

实验报告

实验名称	实验一 Linux 常用命令（一）		
实验教室	丹青 922	实验日期	2023 年 3 月 8 日
学 号	2021213189	姓 名	贺腾艺
专业班级	计算机科学与技术 06 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

```
[thaddeus7@hty ~]$ cd /etc
[thaddeus7@hty etc]$ pwd
/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。

```
[thaddeus7@hty ~]$ ls -a
.          baz      file4      path1      .test.txt.swp
..         .cache   foo.txt    path2      Videos
a          .config  .ICEauthority path3      .vim
bar.txt    .cquery  install.sh Pictures    .VimForCpp
.bash_history .dbus    .lfCache   .pki       .viminfo
.bash_logout Desktop  .local     Public      .vimrc
.bash_profile Documents .mozilla   tar1.txt    .ycm_extra_conf.py
.bashrc     Downloads Music       Templates
.basjrc     .esd_auth package.tar test.c
```

3. 使用命令创建目录/home/lyj/linux，然后删除该目录。

```
[thaddeus7@hty ~]$ ls
baz      Documents  file4      Music  path2  Pictures  Templates
Desktop  Downloads  install.sh path1  path3  Public    Videos
[thaddeus7@hty ~]$ mkdir linux
[thaddeus7@hty ~]$ ls
baz      Documents  file4      linux  path1  path3  Public  Videos
Desktop  Downloads  install.sh Music  path2  Pictures  Templates
[thaddeus7@hty ~]$ rm -r linux
[thaddeus7@hty ~]$ ls
baz      Documents  file4      Music  path2  Pictures  Templates
Desktop  Downloads  install.sh path1  path3  Public    Videos
```

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

```
[thaddeus7@hty ~]$ cat > foo
Hello Linux!
[thaddeus7@hty ~]$ cat foo
Hello Linux!
```

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

```
[thaddeus7@hty ~]$ mkdir foo.bak
[thaddeus7@hty ~]$ ls
baz      Documents  file4  foo.bak  Music  path2  Pictures  Templates
Desktop  Downloads  foo    install.sh  path1  path3  Public    test
[thaddeus7@hty ~]$ cp foo foo.bak
[thaddeus7@hty ~]$ ls
baz      Documents  file4  foo.bak  Music  path2  Pictures  Templates
Desktop  Downloads  foo    install.sh  path1  path3  Public    test
[thaddeus7@hty ~]$ cd foo.bak
[thaddeus7@hty foo.bak]$ ls
foo
[thaddeus7@hty foo.bak]$ rm foo
[thaddeus7@hty foo.bak]$ rmdir foo.abk
rmdir: 删除 "foo.abk" 失败: 没有那个文件或目录
[thaddeus7@hty foo.bak]$ cd ..
[thaddeus7@hty ~]$ rm -r foo.bak
[thaddeus7@hty ~]$ ls
baz      Documents  file4  install.sh  path1  path3  Public    test
Desktop  Downloads  foo    Music       path2  Pictures  Templates  Videos
```

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

```
[root@hty ~]# head -3 /etc/fonts/conf.d/62-google-crosextra-carlito-fontconfig.conf
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fontconfig SYSTEM "../fonts.dtd">
<fontconfig>
[root@hty ~]# head -5 /etc/fonts/conf.d/62-google-crosextra-carlito-fontconfig.conf
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fontconfig SYSTEM "../fonts.dtd">
<fontconfig>
    <alias>
        <family>sans-serif</family>
[root@hty ~]# head -3 /etc/fonts/conf.d/62-google-crosextra-carlito-fontconfig.conf
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fontconfig SYSTEM "../fonts.dtd">
<fontconfig>
[root@hty ~]# tail -5 /etc/fonts/conf.d/62-google-crosextra-carlito-fontconfig.conf
        <family>sans-serif</family>
    </default>
</alias>
</fontconfig>
```

7、分屏查看文件/etc/adduser.conf 的内容。

```

<?xml version="1.0" encoding="UTF-8" ?> # 399 "/usr/include/features.h" 3 4 0/0 ^ v x
<!DOCTYPE fontconfig SYSTEM "../fonts.dtd">
<fontconfig>
  <!-- Generic names -->
  <alias>
    <family>Jomolhari</family>
    <default>
      <family>serif</family>
    </default>
  </alias>
  <!-- Locale-specific overrides -->
  <match>
    <test name="lang">
      <string>bo-cn</string>
    </test>
    <test name="family">
      <string>serif</string>
    </test>
    <edit name="family" mode="prepend">
      <string>Jomolhari</string>
    </edit>
  </match>
  <match>
    <test name="lang">
      <string>bo-in</string>
    </test>
    <test name="family">
      <string>serif</string>
    </test>
  </match>

```

--More-- (36%)

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

