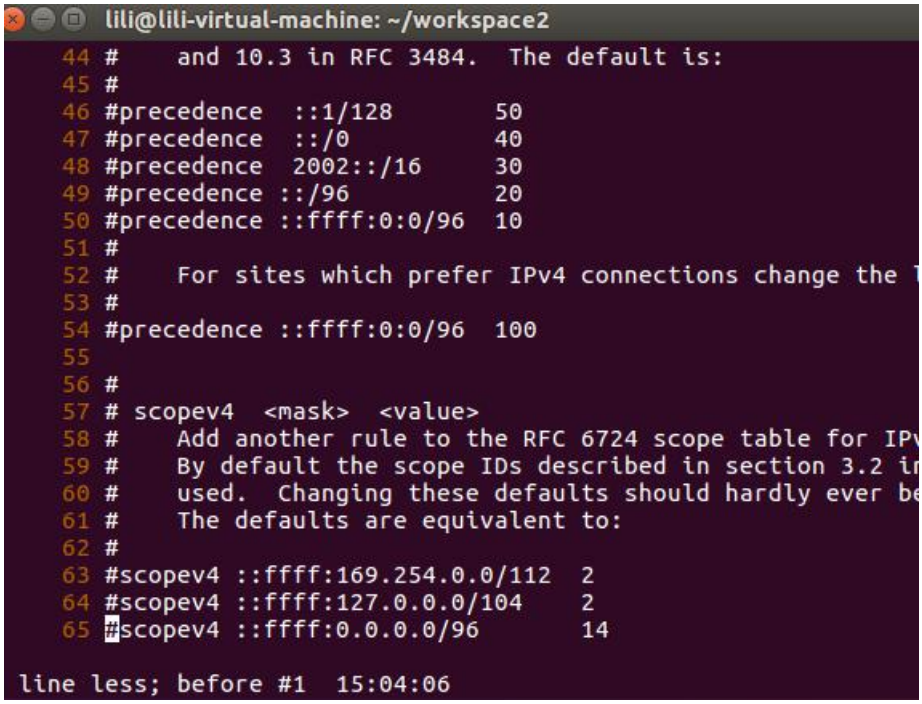
（8）粘贴复制行的内容。

P




9. 撤销第 8 步的动作；

u



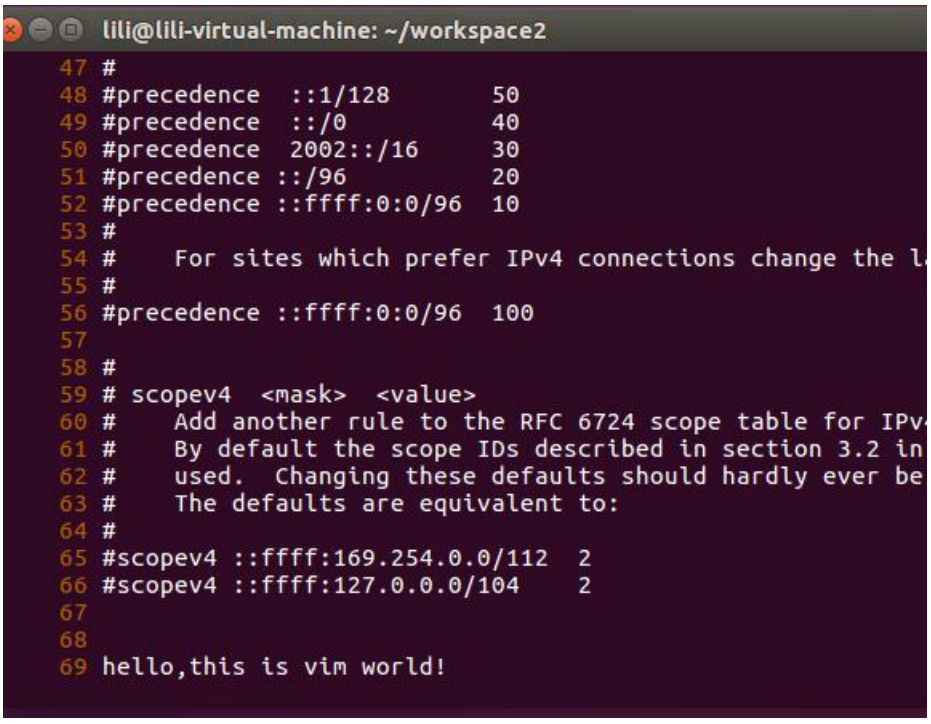

（10）存盘但不退出。

:w


（11）将光标移到首行。

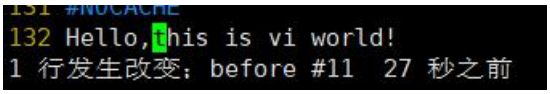
gg

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




（13）删除字符串“this“。

光标指向 this 首字母+dw




（14）强制退出vim，不存盘。

:q!


## 十二、　实验过程分析与讨论

在运行 gcc test.txt -o test 时遇到无法编译的情况，通过使用 sudo apt-get install build-essential 命令解决了该问题

五、指导教师意见

指导教师签字： 卢洋

# 实验报告

| | | | |
|---|---|---|---|
| 实验名称 | **实验四 用户和用户组管理** | | |
| 实验教室 | 丹青 922 | 实验日期 | 2023 年 5 月 18 日 |
| 学　　号 | 2021223115 | 姓　　名 | 李梦婷 |
| 专业班级 | 计算机科学与技术 5 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. 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@lili-virtual-machine:/home/lili# useradd foo -s /bin/sh -b /home/foo -c foo
 -p 123456
```

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@lili-virtual-machine:/home/lili# usermod foo -u 2000 -s /bin/sh
root@lili-virtual-machine:/home/lili# su foo
```

3. 从用户 foo 切换到 root；

```
$ su root
密码:
root@lili-virtual-machine:/home/lili#
```

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@zzc-virtual-machine:/home/zzc# userdel jone -r
userdel：jone 信件池（/var/mail/jone）未找到
