

# 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 4 日
学 号	2021212614	姓 名	温晖
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

```
(dolphln@dolphln)-[~]  
$ cd /etc  
  
(dolphln@dolphln)-[/etc]  
$ pwd  
/etc
```

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

```
(dolphln@dolphln)-[/etc]  
$ ls -la /home/dolphln  
total 40  
drwxr-xr-x 1 dolphln dolphln 4096 Apr 26 12:58 .  
drwxr-xr-x 1 root root 4096 Sep 25 2021 ..  
-rw-r--r-- 1 dolphln dolphln 13 Mar 9 13:57 abc  
-rw----- 1 dolphln dolphln 5017 Apr 27 15:36 .bash_history  
-rw-r--r-- 1 dolphln dolphln 220 Sep 25 2021 .bash_logout  
-rw-r--r-- 1 dolphln dolphln 5349 Sep 25 2021 .bashrc  
-rw-r--r-- 1 dolphln dolphln 3526 Sep 25 2021 .bashrc.original  
drwxr-xr-x 1 dolphln dolphln 4096 Nov 19 16:16 .cache  
drwxr-xr-x 1 dolphln dolphln 4096 Nov 19 16:16 .config  
-rw-r--r-- 1 dolphln dolphln 58 Mar 9 14:23 facebook.txt  
-rw-r--r-- 1 dolphln dolphln 25 Nov 19 16:34 .gitconfig  
drwxr-xr-x 1 dolphln dolphln 4096 Nov 19 16:17 go  
drwxr-xr-x 1 dolphln dolphln 4096 Apr 26 12:50 .kaggle  
drwx----- 1 dolphln dolphln 4096 Feb 12 17:07 .local  
-rw-r--r-- 1 dolphln dolphln 807 Sep 25 2021 .profile  
-rw----- 1 dolphln dolphln 3589 Apr 26 12:58 .viminfo  
-rw-r--r-- 1 dolphln dolphln 10583 Sep 25 2021 .zshrc
```

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

```

(dolphin@dolphin)-[~]
$ mkdir linux

(dolphin@dolphin)-[~]
$ ls -l
total 0
-rw-r--r-- 1 dolphin dolphin 13 Mar 9 13:57 abc
-rw-r--r-- 1 dolphin dolphin 58 Mar 9 14:23 facebook.txt
drwxr-xr-x 1 dolphin dolphin 4096 Nov 19 16:17 go
drwxr-xr-x 1 dolphin dolphin 4096 Apr 27 18:20 linux

(dolphin@dolphin)-[~]
$ rm -r linux

(dolphin@dolphin)-[~]
$ ls -l
total 0
-rw-r--r-- 1 dolphin dolphin 13 Mar 9 13:57 abc
-rw-r--r-- 1 dolphin dolphin 58 Mar 9 14:23 facebook.txt
drwxr-xr-x 1 dolphin dolphin 4096 Nov 19 16:17 go

(dolphin@dolphin)-[~]
$ |

```

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

```

(dolphin@dolphin)-[~]
$ cat > foo
Hello, Linux!
^C

(dolphin@dolphin)-[~]
$ cat foo
Hello, Linux!

```

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

```
(dolphln@Dolphln)-[~]  
$ mkdir foo.bak  
  
(dolphln@Dolphln)-[~]  
$ cp foo foo.bak/  
  
(dolphln@Dolphln)-[~]  
$ ls foo.bak/  
foo  
  
(dolphln@Dolphln)-[~]  
$ rm -r foo.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。

```
(dolphln@Dolphln)-[~]  
$ cat /etc/adduser.conf | head -n 3  
# /etc/adduser.conf: 'adduser' configuration.  
# See adduser(8) and adduser.conf(5) for full documentation.  
  
(dolphln@Dolphln)-[~]  
$ cat /etc/adduser.conf | tail -n 5  
#ADD_EXTRA_GROUPS=1  
  
# check user and group names also against this regular expression.  
#NAME_REGEX="^[a-z][-a-z0-9_]*\\$"
```

7、分屏查看文件/etc/adduser.conf 的内容。

```
Windows PowerShell x dolphin@Dolphin: ~
# /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 GROUPTHOMES is "yes", then the home directories will be created as
# /home/groupname/user.
GROUPTHOMES=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-passwd
# package, may assume that UIDs less than 100 are unallocated.
FIRST_SYSTEM_UID=100
LAST_SYSTEM_UID=999

FIRST_SYSTEM_GID=100
LAST_SYSTEM_GID=999

# FIRST_[GU]ID to LAST_[GU]ID inclusive is the range of UIDs of dynamically
# allocated user accounts/groups.
FIRST_UID=1000
LAST_UID=59999
--More--(44%)
```

When I see your monsters  
当我看到隐藏于你心底的恶魔

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

```
(dolphin@dolphin)~$ cat > facebook.txt
google 110 5000
baidu 100 5000
guge 50 3000
sohu 100 4500
^C

(dolphin@dolphin)~$ cat facebook.txt
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。

```
(dolphins@Dolphins)~  
$ sort facebook.txt  
baidu 100 5000  
google 110 5000  
guge 50 3000  
sohu 100 4500  
  
(dolphins@Dolphins)~  
$ sort -n -k 2 -t "  
> ^C  
  
(dolphins@Dolphins)~  
$ sort -n -k 2 -t " " facebook.txt  
guge 50 3000  
baidu 100 5000  
sohu 100 4500  
google 110 5000  
  
(dolphins@Dolphins)~  
$ sort -n -k 2 -t " " -k 3 facebook.txt  
guge 50 3000  
sohu 100 4500  
baidu 100 5000  
google 110 5000
```

```
(dolphln@Dolphln)-[~]  
$ sort -n -t " " -k 3r -k 2 facebook.txt  
baidu 100 5000  
google 110 5000  
sohu 100 4500  
guge 50 3000  
  
(dolphln@Dolphln)-[~]  
$ sort -t " " -k 1.2 facebook.txt  
baidu 100 5000  
sohu 100 4500  
google 110 5000  
guge 50 3000
```

#### 四、 实验过程分析与讨论

上网搜索之后理解了 sort 指令的排序规则可以进行多重指定，按照先指定的规则优先的原则进行排序即可。



## 五、指导教师意见

指导教师签字：卢洋

# 实验报告

实验名称	实验二 Linux 常用命令（二）		
实验教室	丹青 922	实验日期	2023 年 3 月 11 日
学 号	2021212614	姓 名	温晖
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

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

```
(dolphln@Dolphln)~  
$ cd baz  
  
(dolphln@Dolphln)~/baz  
$ cat >qux  
dakfjaskjda  
asldfajlks  
afdjhlsanfl  
asfdjhads  
sadfkjafdlk  
sakfdald  
^C
```