```

[thaddeus7@hty ~]$ 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

```
[thaddeus7@hty ~]$ sort -r facebook.txt  
sohu 100 4500  
guge 50 3000  
google 110 5000  
baidu 100 5000
```

(2) 按公司人数排序

```
[thaddeus7@hty ~]$ sort -k2n 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

```
[thaddeus7@hty ~]$ sort -k2n -k3nr facebook.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

```
[thaddeus7@hty ~]$ sort -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。

```
[thaddeus7@hty ~]$ sort -k1.2 facebook.txt  
baidu 100 5000  
sohu 100 4500  
google 110 5000  
guge 50 3000
```

四、 实验过程分析与讨论

遇到的困难在最后那个实验，排序的部分，对于多重要求和非第一行的排序命令还是不太熟悉，在查询 CSDN 之后学会了相关命令。

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 15 日
学 号	2021213189	姓 名	贺腾艺
专业班级	计算机科学与技术 06 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；

```
[thaddeus7@hty ~]$ mkdir baz
[thaddeus7@hty ~]$ cd baz
[thaddeus7@hty baz]$ cat > qux << EOF
> 123
> 456
> abc
> EOF
```

(2) 在用户目录下查找文件 `qux`，并显示该文件位置信息；

```
[thaddeus7@hty ~]$ find ~ -name qux
/home/thaddeus7/baz/qux
```

(3) 统计文件 `qux` 中所包含内容的行数、字数和字节数；

(4) 在用户目录下查找文件 `qux`，并删除该文件；

```
[thaddeus7@hty baz]$ find ~ -name qux -delete
[thaddeus7@hty baz]$ ls
```

(5) 查看文件夹 `baz` 内容，看一下是否删除了文件 `qux`。

```
[thaddeus7@hty baz]$ ls
[thaddeus7@hty baz]$
```

2. 文件打包

(1) 在用户目录下新建文件夹 `path1`，在 `path1` 下新建文件 `file1` 和 `file2`；

```
[thaddeus7@hty ~]$ mkdir path1
[thaddeus7@hty ~]$ cd path1
[thaddeus7@hty path1]$ touch file1 file2
[thaddeus7@hty path1]$ ls
file1  file2
```

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

```
[thaddeus7@hty ~]$ mkdir path2
[thaddeus7@hty ~]$ cd path2
[thaddeus7@hty path2]$ touch file3
[thaddeus7@hty path2]$ ls
file3
```

(3) 在用户目录下新建文件 file4 ；

```
[thaddeus7@hty ~]$ touch file4
[thaddeus7@hty ~]$ ls
baz      Downloads  path1     Public    workspace
Desktop  file4      path2     Templates
Documents Music      Pictures  Videos
```

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

```
[thaddeus7@hty ~]$ tar -cvf package.tar path1 file4
path1/
path1/file1
path1/file2
file4
```

(5) 查看包 package.tar 的内容；

```
[thaddeus7@hty ~]$ tar -tvf package.tar
drwxrwxr-x thaddeus7/thaddeus7 0 2023-04-17 10:55 path1/
-rw-rw-r-- thaddeus7/thaddeus7 0 2023-04-17 10:55 path1/file1
-rw-rw-r-- thaddeus7/thaddeus7 0 2023-04-17 10:55 path1/file2
-rw-rw-r-- thaddeus7/thaddeus7 0 2023-04-17 10:56 file4
```

(6) 向包 package.tar 里添加文件夹 path2 的内容；

```
[thaddeus7@hty ~]$ tar -rvf package.tar path2
path2/
path2/file3
```

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

```
[thaddeus7@hty ~]$ mkdir path3
[thaddeus7@hty ~]$ cp package.tar path3
```

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

```
[thaddeus7@hty ~]$ cd path3
[thaddeus7@hty path3]$ tar -xvf package.tar
path1/
path1/file1
path1/file2
file4
```

3. 符号链接内容

(1) 新建文件 `foo.txt`，内容为 `123`；