userdel：未找到 jone 的主目录“/home/jone/jone”
root@zzc-virtual-machine:/home/zzc# su jone
su：用户 jone 不存在
root@zzc-virtual-machine:/home/zzc#
```

5.5. 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；

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

```
user1:123456
user2:12345678
~
```

```
root@zzc-virtual-machine:/home/zzc# newusers <userfile
newusers：第 3 行：无效行
newusers：发现错误，忽略改动
root@zzc-virtual-machine:/home/zzc# newusers <userfile
root@zzc-virtual-machine:/home/zzc# cat userpasswd | chpasswd
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@lili-virtual-machine:/home# groupadd group1 -g 3000
root@lili-virtual-machine:/home#
```

7、在用户组 group1 中添加两个之前批量创建的用户。

```
root@zzc-virtual-machine:/home/zzc# usermod -g group1 user1
root@zzc-virtual-machine:/home/zzc# usermod -g group1 user2
root@zzc-virtual-machine:/home/zzc#
```

```
user1:x:2000:3000:user1,,,:/home/user1:/bin/bash
user2:x:2001:3000:user2,,,:/home/user2:/bin/bash
```

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）。

```
root@zzc-virtual-machine:/home/zzc# su user1
user1@zzc-virtual-machine:/home/zzc$ sudo vi /etc/shadow
[sudo] user1 的密码：
user1 不在 sudoers 文件中。此事将被报告。
user1@zzc-virtual-machine:/home/zzc$
```