(2) 在用户目录下查找文件 qux，并显示该文件位置信息。

```
(dolphln@Dolphln)~  
$ find / -name "qux"  
find: '/etc/ssl/private': Permission denied  
/home/dolphln/baz/qux  
find: '/mnt/c/$Recycle.Bin/S-1-5-18': Permission denied  
find: '/mnt/c/$Recycle.Bin/S-1-5-21-1507205835-3049290107-1512793131-1000': Permission denied  
find: '/mnt/c/Config.Msi': Permission denied  
find: '/mnt/c/Intel/IntelOptaneData': Permission denied  
find: '/mnt/c/OneDriveTemp/S-1-5-21-3741872280-3711046503-353168718-1001': Permission denied  
^C
```

(3) 统计 qux 文件中所包含的行数、字数和字节数。

```
(dolphln@Dolphln)~  
$ cat ./baz/qux | wc  
6      6      67
```

(4) 在用户主目录下查找文件 qux，并删除该文件。

```

(dolphin@dolphin)~]
$ find / -name "qux"
find: '/etc/ssl/private': Permission denied
/home/dolphin/baz/qux
find: '/mnt/c/$Recycle.Bin/S-1-5-18': Permission denied
find: '/mnt/c/$Recycle.Bin/S-1-5-21-1507205835-3049290107-1512793131-1000': Permission denied
find: '/mnt/c/Config.Msi': Permission denied
find: '/mnt/c/Intel/IntelOptaneData': Permission denied
find: '/mnt/c/OneDriveTemp/S-1-5-21-3741872280-3711046503-353168718-1001': Permission denied
^C
(dolphin@dolphin)~]
$ rm baz/qux

```

(5) 查看文件夹 baz 内容，看一下是否删除了文件 qux。

```

(dolphin@dolphin)~]
$ ls baz -al
total 0
drwxr-xr-x 1 dolphin dolphin 4096 May  4 17:17 .
drwxr-xr-x 1 dolphin dolphin 4096 May  4 17:14 ..

```

## 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。

```

(dolphin@dolphin)~]
$ mkdir path1

(dolphin@dolphin)~]
$ cd path1

(dolphin@dolphin)~/path1]
$ touch file1 file2

```

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

```

(dolphin@dolphin)~]
$ mkdir path2 && cd path2 && touch file3

(dolphin@dolphin)~/path2]
$ |

```

(3) 在用户主目录下新建文件 file4。

```
(dolphln@Dolphln)-[~]  
$ touch file4  
  
(dolphln@Dolphln)-[~]  
$
```

(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。

```
(dolphln@Dolphln)-[~]  
$ tar -cvf package.tar path1/ file4  
path1/  
path1/file1  
path1/file2  
file4
```

(5) 查看包 package.tar 的内容。

```
(dolphln@Dolphln)-[~]  
$ tar -tf package.tar  
path1/  
path1/file1  
path1/file2  
file4
```

(6) 向包 package.tar 里添加文件夹 path2 的内容。

```
(dolphln@Dolphln)-[~]  
$ tar -rvf package.tar path2/  
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。

```
(dolphln@Dolphln)-[~]  
$ mkdir path3 && cp package.tar path3/  
  
(dolphln@Dolphln)-[~]  
$ l path3  
package.tar
```

(8) 进入 path3 文件夹，并还原包 package 的内容。

```
(do1phln@Do1phln)-[~]  
$ cd path3  
  
(do1phln@Do1phln)-[~/path3]  
$ tar -xvf package.tar  
path1/  
path1/file1  
path1/file2  
file4  
path2/  
path2/file3  
  
(do1phln@Do1phln)-[~/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。

```
(do1phln@Do1phln)-[~]  
$ cat >foo.txt  
123  
^C
```

(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。

```
(do1phln@Do1phln)-[~]  
$ ln foo.txt bar.txt  
  
(do1phln@Do1phln)-[~]  
$ diff foo.txt bar.txt  
  
(do1phln@Do1phln)-[~]  
$ comm foo.txt bar.txt  
123
```

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

```
(dolphln@Dolphln)-[~]  
$ ls -i foo.txt bar.txt  
28428972647787502 bar.txt 28428972647787502 foo.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。

```
(dolphln@Dolphln)-[~]  
$ vim bar.txt  
  
(dolphln@Dolphln)-[~]  
$ cat bar.txt  
abc  
  
(dolphln@Dolphln)-[~]  
$ diff foo.txt bar.txt  
  
(dolphln@Dolphln)-[~]  
$ 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。

```
(dolphln@Dolphln)-[~]  
$ rm foo.txt  
  
(dolphln@Dolphln)-[~]  
$ ls -i bar.txt  
28428972647787502 bar.txt  
  
(dolphln@Dolphln)-[~]  
$ 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。



```
(dolphln@Dolphln)-[~]  
$ ln -s bar.txt baz.txt  
  
(dolphln@Dolphln)-[~]  
$ ls -li bar.txt baz.txt  
28428972647787502 bar.txt 29554872554628057 baz.txt  
  
(dolphln@Dolphln)-[~]  
$ comm bar.txt baz.txt  
abc
```

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

```
(dolphln@Dolphln)-[~]  
$ rm bar.txt  
  
(dolphln@Dolphln)-[~]  
$ cat baz.txt  
cat: baz.txt: No such file or directory
```

#### 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）。

```
(dolphln@Dolphln)-[~]  
$ touch qux.txt  
  
(dolphln@Dolphln)-[~]  
$ chmod a+x qux.txt  
  
(dolphln@Dolphln)-[~]  
$ ls -l qux.txt  
-rwxr-xr-x 1 dolphln dolphln 0 May  4 17:37 qux.txt
```

## 八、 实验过程分析与讨论

对 linux 中的权限管理还是不太熟悉，在查询 CSDN 之后学会了相关命令，同时还学习了 7 以内的数值如何描述 linux 文件权限，以及如何通过数值形式进行 chmod 指令授权。

## 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 Linux 常用命令（三）		
实验教室	丹青 922	实验日期	2023 年 3 月 18 日
学 号	2021212614	姓 名	温晖
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

掌握 vi 编辑器及 gcc 编译器的使用方法

## 二、 实验环境

(1) 计算机的硬件配置 PC 系列微机。

(2) 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、 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

```
(dolphln@Dolphln)-[~]  
$ mkdir workspace1
```

(2) 进入目录 workspace1 进入目录 vifile

```
(dolphln@Dolphln)-[~]  
$ cd workspace1/  
  
(dolphln@Dolphln)-[~/workspace1]  
$ |
```

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

内容为

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

(4) 保存 test.c 的内容，并退出

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

```
(do1phln@Do1phln) - [~/workspace1]  
$ vim test.c
```

```
#include<stdio.h>  
  
int main()  
{  
    printf("Hello World!\n");  
    return 0;  
}
```

:wq|

```
(dolphln@Dolphln)-[~/workspace1]
$ gcc test.c -o test

(dolphln@Dolphln)-[~/workspace1]
$ test

(dolphln@Dolphln)-[~/workspace1]
$ ./test
Hello World!
```

## 2、vi 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

