

实验报告

实验名称	实验一 Linux 常用命令（一）		
实验教室	丹青 901	实验日期	2023 年 5 月 12 日
学 号	2021223121	姓 名	吴玉琴
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

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

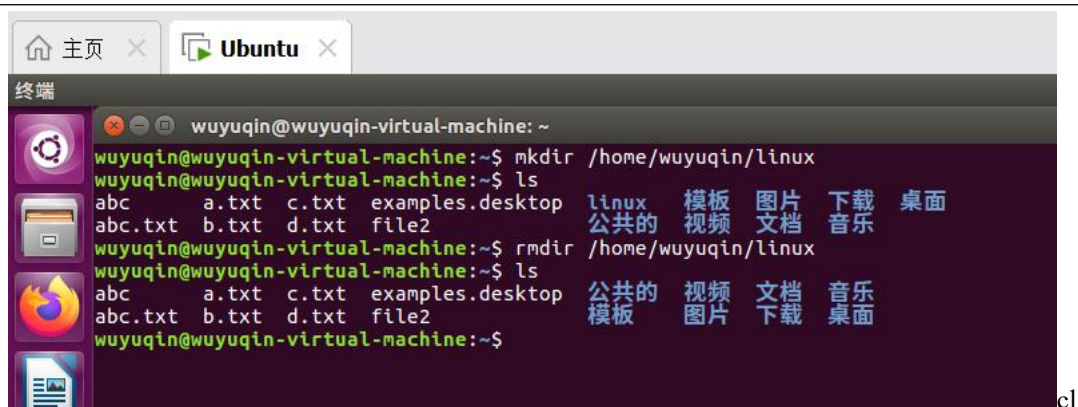
1. 使用命令切换到/etc 目录，并显示当前工作目录路径

```
root@wuyuqin-virtual-machine: /etc
root@wuyuqin-virtual-machine:/home/wuyuqin# cd /etc
root@wuyuqin-virtual-machine:/etc# pwd
/etc
root@wuyuqin-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。

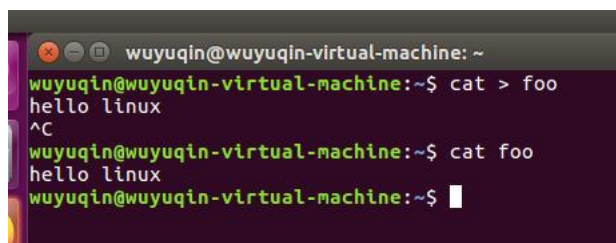
```
wuyuqin@wuyuqin-virtual-machine: ~
wuyuqin@wuyuqin-virtual-machine:~$ ls -al
总用量 144
drwxr-xr-x 19 wuyuqin wuyuqin 4096 5月 22 14:56 .
drwxr-xr-x 3 root root 4096 3月 4 08:18 ..
-rw-rw-r-- 1 wuyuqin wuyuqin 0 5月 22 14:53 abc
-rw-rw-r-- 1 wuyuqin wuyuqin 12 5月 22 14:58 abc.txt
-rw-rw-r-- 1 wuyuqin wuyuqin 15 4月 15 01:29 a.txt
-rw-r--r-- 1 wuyuqin wuyuqin 53 5月 22 14:41 .bash_history
-rw-r--r-- 1 wuyuqin wuyuqin 220 3月 4 08:18 .bash_logout
-rw-r--r-- 1 wuyuqin wuyuqin 3771 3月 4 08:18 .bashrc
-rw-rw-r-- 1 wuyuqin wuyuqin 18 4月 14 17:25 b.txt
drwx----- 14 wuyuqin wuyuqin 4096 4月 14 17:32 .cache
drwx----- 18 wuyuqin wuyuqin 4096 3月 4 09:39 .config
-rw-rw-r-- 1 wuyuqin wuyuqin 30 4月 14 17:26 c.txt
drwx----- 3 wuyuqin wuyuqin 4096 3月 4 08:37 .dbus
-rw-r--r-- 1 wuyuqin wuyuqin 25 3月 4 08:37 .dmrc
-rw-rw-r-- 1 wuyuqin wuyuqin 24 4月 14 22:24 d.txt
-rw-r--r-- 1 wuyuqin wuyuqin 8980 3月 4 08:18 examples.desktop
-rw-rw-r-- 1 wuyuqin wuyuqin 0 4月 14 17:17 file2
drwx----- 3 wuyuqin wuyuqin 4096 3月 18 08:15 .gconf
drwx----- 3 wuyuqin wuyuqin 4096 3月 13 10:25 .gnome2
drwx----- 3 wuyuqin wuyuqin 4096 5月 22 03:03 .gnupg
-rw-r--r-- 1 wuyuqin wuyuqin 3088 5月 22 03:03 .ICEauthority
drwx----- 3 wuyuqin wuyuqin 4096 3月 4 08:37 .local
drwx----- 5 wuyuqin wuyuqin 4096 4月 14 17:32 .mozilla
drwx----- 2 wuyuqin wuyuqin 4096 3月 4 08:37 .presage
-rw-r--r-- 1 wuyuqin wuyuqin 655 3月 4 08:18 .profile