```
root@zzc-virtual-machine:/home/zzc# vi /etc/sudoers
```

```
#
# 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
~
~
```

```
root@zzc-virtual-machine:/home/zzc# sudo vi /etc/shadow
```


## 十六、 实验过程分析与讨论

示例：

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 ~]# grouadd -g 800 ou

创建系统组群 chinese

[root@localhost ~]# groupadd -r chinese

主要概念：

1、基本上，一个组就是一个整数组 ID（gid）
lzgonline:x: 500:
2、每个在系统上运行的进程都是属于一个组的集合（gids）
3、/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 | 2021223115 | 姓　　名 | 李梦婷 |
| 专业班级 | 计算机科学与技术 5 班 | | |
| 指导教师 | 卢洋 | | |

东北林业大学

# 信息与计算机科学技术实验中心

## 十七、 实验目的

1、掌握 Shell 程序的创建过程及 Shell 程序的执行方法。
2、掌握 Shell 变量的定义方法，及用户定义变量、参数位置等。
3、掌握变量表达式，包括字符串比较、数字比较、逻辑测试、文件测试。
4、掌握条件判断语句，如 if 语句、case 语句。

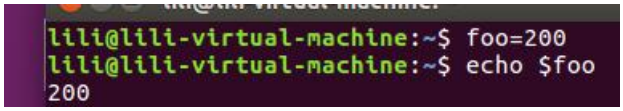
## 十八、 实验环境

（1）计算机的硬件配置 PC 系列微机。

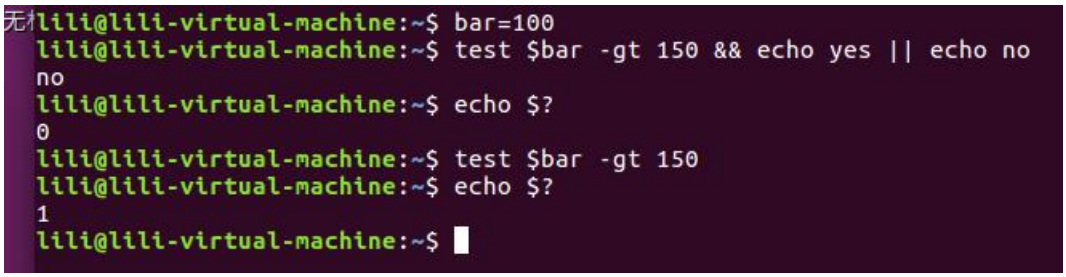
（2）计算机的软件配置 VMware 虚拟机软件及 Ubuntu 虚拟机。

## 十九、　实验内容及结果

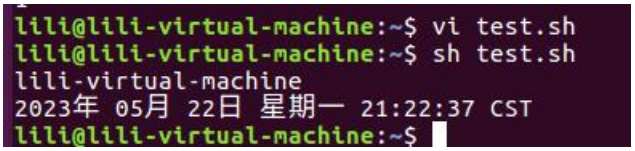
1、定义变量 foo 的值为 200，并将其显示在屏幕上。（终端上执行）

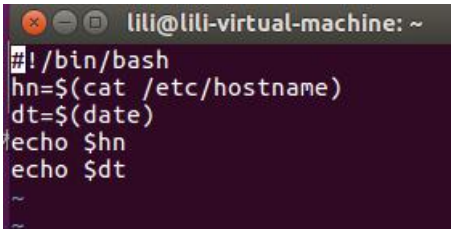



2、定义变量 bar 的值为 100，并使用 test 命令比较其值是否大于 150，并显示 test 命令的退出码。（终端上执行）




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=13+33+53，153 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。

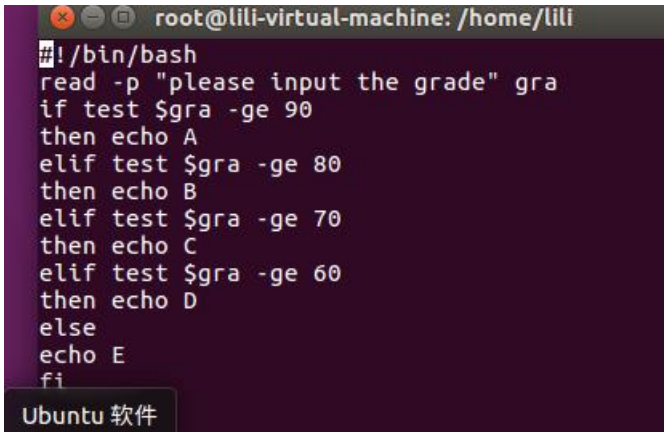
```
root@lili-virtual-machine: /home/lili
#!/bin/bash
read -p "please input the number :" num
numx=$num
num1=$((num%10))
num=$((num/10))
num2=$((num%10))
num=$((num/10))
num3=$((num%10))

num=$((num1*num1*num1+num2*num2*num2+num3*num3*num3))
if test $num -eq $numx
then echo 'this number is daffodils'
else echo 'this number is not daffodils'
fi