```
(dolphln@Dolphln)-[~]
$ mkdir workspace2
```

(2) 进入 workspace2 目录

```
(dolphln@Dolphln)-[~]
$ cd workspace2

(dolphln@Dolphln)-[~/workspace2]
$
```

(3) 使用以下命令：

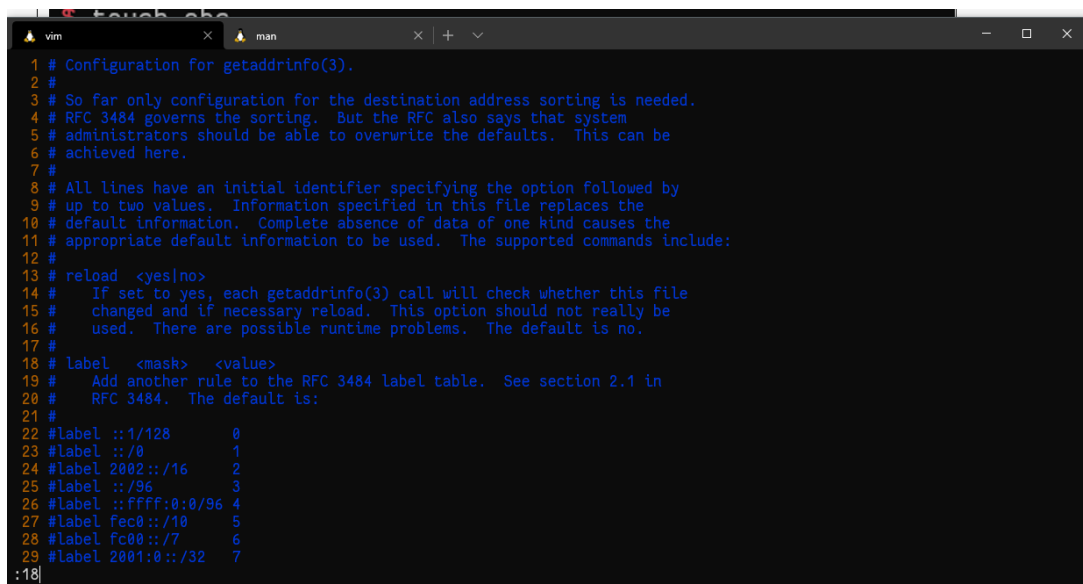
```
cat /etc/gai.conf > ./gai.conf
```

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；

```
(dolphln@Dolphln)-[~/workspace2]
$ cat /etc/gai.conf > ./gai.conf

(dolphln@Dolphln)-[~/workspace2]
$ cat 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 include:
#
# reload <yes|no>
```

- (4) 使用 vim 编辑当前目录下的 gai.conf
- (5) 将光标移到第 18 行。
- (6) 复制该行内容。
- (7) 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。
- (10) 存盘但不退出。
- (11) 将光标移到首行。
- (12) 插入模式下输入 "Hello, this is vim world!"
- (13) 删除字符串 "this "。
- (14) 强制退出 vim ，不存盘



```
1 # Configuration for getaddrinfo(3).
2 #
3 # So far only configuration for the destination address sorting is needed.
4 # RFC 3484 governs the sorting. But the RFC also says that system
5 # administrators should be able to overwrite the defaults. This can be
6 # achieved here.
7 #
8 # All lines have an initial identifier specifying the option followed by
9 # up to two values. Information specified in this file replaces the
10 # default information. Complete absence of data of one kind causes the
11 # appropriate default information to be used. The supported commands include:
12 #
13 # reload <yes|no>
14 #   If set to yes, each getaddrinfo(3) call will check whether this file
15 #   changed and if necessary reload. This option should not really be
16 #   used. There are possible runtime problems. The default is no.
17 #
18 # label <mask> <value>
19 #   Add another rule to the RFC 3484 label table. See section 2.1 in
20 #   RFC 3484. The default is:
21 #
22 #label ::1/128 0
23 #label ::/0 1
24 #label 2002::/16 2
25 #label ::/96 3
26 #label ::ffff:0:0/96 4
27 #label fec0::/10 5
28 #label fc00::/7 6
29 #label 2001:0::/32 7
```

y -> 复制该行内容

^G -> 将光标移到最后一行行首

p -> 粘贴复制行的内容

```
vim man
38 # site-local IPv6 addresses cannot be used while the IPv4 address is
39 # (at least for the foreseeable future) NATed. We also treat Teredo
40 # tunnels special.
41 #
42 # precedence <mask> <value>
43 # Add another rule to the RFC 3484 precedence table. See section 2.1
44 # and 10.3 in RFC 3484. The default is:
45 #
46 #precedence ::1/128 50
47 #precedence ::/0 40
48 #precedence 2002::/16 30
49 #precedence ::/96 20
50 #precedence ::ffff:0:0/96 10
51 #
52 # For sites which prefer IPv4 connections change the last line to
53 #
54 #precedence ::ffff:0:0/96 100
55 #
56 #
57 # scopev4 <mask> <value>
58 # Add another rule to the RFC 6724 scope table for IPv4 addresses.
59 # By default the scope IDs described in section 3.2 in RFC 6724 are
60 # used. Changing these defaults should hardly ever be necessary.
61 # The defaults are equivalent to:
62 #
63 #scopev4 ::ffff:169.254.0.0/112 2
64 #scopev4 ::ffff:127.0.0.0/104 2
65 #scopev4 ::ffff:0.0.0.0/96 14
66 #label <mask> <value>
```

66,1 Bot

:u -> 撤销

```
vim man
38 # site-local IPv6 addresses cannot be used while the IPv4 address is
39 # (at least for the foreseeable future) NATed. We also treat Teredo
40 # tunnels special.
41 #
42 # precedence <mask> <value>
43 # Add another rule to the RFC 3484 precedence table. See section 2.1
44 # and 10.3 in RFC 3484. The default is:
45 #
46 #precedence ::1/128 50
47 #precedence ::/0 40
48 #precedence 2002::/16 30
49 #precedence ::/96 20
50 #precedence ::ffff:0:0/96 10
51 #
52 # For sites which prefer IPv4 connections change the last line to
53 #
54 #precedence ::ffff:0:0/96 100
55 #
56 #
57 # scopev4 <mask> <value>
58 # Add another rule to the RFC 6724 scope table for IPv4 addresses.
59 # By default the scope IDs described in section 3.2 in RFC 6724 are
60 # used. Changing these defaults should hardly ever be necessary.
61 # The defaults are equivalent to:
62 #
63 #scopev4 ::ffff:169.254.0.0/112 2
64 #scopev4 ::ffff:127.0.0.0/104 2
65 #scopev4 ::ffff:0.0.0.0/96 14
~
1 line less; before #1 09:58:15
```

65,1 Bot

:w -> 存盘但不退出