-rw-r--r-- 1 wuyuqin wuyuqin 0 4月 3 10:14 .sudo_as_admin_successful
-rw-r--r-- 1 wuyuqin wuyuqin 68 5月 22 03:03 .Xauthority
-rw-r--r-- 1 wuyuqin wuyuqin 84 5月 22 03:03 .xsession-errors
-rw-r--r-- 1 wuyuqin wuyuqin 84 4月 14 17:07 .xsession-errors.old
drwxr-xr-x 2 wuyuqin wuyuqin 4096 3月 4 08:37 公共的
drwxr-xr-x 2 wuyuqin wuyuqin 4096 3月 4 08:37 模板
drwxr-xr-x 2 wuyuqin wuyuqin 4096 3月 4 08:37 视频
```

3. 使用命令创建目录/home/wuyuqin/linux，然后删除该目录。



A terminal window titled 'Ubuntu' showing a user named 'wuyuqin' in a virtual machine. The user runs the following commands: `mkdir /home/wuyuqin/linux`, `ls`, `rm -r /home/wuyuqin/linux`, and `ls` again. The output of the `ls` commands shows a directory listing with various files and folders, including 'linux', '模板', '图片', '下载', '桌面', '公共的', '视频', '文档', and '音乐'.

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



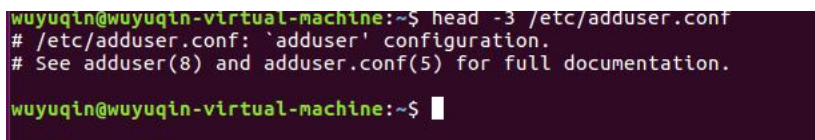
A terminal window showing the user creating a file `foo` using `cat > foo` and entering the text `hello linux`. The user then runs `cat foo` to view the file's content, which outputs `hello linux`.

5、使用命令创建目录 `/home/wuyuqin/foo.bak`，然后将 `/home/wuyuqin/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。



A terminal window showing the user creating a directory `foo.bak` using `mkdir /home/wuyuqin/foo.bak`, copying the file `foo` to it using `cp foo /home/wuyuqin/foo.bak`, and then deleting the directory and its contents using `rm -r /home/wuyuqin/foo.bak`. The user then runs `ls` to check the directory listing, which shows the directory has been removed. The error message `ls: 无法访问 '/home/wuyuqin/foo.bak': 没有那个文件或目录` is displayed.

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。



A terminal window showing the user running `head -3 /etc/adduser.conf` to view the first three lines of the file. The output shows the first three lines of the configuration file.