~
~
~
~
~
~
~
~
"shuixianhua.sh" 15 lines, 305 characters
```

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；

```
root@lili-virtual-machine:/home/lili# sh sum.sh 1 2 3
the sum=6
```

```
eOffice Writer  ot@lili-virtual-machine: /
#!/bin/bash
a=$1
b=$2
c=$3
num=$((a+b+c))
echo "the sum=$num"
~
```

6、创创建一个 Shell 程序，输入学生成绩，给出该成绩对应的等级：90 分以上为 A，80-90 为 B，70-80 为 C，60-70 为 D，小于 60 分为 E。要求使用
实现。
if
elif
else
fi




```
root@lili-virtual-machine:/home/lili# vi grade.sh
root@lili-virtual-machine:/home/lili# sh grade.sh
please input the grade99
A
```

## 二十、 实验过程分析与讨论

在进行数字运算的时候，要用$(())的格式。

shell 中的逻辑判断一般用 if 语句，if 语句中通常用[]来表示条件测试，可以比较字符串、判断文件是否存等。备注：[ ] 中表达式两边与括号之间要有空格

if … else 语句常用基本的语法如下：

1.if [];then fi 语句

建一个测试脚本 test.sh 如下

```
#!/bin/bash
a=$1
b=$2
if [ $a == $b ];then
   echo "a and b is equal"
fi
if [ $a != $b ];then
   echo "a and b is not equal"
fi
```

执行命令 sh test.sh 2 3 给参数$1 和$2 赋值 2 和 3，输出结果 a and b is not equal

不加 else 的 if 语句表达式成立执行 then 后面的语句，表达式不成立则不执行任何命令。

2.if [];then else fi 语句

```
      if [ expression ];then
            executed Statement_expression_true
            else
            executed Statement_expression_false
        fi
```

备注：expression 表达式 和方括号[ ]之间必须有空格，否则会有语法错误。如果表达式成立，then 后面的语句将会被执行；如果表达式不成立则执行 else 后面的语句。

3.if [];then elif []; then else fi 语句，哪个 expression 表达式成立则执行哪个 then 后面的语句，否则执行 else 后面的语句。

```
if [ expression1 ];then
    executed Statement_expression1_true
 elif [ expression2 ];then
    executed Statement_expression2_true
 else
    executed Statement_expression1_2_false
 fi
#!/bin/bash
a=$1
b=$2
if [ $a == $b ];then
   echo "a and b is equal"
elif [ $a -lt $b ];then
   echo "a less than b"
else
   echo "a bigger than b"
fi
```

例如建个测试脚本 test.sh 如上，执行命令 sh test.sh 2 3 给参数$1、$2 赋值 2、3，输出结果 a less than b；执行 sh test.sh 3 2 结果为 a bigger then b

```
#!/bin/bash
a=$1
b=$2
if [ $a == $b ];then
   echo "a and b is equal"
else
   if [ $a -lt $b ];then
     echo "a less than b"
   else
      echo "a bigger than b"
   fi
fi
```

些

4.if … else 语句也经常与 test 命令结合使用，test 命令用于检查某个条件是否成立，与方括号[ ]功能类似

```
#!/bin/bash
a=$1
b=$2
if test $a == $b;then
   echo "a and b is equal"
else
   echo "a and b is not equal"
fi
```

例如上述脚本，其中 if test $a == $b;与 if [ $a == $b ];效果一样。

5.if 语句常用命令选项有：

== or =： 等于
-eq ： 等于
-ne ： 不等于
-gt ： 大于
-ge ： 大于等于
-lt ： 小于
-le ： 小于等于

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 | 2021 年 5 月 19 日 |
| 学　　号 | 2021223115 | 姓　　名 | 李梦婷 |
| 专业班级 | 计算机科学与技术 5 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

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。