```
vim man
38 # site-local IPv6 addresses cannot be used while the IPv4 address is
39 # (at least for the foreseeable future) NATed. We also treat Teredo
40 # tunnels special.
41 #
42 # precedence <mask> <value>
43 # Add another rule to the RFC 3484 precedence table. See section 2.1
44 # and 10.3 in RFC 3484. The default is:
45 #
46 #precedence ::1/128 50
47 #precedence ::/0 40
48 #precedence 2002::/16 30
49 #precedence ::/96 20
50 #precedence ::ffff:0:0/96 10
51 #
52 # For sites which prefer IPv4 connections change the last line to
53 #
54 #precedence ::ffff:0:0/96 100
55 #
56 #
57 # scopev4 <mask> <value>
58 # Add another rule to the RFC 6724 scope table for IPv4 addresses.
59 # By default the scope IDs described in section 3.2 in RFC 6724 are
60 # used. Changing these defaults should hardly ever be necessary.
61 # The defaults are equivalent to:
62 #
63 #scopev4 ::ffff:169.254.0.0/112 2
64 #scopev4 ::ffff:127.0.0.0/104 2
65 #scopev4 ::ffff:0.0.0.0/96 14
~
"gai.conf" 65L, 2584B written 65,1 Bot
```

gg ->将光标移到首行

```
vim man
1 | Configuration for getaddrinfo(3).
2 #
3 # So far only configuration for the destination address sorting is needed.
4 # RFC 3484 governs the sorting. But the RFC also says that system
5 # administrators should be able to overwrite the defaults. This can be
6 # achieved here.
7 #
8 # All lines have an initial identifier specifying the option followed by
9 # up to two values. Information specified in this file replaces the
10 # default information. Complete absence of data of one kind causes the
11 # appropriate default information to be used. The supported commands include:
12 #
13 # reload <yes|no>
14 # If set to yes, each getaddrinfo(3) call will check whether this file
15 # changed and if necessary reload. This option should not really be
16 # used. There are possible runtime problems. The default is no.
17 #
18 # label <mask> <value>
19 # Add another rule to the RFC 3484 label table. See section 2.1 in
20 # RFC 3484. The default is:
21 #
22 #label ::1/128 0
23 #label ::/0 1
24 #label 2002::/16 2
25 #label ::/96 3
26 #label ::ffff:0:0/96 4
27 #label fec0::/10 5
28 #label fc00::/7 6
29 #label 2001:0::/32 7
1,1 Top
```

```

1 Hello,this is vim world!# Configuration for getaddrinfo(3).
2 #
3 # So far only configuration for the destination address sorting is needed.
4 # RFC 3484 governs the sorting. But the RFC also says that system
5 # administrators should be able to overwrite the defaults. This can be
6 # achieved here.
7 #
8 # All lines have an initial identifier specifying the option followed by
9 # up to two values. Information specified in this file replaces the
10 # default information. Complete absence of data of one kind causes the
11 # appropriate default information to be used. The supported commands include:
12 #
13 # reload <yes|no>
14 #   If set to yes, each getaddrinfo(3) call will check whether this file
15 #   changed and if necessary reload. This option should not really be
16 #   used. There are possible runtime problems. The default is no.
17 #
18 # label <mask> <value>
19 #   Add another rule to the RFC 3484 label table. See section 2.1 in
20 #   RFC 3484. The default is:
21 #
22 #label ::1/128      0
23 #label ::/0         1
24 #label 2002::/16    2
25 #label ::/96        3
26 #label ::ffff:0:0/96 4
27 #label fec0::/10    5
28 #label fc00::/7     6
29 #label 2001:0::/32  7
-- INSERT --

```

```

1 Hello,this is vim world!# Configuration for getaddrinfo(3).
2 #
3 # So far only configuration for the destination address sorting is needed.
4 # RFC 3484 governs the sorting. But the RFC also says that system
5 # administrators should be able to overwrite the defaults. This can be
6 # achieved here.
7 #
8 # All lines have an initial identifier specifying the option followed by
9 # up to two values. Information specified in this file replaces the
10 # default information. Complete absence of data of one kind causes the
11 # appropriate default information to be used. The supported commands includ
12 #
13 # reload <yes|no>
14 #   If set to yes, each getaddrinfo(3) call will check whether this file
15 #   changed and if necessary reload. This option should not really be
16 #   used. There are possible runtime problems. The default is no.
17 #
18 # label <mask> <value>
19 #   Add another rule to the RFC 3484 label table. See section 2.1 in
20 #   RFC 3484. The default is:
21 #
22 #label ::1/128      0
23 #label ::/0         1
24 #label 2002::/16    2
25 #label ::/96        3
26 #label ::ffff:0:0/96 4
27 #label fec0::/10    5
28 #label fc00::/7     6
29 #label 2001:0::/32  7
:q!

```

```
(dolphln@Dolphln)-[~/vi]  
$ vim gai.conf
```

```
(dolphln@Dolphln)-[~/vi]  
$ head -n 3 gai.conf
```

```
# Configuration for getaddrinfo(3).
```

```
#
```

```
# So far only configuration for the destination address sorting is needed.
```

#### 四、 实验过程分析与讨论

遇到的困难在最后的 vim 的部分，对于 vim 的具体操作还是不太熟悉，尤其是字符串匹配部分，经过上网查阅相关资料最终了解了字符串匹配模式各个指令的使用方法和参数的编排方式。

#### 五、指导教师意见

指导教师签字：卢洋

# 实验报告

实验名称	实验四 用户和用户组管理		
实验教室	丹青 922	实验日期	2023 年 3 月 25 日
学 号	2021212614	姓 名	温晖
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`

## 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、 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` ;

```

(dolphin@Do1phln)-[~]
$ sudo useradd foo -c "bar" -s /bin/sh -d /home/jone
[sudo] password for dolphin:
Sorry, try again.
[sudo] password for dolphin:

(dolphin@Do1phln)-[~]
$ sudo passwd foo
New password:
Retype new password:
passwd: password updated successfully

(dolphin@Do1phln)-[~]
$ cat /etc/passwd | tail -n 3
linuxcool1:x:521:521::/home/zhangsan1:/bin/bash
linuxcool2:x:521:521::/home/zhangsan2:/bin/bash
foo:x:1001:1001:bar:/home/jone:/bin/sh

```

- 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

```

(dolphin@Do1phln)-[~]
$ sudo usermod foo -u 2000 -s /bin/csh

```

- 3、 使用命令从用户 jone 切换到 root。