```
wuyuqin@wuyuqin-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
wuyuqin@wuyuqin-virtual-machine:~$
```

7. 分屏查看文件/etc/adduser.conf 的内容。

```
# /etc/adduser.conf: 'adduser' configuration.
# See adduser(8) and adduser.conf(5) for full documentation.

# The DSHELL variable specifies the default login shell on your
# system.
DSHELL=/bin/bash

# The DHOME variable specifies the directory containing users' home
# directories.
DHOME=/home

# If GROUPHOMES is "yes", then the home directories will be created as
# /home/groupname/user.
GROUPHOMES=no

# If LETTERHOMES is "yes", then the created home directories will have
# an extra directory - the first letter of the user name. For example:
# /home/u/user.
LETTERHOMES=no

# The SKEL variable specifies the directory containing "skeletal" user
# files; in other words, files such as a sample .profile that will be
# copied to the new user's home directory when it is created.
SKEL=/etc/skel

# FIRST_SYSTEM_[GU]ID to LAST_SYSTEM_[GU]ID inclusive is the range for UIDs
# for dynamically allocated administrative and system accounts/groups.
# Please note that system software, such as the users allocated by the base-pa
# 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
--更多--(41%)
```

8、使用命令 cat 用输出重定向在 /home/wuyuqin 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


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

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

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

(2) 按公司人数排序

```
wuyuqin@wuyuqin-virtual-machine:~$ sort -n -t' ' -k 2 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

```
wuyuqin@wuyuqin-virtual-machine:~$ 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

```
wuyuqin@wuyuqin-virtual-machine:~$ sort -t' ' -k3nr -k2n facebook.txt  
baidu 100 5000  
google 110 5000  
sohu 100 4500  
guge 50 3000
```

(5) 从公司英文名称的第2个字母开始进行排序。

```
wuyuqin@wuyuqin-virtual-machine:~$ 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

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		实验日期	2023 年 5 月 19 日

学 号	2021223121	姓 名	吴玉琴
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、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 baz，在 baz 下新建文件 qux，内容随意写几行。

```
wuyuqin@wuyuqin-virtual-machine:~$ mkdir baz
wuyuqin@wuyuqin-virtual-machine:~$ cd baz/
wuyuqin@wuyuqin-virtual-machine:~/baz$ cat > qux
haihaizaimazaina
ruguonilaipewoguoshengridehua
wohuixuyuanzhunishentijiankang
^C
Firefox 网络浏览器 iayiniannihuilaipeiwo
^C
```

- (2) 在用户主目录下查找文件 newfile，并显示该文件位置信息。

```
wuyuqin@wuyuqin-virtual-machine:~/baz$ find ~ -name qux
/home/wuyuqin/baz/qux
```

- (3) 统计 qux 文件中所包含的行数、字数和字节数。

```
wuyuqin@wuyuqin-virtual-machine:~/baz$ wc qux
6 6 140 qux
```

- (4) 在用户主目录下查找文件 qux，并删除该文件。

```
wuyuqin@wuyuqin-virtual-machine:~$ find -name qux -exec rm -rf {} \;
```

- (5) 查看文件夹 baz 内容，看一下是否删除了文件 qux。

```
wuyuqin@wuyuqin-virtual-machine:~/baz$ ls
wuyuqin@wuyuqin-virtual-machine:~/baz$
```

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。

```
wuyuqin@wuyuqin-virtual-machine:~$ mkdir path1
```

```
wuyuqin@wuyuqin-virtual-machine:~$ cd path1/
wuyuqin@wuyuqin-virtual-machine:~/path1$ cat > file1
11
222
33333
^C
wuyuqin@wuyuqin-virtual-machine:~/path1$ cat >file2
83838
jdi j
kk
^C
wuyuqin@wuyuqin-virtual-machine:~/path1$
```

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

```
wuyuqin@wuyuqin-virtual-machine:~$ mkdir path2
wuyuqin@wuyuqin-virtual-machine:~$ cd path2/
wuyuqin@wuyuqin-virtual-machine:~/path2$ cat > file3
ychbcenx
4ju48fh
33
^C
wuyuqin@wuyuqin-virtual-machine:~/path2$
```

- (3) 在用户主目录下新建文件 file4。

```
wuyuqin@wuyuqin-virtual-machine:~$ cat >file4
3330jj
99
^C
wuyuqin@wuyuqin-virtual-machine:~$
```

- (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。

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

- (5) 查看包 package.tar 的内容。

```
wuyuqin@wuyuqin-virtual-machine:~$ tar -tf package.tar
path1/
path1/file2
path1/file1
file4
```

- (6) 向包 package.tar 里添加文件夹 path2 的内容。

```
wuyuqin@wuyuqin-virtual-machine:~$ tar -rvf package.tar path2
path2/
path2/file3
wuyuqin@wuyuqin-virtual-machine:~$
```

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

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

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

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

3、符号链接内容

(1) 新建文件 foo.txt, 内容为 123。

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

```
zcc@zcc-virtual-machine:~$ ln a.txt b.txt
zcc@zcc-virtual-machine:~$ ls
公共的 音乐 ba01.tar facebook.txt m3 vmware-tools-distrib
模板 桌面 b.txt file1 manpath.config
视频 a1.c def linux000 mmd
图片 a2.c ex locate readme
文档 a3 f3 m1 README.md
下载 a.txt f4 m2 snap
zcc@zcc-virtual-machine:~$
```