```
root@zzc-virtual-machine:/home/zzc# sh move.shi
Please input a file name: .
./chazhao.sh
./grade.sh
./shuixian.sh
./test.sh
./chazhao.sh
./grade.sh
./shuixian.sh
root@zzc-virtual-machine:/home/zzc# ls
chazhao.sh  grade.sh  kpl  movedir  move.shi  shuixian.sh  test.sh
kpl@hadoop100:~/linuxlearn$ ls movedir/
chazhao.sh  grade.sh  shuixian.sh  test.sh
kpl@hadoop100:~/linuxlearn$
```

```
#!/bin/bash
#function 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）

read -p "Please input a file 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。

```
lili@lili-virtual-machine: ~
lili@lili-virtual-machine:~$ vi oushu.sh
lili@lili-virtual-machine:~$ sh oushu.sh
90
lili@lili-virtual-machine:~$
```

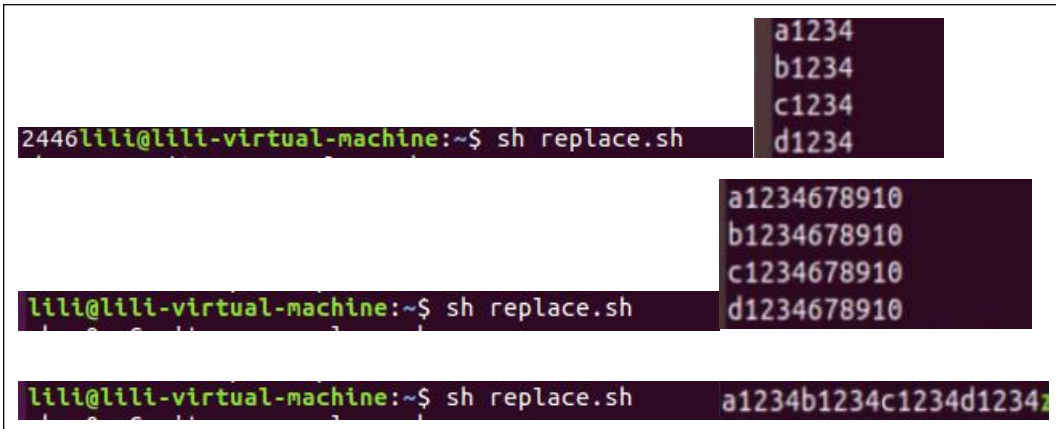
（3） 编写 shell 脚本，利用 until 循环求 1 到 10 的平方和。

```
#!/bin/bash
#function 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"
~
```

```
lili@lili-virtual-machine:~$ vi ping.sh
lili@lili-virtual-machine:~$ sh ping.sh
385
lili@lili-virtual-machine:~$ cat ping.sh
#!/bin/bash
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，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 in12345678910; do
  if [[ $j-eq5 ]]; then
  ---
  fi
  echo  -n $j
  done
  echo"
done
```




## 二十四、 实验过程分析与讨论

Until 循环条件中若为假继续循环，若为真则跳出循环，这一点是我们平常容易混淆的地方。另外在引用变量时千万不要忘记加‘$‘符号。Break 是终止一层循环，continue 是跳出当前循环；而 break2 是终止两层循环，continue2 是跳过两个循环。

for 循环使用
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

```
#!/bin/bash

for varible1 in {1..5}
#for varible1 in 1 2 3 4 5
do
        echo "Hello, Welcome $varible1 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。

```
#!/bin/bash

sum=0

for i in {1..100..2}
do
      let "sum+=i"
done

echo "sum=$sum"
```

通过 i 的按步数 2 不断递增，计算 sum 值为 2500。同样可以使用 seq 命令实现按 2 递增来计算 1～100 内的所有奇数之和，for i in $(seq 1 2 100)，seq 表示起始数为 1，跳跃的步数为 2，结束条件值为 100。
for 循环对字符串进行操作，例如通过 for 循环显示当前目录下所有的文件。

```
#!/bin/bash

for file in $( ls )
#for file in *
do
    echo "file: $file"
done
```

## 五、指导教师意见

指导教师签字： 卢洋

# 实验报告

| | | | |
|---|---|---|---|
| 实验名称 | 实验七 Shell 函数 | | |
| 实验教室 | 丹青 922 | 实验日期 | 2023 年 5 月 20 日 |
| 学　　号 | 2021223115 | 姓　　名 | 李梦婷 |
| 专业班级 | 计算机科学与技术 5 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

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；