```
[thaddeus7@hty ~]$ echo "123" > foo.txt
```

(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；

```
[thaddeus7@hty ~]$ ln foo.txt bar.txt
[thaddeus7@hty ~]$ diff bar.txt foo.txt
```

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

```
[thaddeus7@hty ~]$ ls -li foo.txt bar.txt
2605132 bar.txt 2605132 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；

```
[thaddeus7@hty ~]$ echo "abc" > bar.txt
[thaddeus7@hty ~]$ diff foo.txt bar.txt
```

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

```
[thaddeus7@hty ~]$ rm foo.txt
[thaddeus7@hty ~]$ ls -li bar.txt
2605132 bar.txt
[thaddeus7@hty ~]$ cat bar.txt
abc
```

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

```
[thaddeus7@hty ~]$ ln -s bar.txt baz.txt  
[thaddeus7@hty ~]$ ls -li bar.txt baz.txt  
2605132 bar.txt 2605134 baz.txt  
[thaddeus7@hty ~]$ diff baz.txt bar.txt
```

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

```
[thaddeus7@hty ~]$ rm bar.txt  
[thaddeus7@hty ~]$ cat baz.txt  
cat: baz.txt: 没有那个文件或目录
```

4. 权限管理

(1) 新建文件 qux.txt；

```
[thaddeus7@hty ~]$ touch qux.txt
```

(2) 为文件 qux.txt 增加执行权限（所有用户都可以执行）

```
[thaddeus7@hty ~]$ chmod +x qux.txt
```

八、 实验过程分析与讨论

本次实验进行了 Linux 系统中的查找、压缩、链接文件和修改权限命令的练习，其中压缩文件的命令选项稍微有些复杂，遇到了一些困难，但最后都顺利解决了。

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 vim 编辑器及 gcc 编译器的使用		
实验教室	丹青 922	实验日期	2023 年 3 月 22 日
学 号	2021213189	姓 名	贺腾艺
专业班级	计算机科学与技术 06 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 vim 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) 在用户目录下新建一个目录, 命名为 workspace1 ;

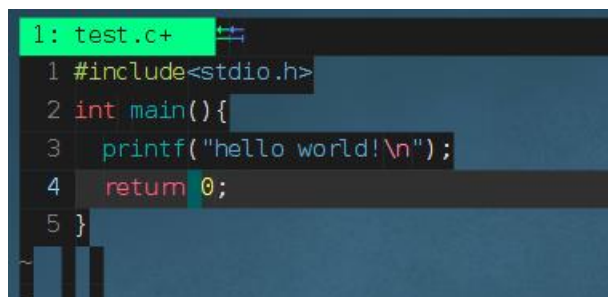
```
[thaddeus7@hty ~]$ mkdir workspace1
```

(2) 进入目录 workspace1 ;

```
[thaddeus7@hty ~]$ cd workspace1  
[thaddeus7@hty 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: #include int main() { printf("hello world!\n"); return 0; }

```
[thaddeus7@hty workspace1]$ vim test.c
```



```
1: test.c+  
1 #include<stdio.h>  
2 int main(){  
3     printf("hello world!\n");  
4     return 0;  
5 }
```

(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, 查看执行结果。

```
[thaddeus7@hty workspace1]$ gcc test.c -o test  
[thaddeus7@hty workspace1]$ ./test  
hello world!
```

2. vim 编辑器的详细使用:

(1) 在用户目录下创建一个名为 workspace2 的目录;

```
[thaddeus7@hty ~]$ mkdir workspace2
```

(2) 进入 workspace2 目录;

```
[thaddeus7@hty ~]$ cd workspace2
```

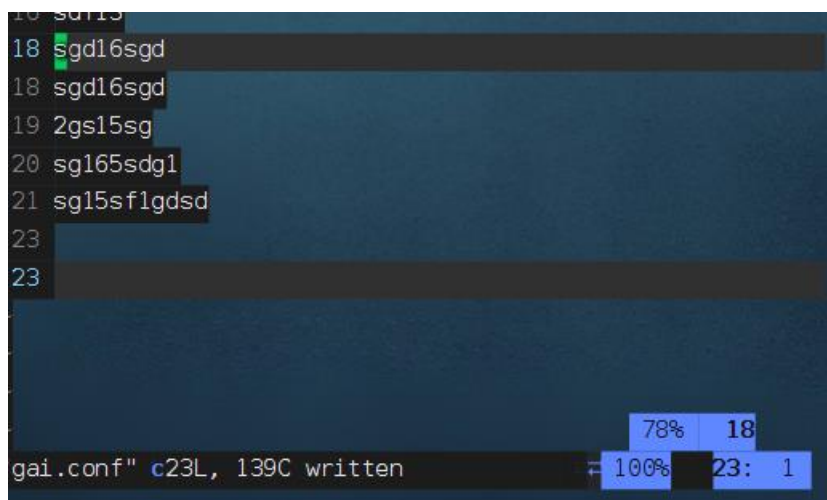
(3) 使用以下命令： 将文件 `/etc/gai.conf` 的内容复制到当前目录下的新建文件 `gai.conf` 中；

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

(4) 使用 `vim` 编辑当前目录下的 `gai.conf` ；