```

(dolphin@Do1phln)-[~]
$ su foo
Password:
% whoami
foo
% su
Password:
(root@Do1phln)-[/home/dolphin]
# |

```

2

- 4、 删除 foo 用户，并在删除该用户的同时一并删除其家目录

```
(dolphln@dolphln)-[~]  
$ sudo userdel -r foo  
[sudo] password for dolphln:  
userdel: foo mail spool (/var/mail/foo) not found  
userdel: foo home directory (/home/jone) not found
```

- 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。

```
(root@dolphln)-[/home/dolphln]  
# vim newuserfile  
  
(root@dolphln)-[/home/dolphln]  
# vim passwdfile  
linuxcool0:123456  
  
(root@dolphln)-[/home/dolphln]  
# cat newuserfile passwdfile  
linuxcool0:x:520:520::/home/zhangsan0:/bin/bash  
linuxcool1:x:521:521::/home/zhangsan1:/bin/bash  
linuxcool2:x:521:521::/home/zhangsan2:/bin/bash  
zhangsan0:123456  
zhangsan1:123456  
zhangsan2:123456  
  
(root@dolphln)-[/home/dolphln]  
# newusers newuserfile  
  
(root@dolphln)-[/home/dolphln]  
# cat /etc/passwd | tail -n 3  
linuxcool0:x:520:520::/home/zhangsan0:/bin/bash  
linuxcool1:x:521:521::/home/zhangsan1:/bin/bash  
linuxcool2:x:521:521::/home/zhangsan2:/bin/bash
```

```
(root@dolphln)-[/home/dolphln]  
# chpasswd < passwdfile  
  
(root@dolphln)-[/home/dolphln]  
# cat /etc/passwd | tail -n 3  
linuxcool0:x:520:520::/home/zhangsan0:/bin/bash  
linuxcool1:x:521:521::/home/zhangsan1:/bin/bash  
linuxcool2:x:521:521::/home/zhangsan2:/bin/bash
```

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



3000。

```
(root👤Dolphin)-[/home/dolphin]
# 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👤Dolphin)-[/home/dolphin]
# gpasswd -a linuxcool1 group1
Adding user linuxcool1 to group group1

(root👤Dolphin)-[/home/dolphin]
# gpasswd -a linuxcool2 group1
Adding user linuxcool2 to group group1
```

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。

```
(root👤Dolphin)-[/home/dolphin]
# su linuxcool1
linuxcool1@Dolphin:/home/dolphin$ cat /etc/shadow
cat: /etc/shadow: Permission denied
```