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

```
wuyuqin@wuyuqin-virtual-machine:~$ df -i foo.txt bar.txt
文件系统 Inode 已用(I) 可用(I) 已用(I)% 挂载点
/dev/sda1 1248480 233473 1015007 19% /
/dev/sda1 1248480 233473 1015007 19% /
```

(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。

```
wuyuqin@wuyuqin-virtual-machine:~$ cat >> bar.txt
abc^C
wuyuqin@wuyuqin-virtual-machine:~$ comm foo.txt bar.txt
123
wuyuqin@wuyuqin-virtual-machine:~$ diff foo.txt bar.txt
wuyuqin@wuyuqin-virtual-machine:~$
```

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

```
wuyuqin@wuyuqin-virtual-machine:~$ rm foo.txt
wuyuqin@wuyuqin-virtual-machine:~$ df -i bar.txt
文件系统 Inode 已用(I) 可用(I) 已用(I)% 挂载点
/dev/sda1 1248480 233473 1015007 19% /
```

(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。

```
wuyuqin@wuyuqin-virtual-machine:~$ ln -s bar.txt baz.txt
wuyuqin@wuyuqin-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 233476 1015004    19% /
/dev/sda1      1248480 233476 1015004    19% /
wuyuqin@wuyuqin-virtual-machine:~$ cat bar.txt
123
wuyuqin@wuyuqin-virtual-machine:~$ cat baz.txt
123
```

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

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

4、权限管理

(1) 新建文件 qux.txt

```
wuyuqin@wuyuqin-virtual-machine:~$ cat > qux.txt
33
^?^C
```

(2) 为文件 qux.txt 增加执行权限（所有用户都可以执行）。

```
zzc@zzc-virtual-machine:~$ chmod a+w test/
zzc@zzc-virtual-machine:~$ cd test/
zzc@zzc-virtual-machine:~/test$ cat >tt.txt
12344556
1234^Z
[9]+ 已停止                  cat > tt.txt
zzc@zzc-virtual-machine:~/test$ cat tt.txt
1
显示应用程序
zzc@zzc-virtual-machine:~/test$
```

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		实验日期	2023 年 5 月 17 日

学 号	2021223121	姓 名	吴玉琴
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、vim 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 workspace1

```
wuyuqin@wuyuqin-virtual-machine:~$ mkdir workspace1
```

- (2) 进入目录 workspace1

```
wuyuqin@wuyuqin-virtual-machine:~$ cd workspace1/
```

- (3) 在 workspace1 下用 vim 编辑器新建一个 c 语言程序文件，文件名为 test.c

test.c 文件内容为：

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

```
wuyuqin@wuyuqin-virtual-machine:~$ 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。

```
wuyuqin@wuyuqin-virtual-machine:~$ mkdir workspace2  
wuyuqin@wuyuqin-virtual-machine:~$ ls  
abc      baz.txt      facebook.txt  package.tar  test.c      视频  桌面  
abc.txt  b.txt       file2        path1       workspace1  图片  
a.txt    c.txt       file4        path2       workspace2  文档  
baz      d.txt       foo          path3       公共的     下载  
baz.txt  examples.desktop m3          qux.txt     模板       音乐
```

- (2) 进入 workspace2 目录。

```
wuyuqin@wuyuqin-virtual-machine:~$ cd workspace2
```

- (3) 将文件/etc/gai.conf 复制到当前目录下的新建文件 gar.conf 中

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

(4) 使用 vi 编辑当前目录下的 gai.conf。

```
wuyuqin@wuyuqin-virtual-machine: ~/workspace2
## Configuration for getaddrinfo(3).
#
# So far only configuration for the destination address sorting is needed.
# RFC 3484 governs the sorting. But the RFC also says that system
# administrators should be able to overwrite the defaults. This can be
# achieved here.
#
# All lines have an initial identifier specifying the option followed by
# up to two values. Information specified in this file replaces the
# default information. Complete absence of data of one kind causes the
# appropriate default information to be used. The supported commands include:
#
# reload <yes|no>
#   If set to yes, each getaddrinfo(3) call will check whether this file
#   changed and if necessary reload. This option should not really be
#   used. There are possible runtime problems. The default is no.
#
# label <mask> <value>
#   Add another rule to the RFC 3484 label table. See section 2.1 in
#   RFC 3484. The default is:
#
#label ::1/128      0
#label ::/0         1
"gai.conf" 65 lines, 2584 characters
```

(6) 将光标移到第 18 行。

gg+17+ ↓