```
[thaddeus7@hty workspace2]$ vim gai.conf
```

(5) 将光标移到第 18 行；



A screenshot of the vim editor interface. The file being edited is `gai.conf`. The cursor is positioned at line 18, column 1. The line content is `sgd16sgd`. The status bar at the bottom indicates the file is `gai.conf`, the cursor is at column 23, line 18, and the file has 1390 characters written.

(6) 复制该行内容；



A screenshot of the vim editor interface showing the command `:18C` being entered. The cursor is at the end of line 18.

(7) 将光标移到最后一行行首；



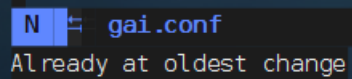
A screenshot of the vim editor interface showing the command `:$` being entered. The cursor is at the end of the file.

(8) 粘贴复制行的内容；

yy

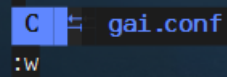
(9) 撤销第 8 步的动作；

u



```
N  gai.conf
Already at oldest change
```

(10) 存盘但不退出；

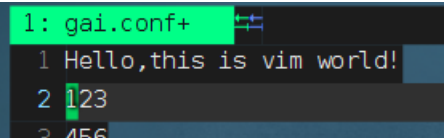


```
C  gai.conf
:w
```

(11) 将光标移到首行；

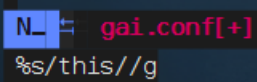
gg

(12) 插入模式下输入 "Hello, this is vim world!" ；



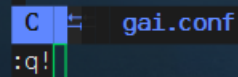
```
1: gai.conf+
1 Hello,this is vim world!
2 123
3 456
```

(13) 删除字符串 "this" ；



```
N  gai.conf[+]
%s/this//g
```

(14) 强制退出 vim ，不存盘。



```
C  gai.conf
: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 3 月 29 日
学 号	2021213189	姓 名	贺腾艺
专业班级	计算机科学与技术 06 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` 、 `gpasswd` ；
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, 并设置登录口令为 123456;

```
[root@hty ~]# sudo useradd -c "bar" -s /bin/sh -d /home/foo -p $(openssl p  
asswd -1 "123456") foo
```

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@hty ~]# usermod -u 2000 -s /bin/csh foo  
[root@hty ~]# su - foo  
Last login: Thu Apr 27 17:25:50 CST 2023 on pts/0
```

3. 从用户 foo 切换到 root

```
[root@hty ~]# su - root  
Last login: Thu Apr 27 17:30:34 CST 2023 on pts/0
```

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@hty ~]# userdel foo -r
```

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

2,1

All

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

```
~
```

```
~
```

```
~
```

```
~
```

```
~
```

```
~
```

```
~
```

```
~
```

```
~
```

```
~
```

```
~
```

```
~
```

```
~
```

```
~
```

```
~
```

```
~
```

```
~
```

```
~
```

```
~
```

```
~
```

```
~
```

```
~
```

```
~
```

```
~
```

```
~
```

```
"userpwdfile.txt" 3L, 29C
```

```
[root@hty ~]# newusers <userfile
newusers: line 1: invalid line
newusers: line 2: invalid line
newusers: error detected, changes ignored
[root@hty ~]# cat userpwdfile.txt | chpasswd
```

6. 创建用户组 group1，并在创建时设置其 GID 为 3000.