```
root    ALL=(ALL:ALL) ALL
linuxcool1    ALL=(ALL) ALL
```

```
(root@dolphin)-[/home/dolphin]
# su linuxcool1
linuxcool1@dolphin:/home/dolphin$ cat /etc/shadow
cat: /etc/shadow: Permission denied
linuxcool1@dolphin:/home/dolphin$ sudo cat /etc/shadow
[sudo] password for linuxcool1:
root:$y$j9T$E88TkoPzTzjXG2txocgs1$F6XWLVwAK0TQ4a0iv7/3xdsUfeIL5RIae07mnCXjVi5:19109:0:99999:7:::
daemon*:18883:0:99999:7:::
bin*:18883:0:99999:7:::
sys*:18883:0:99999:7:::
sync*:18883:0:99999:7:::
games*:18883:0:99999:7:::
man*:18883:0:99999:7:::
lp*:18883:0:99999:7:::
mail*:18883:0:99999:7:::
news*:18883:0:99999:7:::
uucp*:18883:0:99999:7:::
proxy*:18883:0:99999:7:::
www-data*:18883:0:99999:7:::
backup*:18883:0:99999:7:::
list*:18883:0:99999:7:::
irc*:18883:0:99999:7:::
gnats*:18883:0:99999:7:::
nobody*:18883:0:99999:7:::
_apt*:18883:0:99999:7:::
systemd-timesync*:18883:0:99999:7:::
systemd-network*:18883:0:99999:7:::
systemd-resolve*:18883:0:99999:7:::
dolphin:$y$j9T$DT1lnHe4ZrLpVRQxLV3IE1$inwXo2aefaAD.6nKk7oXBkZqpCnI19FcZrSL8GyHn89:18895:0:99999:7:::
messagebus*:18937:0:99999:7:::
linuxcool0:$y$j9T$SffU9E2ZDMghMA8angFmo0$TAuY5ompBURJ9Yssv6BsAZ3RgH.AcF105p4gRHYrvH9:19109:0:99999:7:::
linuxcool1:$y$j9T$HTMH2d4Fvv.WelrZqDpn91$fAhNiiRM/a72TfW5UcGUaobmbyFszJ8PmXadJFUD1K1:19109:0:99999:7:::
linuxcool2:$y$j9T$05tNjcwZvM2D/advZG0df/$vcR09FrflU4EinNRwaD3M6Kd4tt12zhCeeHxYcPb7/8:19109:0:99999:7:::
linuxcool1@dolphin:/home/dolphin$
```

## 八、 实验过程分析与讨论

遇到的困难在最后实验最后修改 `sudoers` 的部分，在不知道 `sudoers` 文件具体未知的情况下，应用 `find` 命令最终找到了 `sudoers`，并且使用 `root` 用户强制修改了内容。