```
# Configuration for getaddrinfo(3).
#
# So far only configuration for the destination address sorting is needed.
# RFC 3484 governs the sorting. But the RFC also says that system
# administrators should be able to overwrite the defaults. This can be
# achieved here.
#
# All lines have an initial identifier specifying the option followed by
# up to two values. Information specified in this file replaces the
# default information. Complete absence of data of one kind causes the
# appropriate default information to be used. The supported commands include:
#
# reload <yes|no>
#   If set to yes, each getaddrinfo(3) call will check whether this file
#   changed and if necessary reload. This option should not really be
#   used. There are possible runtime problems. The default is no.
#
# label <mask> <value>
#   Add another rule to the RFC 3484 label table. See section 2.1 in
#   RFC 3484. The default is:
#
#label ::1/128      0
#label ::/0         1
```

(7) 复制该行内容。

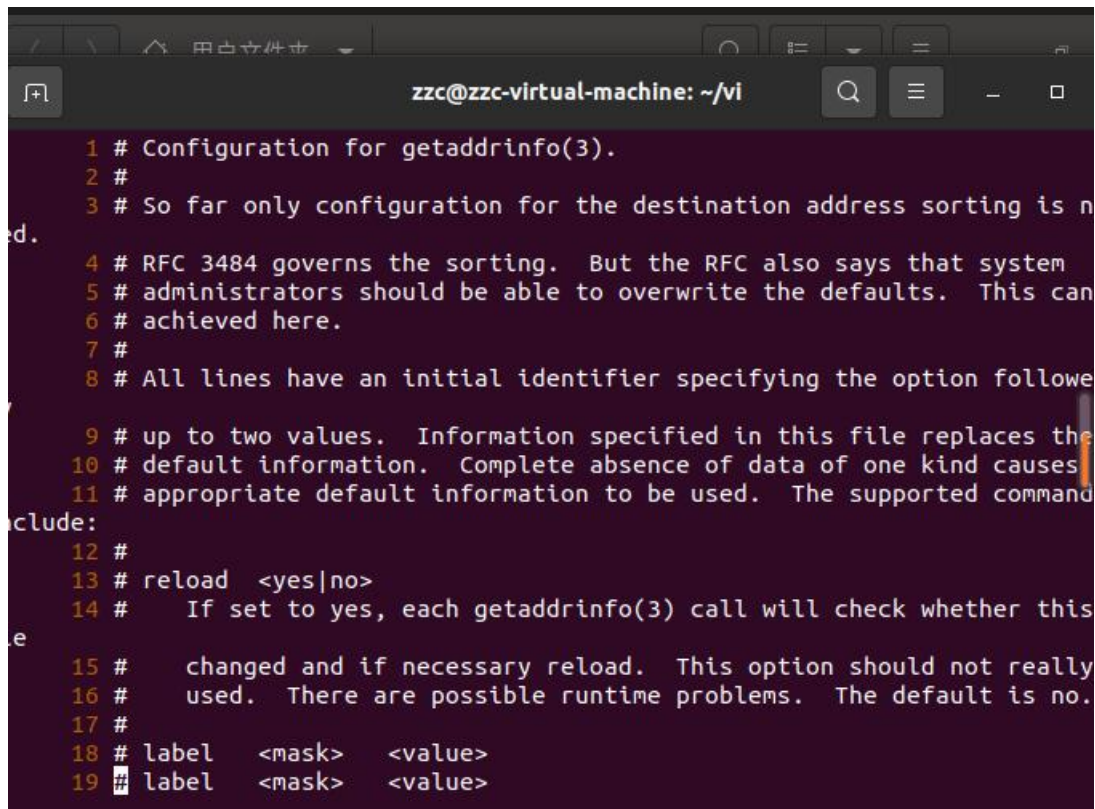
yy

(8) 将光标移到最后一行行首。

G

(9) 粘贴复制行的内容。

p



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

(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 vi world!”。