```
lili@lili-virtual-machine:~$ vi he.sh
lili@lili-virtual-machine:~$ sh he.sh 1 2
3
```

```
lili@lili-virtual-machine:~$ cat he.sh
#!/bin/bash

sum(){
        return $(($1+$2))
}

sum $1 $2
echo $?
```

(2) 编写 shell 脚本，该脚本中定义一个递归函数，求 n 的阶乘。

```
lili@lili-virtual-machine:~$ vi jie.sh
lili@lili-virtual-machine:~$ sh jie.sh
please input a positive number :5
```

```
#!/bin/bash

factor(){
        if [ $1 -eq 1 ]
         then  echo 1
        else
           local temp=$[ $1 - 1 ]
           local num=$(factor $temp)
           echo $[ $1 * $num ]
        fi
}
read -p "please input a positive number:" n
sum=$(factor $n)
echo $sum
```

(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 "-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..."
```

```
lili@lili-virtual-machine:~$ sh funtest.sh
starting...
-1- here is in the first func.
-2- here is in the second func.
-3- here is in the third func.
```

## 二十八、 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<br><br>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 21 日 |

| | | | |
|---|---|---|---|
| 学　　号 | 2021223115 | 姓　　名 | 李梦婷 |
| 专业班级 | 计算机科学与技术 5 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 shel 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

```
lili@lili-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 atb23:10.
The local nurse Miss P.Neave was in attendance.
```

（2） 显示包含 music 文字的行内容及行号

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

（3） 在第 4 行后面追加文件“hello world！”

```
lili@lili-virtual-machine:~$ cat quote.txt |sed '4a hello world'
The honeysuckle band played all night long for only $90.
It was an evening of splendid music and company.
Too bad the disco floor fell through atb23:10.
The local nurse Miss P.Neave was in attendance.
hello world
```

（4） 将文本“The”修改为“Quod”

```
lili@lili-virtual-machine:~$ cat quote.txt |sed 's/The/Quod/g'
Quod honeysuckle band played all night long for only $90.
It was an evening of splendid music and company.
Too bad the disco floor fell through atb23:10.
Quod local nurse Miss P.Neave was in attendance.
```

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

```
lili@lili-virtual-machine:~$ sed '3cThis is the third line.' quote.txt
The honeysuckle band played all night long for only $90.
It was an evening of splendid music and company.
This is the third line.
The local nurse Miss P.Neave was in attendance.
```

（6） 删除第 2 行内容。

```
lili@lili-virtual-machine:~$ sed '2d' quote.txt
The honeysuckle band played all night long for only $90.
Too bad the disco floor fell through atb23:10.
The local nurse Miss P.Neave was in attendance.
```

（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。

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

2、已知文件 numbers.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

```
lili@lili-virtual-machine:~$ awk -F " " '{printf $2}' numbers.txt
::lili@lili-virtual-machine:~$ awk -F " " '{printf $2}' numbers.txt
::lili@lili-virtual-machine:~$ awk -F ":" '{printf $2}' numbers.txt
 two  five lili@lili-virtual-machine:~$
```

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

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 内容为：

2 4 3

15 46 79

则输出为：

2

4

46

Numbers：3

```
lili@lili-virtual-machine:~$ cat foo.txt | awk 'BEGIN{sum=0}{for(i=1;i<=
NF;i++){if($i%2==0){printf $i;sum+=1}}}END{printsum}'
2446lili@lili-virtual-machine:~$
```

4、脚本的内容如下所示：#!/bin/bashread －p ”enter search pattern: “patternawk “/$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。

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

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
脚本实现功能是匹配当前字符在文件中出现了多少行
首先输入你要匹配的字符串，脚本中指定的文件为 info.txt。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

## 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 的后面可以接字串，而这些字串会在目标行末尾追加～
- c：取代， c 的后面可以接字串，这些字串可以取代 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 的后面可以接字串，而这些字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