## 五、 指导教师意见

指导教师签字：卢洋

# 实验报告

实验名称	实验五 <b>Shell</b> 程序的创建及条件判断语句		
实验教室	丹青 922	实验日期	2023 年 4 月 6 日
学 号	2021212614	姓 名	温晖
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

- 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、 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

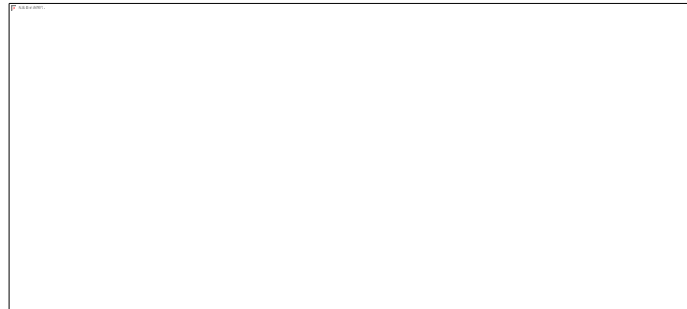
- 1、 定义变量 foo 的值为 200，并将其显示在屏幕上。（终端上执行）



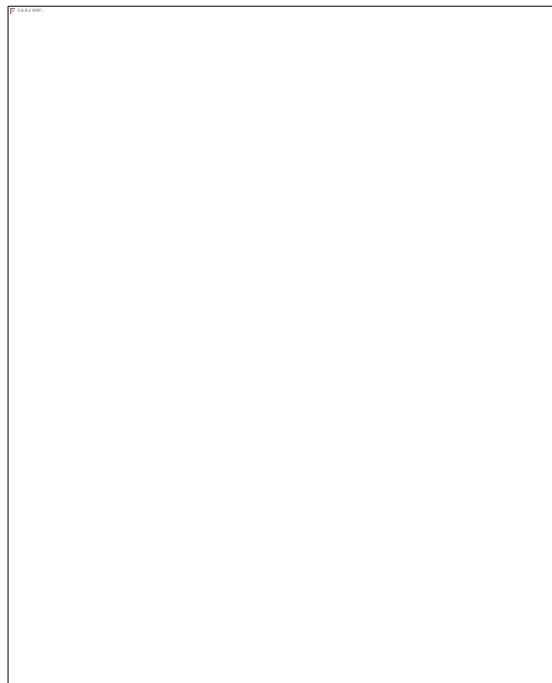
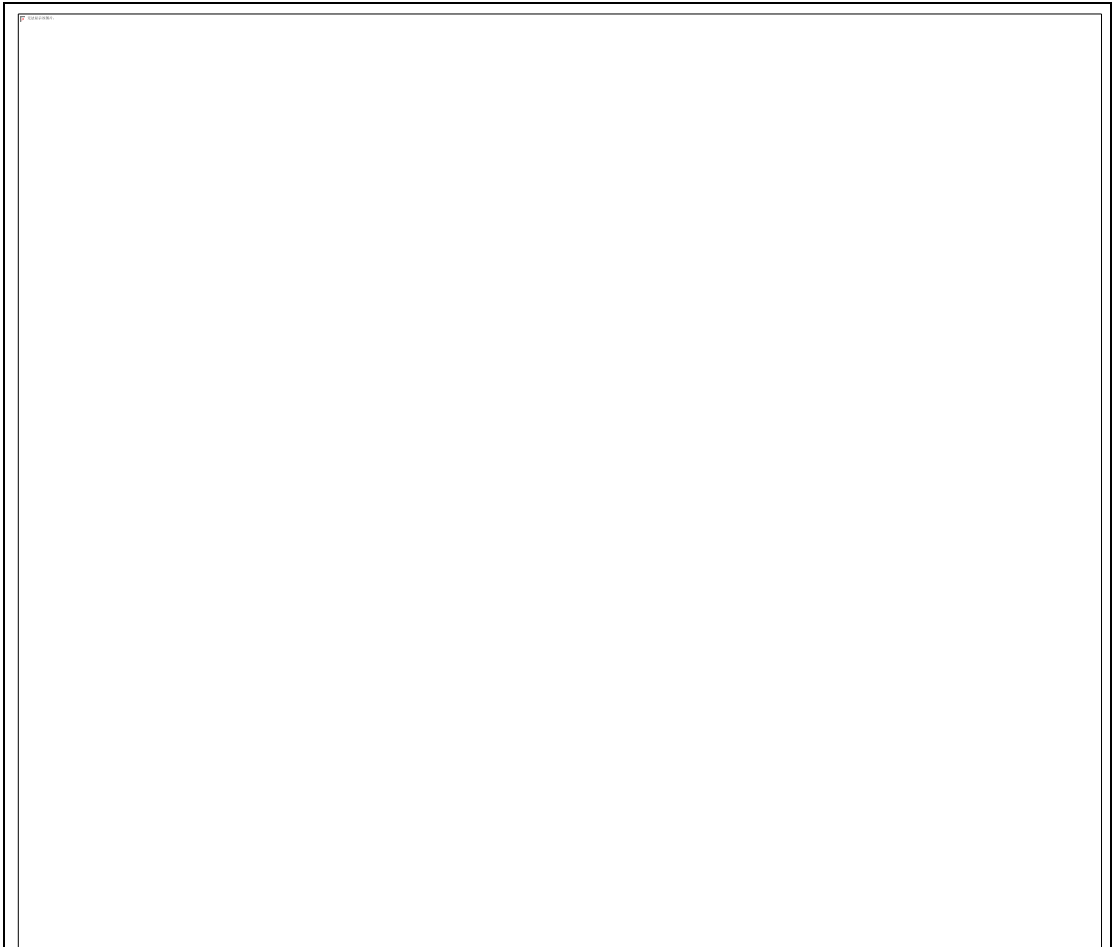
- 2、 定义变量 bar 的值为 100，并使用 test 命令比较其值是否大于 150，并显示 test 命令的退出码。（终端上执行）



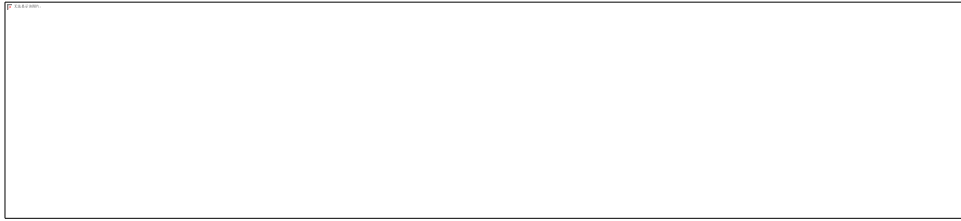
3、创建一个简单的 Shell 程序，其功能为显示计算机主机名（hostname）和系统时间（date）。



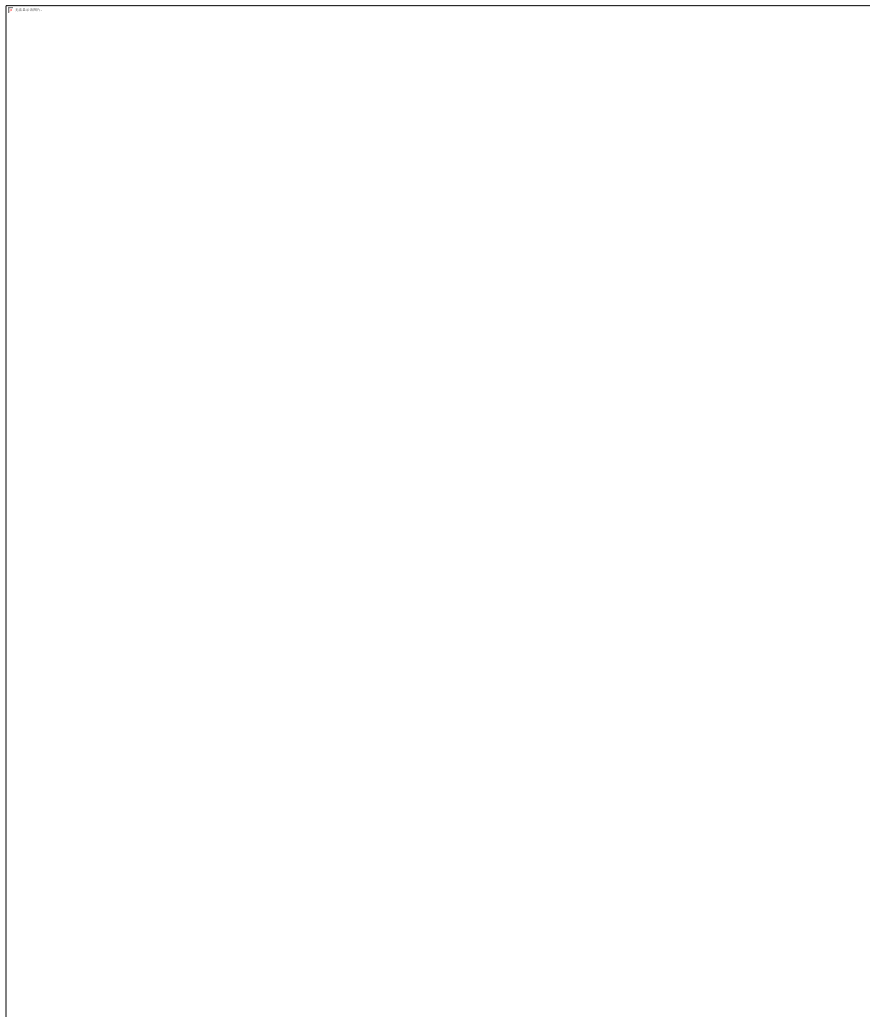
3、 创建一个简单的 Shell 程序，要求带一个参数，判断该参数是否是水仙花数。所谓水仙花数是指一个 3 位数，它的每个位上的数字的 3 次幂之和等于它本身。例如  $153=1^3+3^3+5^3$ ，153 是水仙花数。编写程序时要求首先进行参数个数判断，判断是否带了一个参数，如果没有参数则给出提示信息，否则给出该数是否是水仙花数。要求对 153, 124, 370 分别进行测试判断。



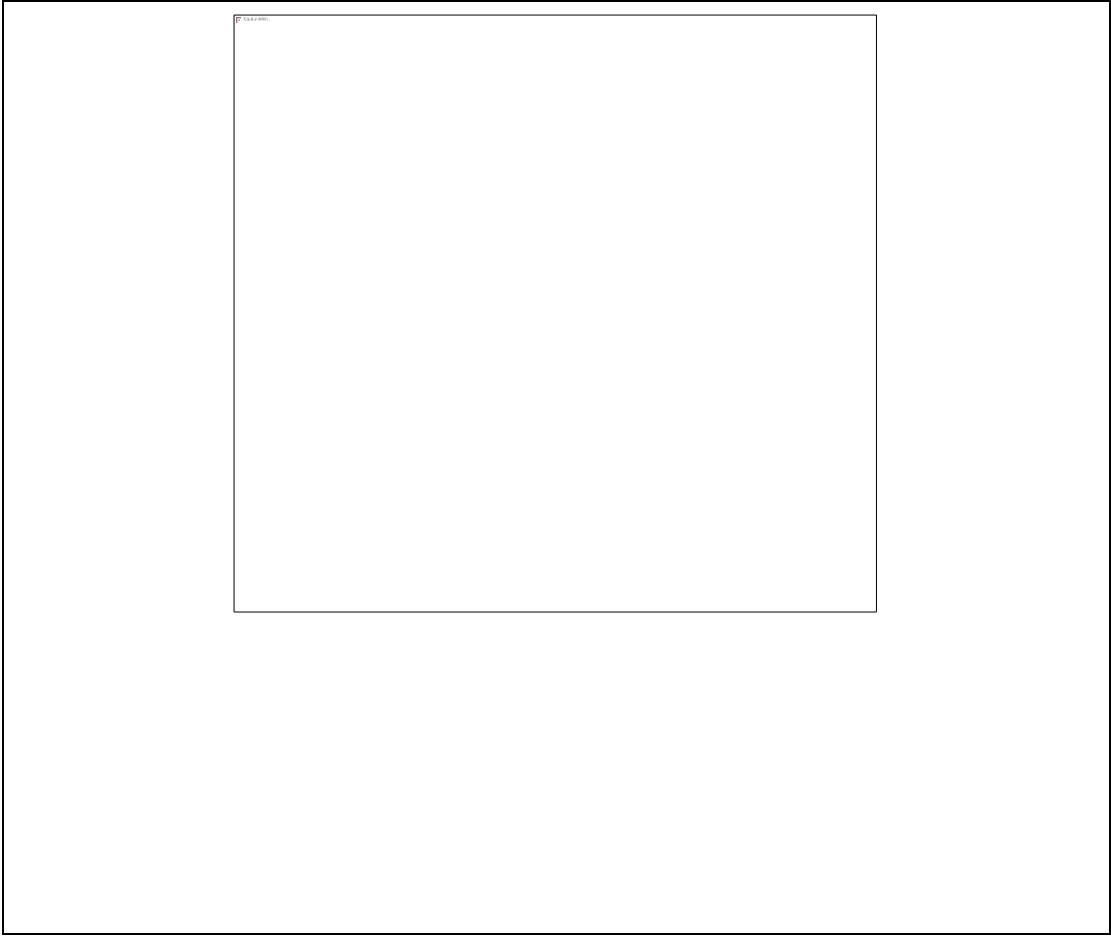
5、创建一个简单的 shell 程序，输入 3 个数，输出这 3 个数的和。



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。







## 十二、 实验过程分析与讨论

遇到的困难在 shell 编程，与 C 语言编程逻辑大同小异，在输入变量获取上还是不太理解，通过自行搜索和尝试后最终掌握了相关知识，编写出了 shell 程序。

## 五、指导教师意见

指导教师签字：卢洋

# 实验报告

实验名称	实验六 <b>Shell</b> 循环控制语句		
实验教室	丹青 922	实验日期	2023 年 4 月 13 日
学 号	2021212614	姓 名	温晖
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) 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

### 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) 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。（可以在当前目录下先建立几个 \*.c 文件，用来测试，复制到的指定目录可以自己建立一个）