```

40 # (at least for the foreseeable future) NATed. We also treat Teredo
41 # tunnels special.
42 #
43 # precedence <mask> <value>
44 # Add another rule to the RFC 3484 precedence table. See section 2.
45 # and 10.3 in RFC 3484. The default is:
46 #
47 #precedence ::1/128 50
48 #precedence ::/0 40
49 #precedence 2002::/16 30
50 #precedence ::/96 20
51 #precedence ::ffff:0:0/96 10
52 #
53 # For sites which prefer IPv4 connections change the last line to
54 #
55 #precedence ::ffff:0:0/96 100
56 #
57 #
58 # scopev4 <mask> <value>
59 # Add another rule to the RFC 6724 scope table for IPv4 addresses.
60 # By default the scope IDs described in section 3.2 in RFC 6724 are
61 # used. Changing these defaults should hardly ever be necessary.
62 # The defaults are equivalent to:
63 #
64 #scopev4 ::ffff:169.254.0.0/112 2
65 #scopev4 ::ffff:127.0.0.0/104 2
66 Hello ,This is vi world!

```

(13) 删除字符串 “this “。

光标指向 this 首字母+dw

```

131 #NOCACHE
132 Hello, this is vi world!
1 行发生改变: before #11 27 秒之前

```

(14) 强制退出vi，不存盘。

:q!

```

: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		实验日期	2023 年 5 月 18 日

学 号	2021223121	姓 名	吴玉琴
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、建立一个用户名为 foo，描述信息为 foo，登录 shell 为 /bin/sh，登录主目录为 /home/foo 的用户，并设置口令为 123456。

```
root@wuyuqin-virtual-machine:/home/wuyuqin# useradd foo -s /bin/sh -b /home/foo -c foo -p 123456
```

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

```
root@wuyuqin-virtual-machine:/home/wuyuqin# usermod foo -u 2000 -s /bin/sch
root@wuyuqin-virtual-machine:/home/wuyuqin# su foo
无法执行 /bin/sch: 没有那个文件或目录
root@wuyuqin-virtual-machine:/home/wuyuqin# usermod foo -u 2000 -s /bin/sh
root@wuyuqin-virtual-machine:/home/wuyuqin# su foo
```

- 3、使用命令从用户 foo 切换到 root。

```
$ su
显示应用程序
root@wuyuqin-virtual-machine:/home/zzc#
```

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

```
wuyuqin@wuyuqin-virtual-machine:~$ sudo userdel foo -r
[sudo] wuyuqin 的密码:
对不起，请重试。
[sudo] wuyuqin 的密码:
userdel: foo 邮件池 (/var/mail/foo) 未找到
userdel: 未找到 foo 的主目录"/home/foo/foo"
wuyuqin@wuyuqin-virtual-machine:~$
```

- 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
```

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

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

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

```
root@wuyuqin-virtual-machine:/home/wuyuqin# groupadd group1 -g 3000
```

7、在用户组 group1 中添加两个之前批量创建的用户。

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

```
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# usermod -g group1 user1
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:

#include_dir /etc/sudoers.d

~
~
```

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

```
root:$6$dMBXhYwd$5nS0/oZN1xfwX0jpe4nHLboHAWbXZyHCB22FBmuVs/Isi9KbahcPSrLeob/j1k641erdFuACjWnZWtxq4E1j0r.
:18364:0:99999:7:::
daemon:*:18113:0:99999:7:::
bin:*:18113:0:99999:7:::
sys:*:18113:0:99999:7:::
sync:*:18113:0:99999:7:::
games:*:18113:0:99999:7:::
man:*:18113:0:99999:7:::
lp:*:18113:0:99999:7:::
mail:*:18113:0:99999:7:::
news:*:18113:0:99999:7:::
uucp:*:18113:0:99999:7:::
proxy:*:18113:0:99999:7:::
www-data:*:18113:0:99999:7:::
backup:*:18113:0:99999:7:::
list:*:18113:0:99999:7:::
irc:*:18113:0:99999:7:::
gnats:*:18113:0:99999:7:::
nobody:*:18113:0:99999:7:::
systemd-network:*:18113:0:99999:7:::
systemd-resolve:*:18113:0:99999:7:::
syslog:*:18113:0:99999:7:::
messagebus:*:18113:0:99999:7:::
_apt:*:18113:0:99999:7:::
uidd:*:18113:0:99999:7:::
avahi-autoipd:*:18113:0:99999:7:::
usbmux:*:18113:0:99999:7:::
dnsmasq:*:18113:0:99999:7:::
rtkit:*:18113:0:99999:7:::
cups-pk-helper:*:18113:0:99999:7:::
speech-dispatcher:1:18113:0:99999:7:::
whoopsie:*:18113:0:99999:7:::
kernoops:*:18113:0:99999:7:::
saned:*:18113:0:99999:7:::
pulse:*:18113:0:99999:7:::
avahi:*:18113:0:99999:7:::
colord:*:18113:0:99999:7:::
hplip:*:18113:0:99999:7:::
geoclue:*:18113:0:99999:7:::
gnome-initial-setup:*:18113:0:99999:7:::
"/etc/shadow" 45L, 1662C
```

1,1 顶端

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 ~]# groupadd -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

文件： /etc/passwd 文
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		实验日期	2023 年 5 月 19 日
学 号	2021223121	姓 名	吴玉琴
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 AK 的值为 200，并将其显示在屏幕上。（终端上执行）