```
(dolphins@Dolphins)-[~]  
$ touch a.c b.c c.c d.c e.c f.c g.c  
  
(dolphins@Dolphins)-[~]  
$ ls  
a.c  b.c  c.c  d.c  e.c  f.c  g.c  go  tmp2.sh  tmp.sh  
  
(dolphins@Dolphins)-[~]  
$ mkdir dir1
```

```

(dolphin@dolphin)-[~]
$ bash tmp3.sh
./a.c ./b.c ./c.c ./d.c ./e.c ./f.c ./g.c
Input dst path:dir1

(dolphin@dolphin)-[~]
$ ls dir1
a.c  b.c  c.c  d.c  e.c  f.c  g.c

```

```

1 #!/bin/bash
2 filelist=$(ls ./tmp/*.c)
3 echo $filelist
4
5 read -p "Input dst path: " dir
6 test -e $dir || mkdir $dir
7
8 for i in $filelist
9 do
10     cp $i $dir
11 done

```

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

```

#!/bin/bash

i=1
sum=0

while [[ $i -le 20 ]]; do
    if [[ $((i%2)) -eq 0 ]]; then
        let sum=sum+i
    fi

    echo $sum

    let i=i+1
done

echo "Sum is: $sum"

```

```

(dolphin@Dolphin)-[~]
$ vim tmp4.sh

(dolphin@Dolphin)-[~]
$ bash tmp4.sh
0
2
2
6
6
12
12
20
20
30
30
42
42
56
56
72
72
90
90
110
Ans: 110

```

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

```

#!/bin/bash

i=0
sum=0

until [[ $i -gt 10 ]]; do
    let sum=sum+i*i
    let i=i+1
done

echo $sum

```

```

(dolphin@Dolphin)-[~]
$ bash sqreadd.sh
385

```

(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。

Break

```
(dolpnl⊗Dolpnl)-[~]  
$ bash tmp5.sh  
a1234  
b1234  
c1234  
d1234
```

Break 2

```
(dolpnl⊗Dolpnl)-[~]  
$ bash tmp5.sh  
a1234
```

Continue

```
(dolpnl⊗Dolpnl)-[~]  
$ bash tmp5.sh  
a1234678910  
b1234678910  
c1234678910  
d1234678910
```

Continue2

```
(dolpnl⊗Dolpnl)-[~]  
$ bash tmp5.sh  
a1234b1234c1234d1234
```

## 十六、 实验过程分析与讨论

通过本次实验,最大的收获是从最后一部分替换 `continue` 和 `break` 理解了其在循环内所体现的作用和具体用法。

## 五、指导教师意见

指导教师签字：卢洋



# 实验报告

实验名称	实验七 Shell 函数		
实验教室	丹青 922	实验日期	2023 年 4 月 20 日
学 号	2021212614	姓 名	温晖
专业班级	计算机科学与技术六班		
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 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。

## 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

- a) 编写 shell 脚本，定义一个函数，对两个数的和进行求解，并输出结果。

```
1 #/bin/sh
2 function sum() {
3     return $(( $1+$2 ))
4 }
5
6 read -p "Please input a: " a
7 read -p "Please input b: " b
8 sum $a $b
9
10 echo $a + $b = $?
```

```

(dolphin@dolphin)-[~]
$ vim add.sh

(dolphin@dolphin)-[~]
$ bash add.sh
Please input a:1
Please input b:2
1 + 2 = 3

```

2、编写shell脚本，该脚本中定义一个递归函数，求n的阶乘。

```

#!/bin/bash

function n(){
    local n=$1
    if [[ $n -eq 0 ]]; then
        return 1
    else
        n=$((n-1))
        return $((n*$?))
    fi
}

read -p "a:" a

n $a

echo $?

```

```

(dolphin@dolphin)-[~]
$ vim fib.sh

(dolphin@dolphin)-[~]
$ bash fib.sh
a:2
2

(dolphin@dolphin)-[~]
$ bash fib.sh
a:4
24

```

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

```
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..."  
first
```

```
(dolpnl@Dolpnl)-[~]  
$ bash tmp4.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、 实验过程分析与讨论

本次通过最后一个问题，了解到了在 shell 编程里面可以利用声明 first, second, third 来指定程序的运行次序，这个方法在之前的编程中都没有遇到过，需要经常复习回顾，在有些需要临时改变运行逻辑的脚本中应该会很有用。

## 五、指导教师意见

指导教师签字：卢洋

# 实验报告

实验名称	实验八 sed 和 awk		
实验教室	丹青 922	实验日期	2023 年 4 月 27 日
学 号	2021212614	姓 名	温晖
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、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

## 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、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

```
(dolphln@Dolphln)-[~]  
$ 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

```
(dolphln@Dolphln)-[~]  
$ cat -n quote.txt | sed -n '/music/p'  
2 It was an evening of splendid music and company.
```

(3) 在第 4 行后面追加文件 “hello world!”

```
(dolphln@Dolphln)-[~]  
$ cat -n quote.txt | sed '4a hello world'  
1 The honeysuckle band played all night long for only $90.  
2 It was an evening of splendid music and company.  
3 Too bad the disco floor fell through at 23:10.  
4 The local nurse Miss P.Neave was in attendance.  
hello world
```

(4) 将文本 “The” 修改为 “Quod”

```
(dolphln@Dolphln)-[~]  
$ cat -n quote.txt | sed 's/The/Quod/g'  
1 Quod honeysuckle band played all night long for only $90.  
2 It was an evening of splendid music and company.  
3 Too bad the disco floor fell through at 23:10.  
4 Quod local nurse Miss P.Neave was in attendance.
```

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

```
(dolphln@Dolphln)-[~]  
$ cat -n quote.txt | sed '3c This is the third line.'  
1 The honeysuckle band played all night long for only $90.  
2 It was an evening of splendid music and company.  
This is the third line.  
4 The local nurse Miss P.Neave was in attendance.
```

(6) 删除第 2 行内容。



```
(dolphins@Dolphins)~$ cat -n quote.txt | sed '2d'
1 The honeysuckle band played all night long for only $90.
3 Too bad the disco floor fell through at 23:10.
4 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。

```
(dolphins@Dolphins)~$ var=evening

(dolphins@Dolphins)~$ echo $var
evening

(dolphins@Dolphins)~$ cat -n quote.txt | sed -n "/$var/p"
2 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

```
(dolphins@Dolphins)~$ cat numbers.txt | awk -v FS=':' '{print $2}'
two
five
```

```
(dolphins@Dolphins)~$ cat numbers.txt | awk -v FS=' ' '{print $2}'
:
:
```

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 foo.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

```
(dolpnlⓈDolpnl)-[~]  
$ cat foo.txt  
2 4 3  
15 46 79
```

```
(dolpnlⓈDolpnl)-[~]  
$ awk 'BEGIN{count=0;print "even:"}{for(i=1;i<=NF;i++){if($i%2==0){count=count+1;print $i}}END{print "numbers:\n", count}' fo  
.txt  
even:  
2  
4  
46  
numbers:  
3
```

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

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

```
read -p "enter search pattern: " pattern
```

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

```
(do1phln@Do1phln)-[~]  
$ cat >info.txt  
test1  
test2  
test3  
asd1  
asd2  
asd3  
^C  
  
(do1phln@Do1phln)-[~]  
$ bash tmp4.sh  
enter search pattern: test  
test1  
test2  
test3  
3 found.
```

上述脚本实现了自定义模式匹配

## 二十四、 实验过程分析与讨论

遇到的困难在最后的 awk 部分，awk 的 BEGIN+END 结构查询了很久资料才搞明白，现在已基本掌握 awk 的基本使用语法。

## 五、指导教师意见

指导教师签字：卢洋