```
root@wuyuqin-virtual-machine:/home/wuyuqin# foo=200
root@wuyuqin-virtual-machine:/home/wuyuqin# echo $foo
200
```

- 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@wuyuqin-virtual-machine:/home/wuyuqin# bar=100
root@wuyuqin-virtual-machine:/home/wuyuqin# test $bar -gt 150 && echo 'yes'
```

- 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）

```
root@wuyuqin-virtual-machine:/home/wuyuqin# vi test.sh
root@wuyuqin-virtual-machine:/home/wuyuqin# sh test.sh
```

```
#!/bin/bash
#name KPL
#time 2020.6.20
#function test
#version 1.0

echo ${hostname}
echo ${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。

```
kpl@hadoop100:~/linuxlearn$ sh shuixian.sh 153
Total paramter are: 1
The num is: 153
153 is shuixianhua num!
kpl@hadoop100:~/linuxlearn$
```

```
#!/bin/bash
#function 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。

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

```
test $# -eq 0 && echo "You don't give one paramter 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
```

- 5、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”。

```
zzc@zzc-virtual-machine:~$ sh chazhao.sh
```



```
#!/bin/bash
#function 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。

```
root@wuyuqin-virtual-machine:/home/wuyuqin# sh grade.sh 91 12 60
```

```
The parmater num are: 3
```

```
A
```

```
E
```

```
D
```

```
kpl@hadoop100:~/linuxlearn$
```

```
#!/bin/bash
#function 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 parmater 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

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

```

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		实验日期	2023 年 5 月 19 日

学 号	2021223121	姓 名	吴玉琴
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

(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）

```
root@wuyuqin-virtual-machine:/home/wuyuqin# sh move.shi
```

```
Please input a file name: .
./chazhao.sh
./grade.sh
./shuixian.sh
./test.sh
./chazhao.sh
./grade.sh
./shuixian.sh
```

```
root@wuyuqin-virtual-machine:/home/wuyuqin# 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。

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

```
#!/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"
~
```

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

```
zxc@zxc-virtual-machine:~$ sh pingfang.sh
385
zxc@zxc-virtual-machine:~$
```

```
#!/bin/bash
#function 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

```
#!/bin/bash
#function 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。

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
        fi
    done
    echo $j
done
```

```
385
zcc@zcc-virtual-machine:~$ sh findinfo.sh
findinfo.sh: 2: for i in a b c d: not found
findinfo.sh: 3: do : not found
findinfo.sh: 5: for j in 1: not found
findinfo.sh: 6: do: not found
findinfo.sh: 7: if [ -eq 5: not found
findinfo.sh: 7: then : not found
findinfo.sh: 8: break或continue: not found
findinfo.sh: 9: fi: not found
findinfo.sh: 10: echo: not found
findinfo.sh: 11: done: not found

findinfo.sh: 13: done: not found
zcc@zcc-virtual-machine:~$
```

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

```
zcc@zcc-virtual-machine:~$ sh findinfo.sh
```

```
a
```

```
#!/bin/bash
#function 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。

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
```

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

```
#!/bin/bash
#function 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。

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
        fi
    done
    echo $j
done
```

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 \$i

```
#!/bin/bash
#function 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。

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

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 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。

Shell 支持列表 for 循环使用略写的计数方式，1~5 的范围用 {1...5} 表示（大括号不能去掉，否则会当作一个字符串处理）。

Shell 中还支持按规定的步数进行跳跃的方式实现列表 for 循环，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
```

也可一使用 `for file in` ，通配符产生文件名扩展，匹配当前目录下的所有文件。
`for` 通过命令行来传递脚本中 `for` 循环列表参数

```
#!/bin/bash

echo "number of arguments is $#"
```

echo "What you input is: "

```
for argument in "$@"
do
    echo "$argument"
done
```

`$#`表示参数的个数，`@` 表示参数列表而 `@`表示参数列表而 `@`表示参数列表而`*`则把所有的参数当作一个字符串显示。

（3）不带列表 `for` 循环

由用户制定参数和参数的个数，与上述的 `for` 循环列表参数功能相同。

```
#!/bin/bash

echo "number of arguments is $#"
```

echo "What you input is: "

```
for argument
do
    echo "$argument"
done
```

比上述代码少了 `$@`参数列表，`$*`参数字符串。

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		实验日期	2023 年 5 月 20 日
学 号	2021223121	姓 名	吴玉琴
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，定义一个函数返回两个数的和。

```
zzc@zzc-virtual-machine:~$ sh qiuhe3.sh 1 2
0
zzc@zzc-virtual-machine:~$ vi qiuhe.sh
```

```
#!/bin/bash
#function 求和

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。

```
jiecheng.sh: 7: Syntax error: do: unexpected (exp
zzc@zzc-virtual-machine:~$ sh jiecheng.sh 5 120
120
zzc@zzc-virtual-machine:~$
```

```
kpt@hadoop100:~/linuxlearn$ vi jiecheng.sh

#!/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"
```



```
        third
    }
    echo "this is the first"
    second
}

echo "start..."
first
```

```
func_test.sh: 2: Syntax error: ( unexpected
zzc@zzc-virtual-machine:~$ sh func_test.sh
```

```
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

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() {
```

<pre>statements [return value] }</pre> <p>如果写了 <code>function</code> 关键字，也可以省略函数名后面的小括号：</p> <pre>function name { statements [return value] }</pre> <p>函数调用</p> <p>调用 <code>Shell</code> 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：</p> <p><code>name</code></p> <p>如果传递参数，那么多个参数之间以空格分隔：</p> <p><code>name param1 param2 param3</code></p> <p>不管是哪种形式，函数名字后面都不需要带括号。</p> <p>和其它编程语言不同的是，<code>Shell</code> 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。</p> <p><code>Shell</code> 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。</p>
<p>五、指导教师意见</p> <p style="text-align: right;">指导教师签字： 卢洋</p>

实验报告

实验名称	实验八 sed 和 awk		
实验教室		实验日期	2023 年 5 月 21 日
学 号	2021223121	姓 名	吴玉琴
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

```
root@wuyuqin-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.
```

- (2) 显示包含 music 文字的行内容及行号

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

- (3) 在第 4 行后面追加文件“hello world! ”

```
root@wuyuqin-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 at 23:10.  
The local nurse Miss P.Neave was in attendance.  
hello world
```

- (4) 将文本“The”修改为“Ok”

```
root@wuyuqin-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.
```

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

```
root@wuyuqin-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.
```

- (6) 删除第 2 行内容。

```
root@wuyuqin-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.
```

(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。

```
root@wuyuqin-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

```
root@wuyuqin-virtual-machine:~# cat aaa.txt | awk '{FS=":"}{print $2}'  
:  
five  
root@wuyuqin-virtual-machine:~# ^C  
root@wuyuqin-virtual-machine:~# cat aaa.txt | awk '{FS=" "}{print $2}'  
:  
:  
:  
root@wuyuqin-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 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: foo.txt 内容为:

2 4 3

15 46 79

则输出为:

2

4

46

```
root@wuyuqin-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}}'  
2446root@wuyuqin-virtual-machine:~#
```

4、已知脚本的内容如下, 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
```

```
root@wuyuqin-virtual-machine:~# cat info.txt
```

```
nux - Sysadmin  
Database - Oracle,MySQL etc.  
Security - Firewall,Network, Online Security etc.  
Cool - Websites
```

```
root@wuyuqin-virtual-machine:~# touch t.sh
```

```
root@wuyuqin-virtual-machine:~# cat t.txt
```

```
#!/bin/bash  
read -p "enter search pattern: " pattern  
awk "/$pattern/" '{ nmatches++; print } END { print nmatches "found." }' info.t  
xt
```

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

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