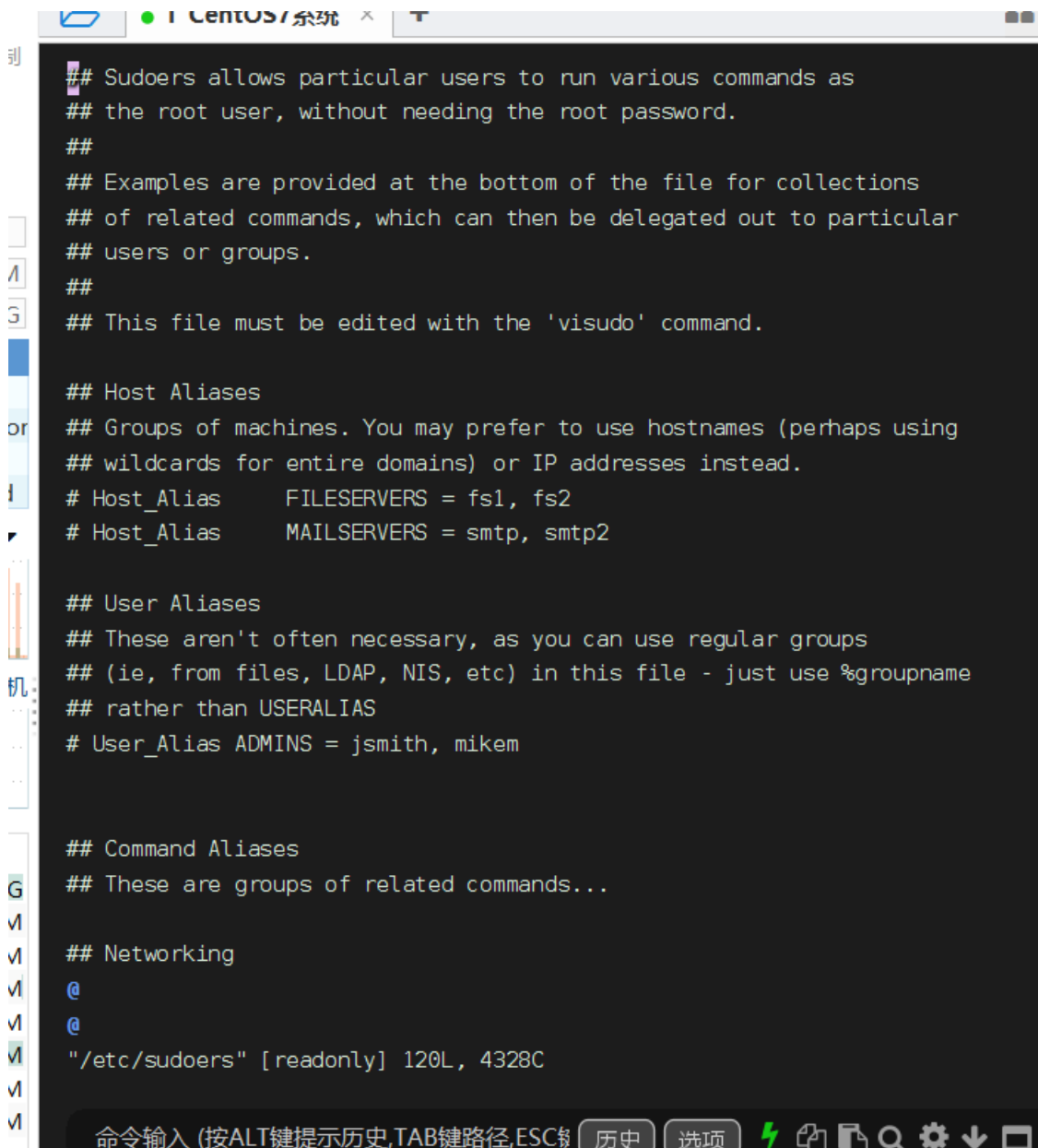
```
[root@hty ~]# groupadd group1 -g 3000
```

7. 在用户组 group1 中添加两个之前批量创建的用户

```
[root@hty ~]# usermod -g group1 user1
```

```
[root@hty ~]# usermod -g group1 user2
```

8. 切换到 group1 组中的任一用户,在该用户下使用 sudo 命令查看 /etc/shadow 文件,检查上述操作是否可以执行;若不能执行,修改 sudoers 文件使得该用户可以查看文件 /etc/shadow 的内容



```
## Sudoers allows particular users to run various commands as
## the root user, without needing the root password.
##
## Examples are provided at the bottom of the file for collections
## of related commands, which can then be delegated out to particular
## users or groups.
##
## This file must be edited with the 'visudo' command.

## Host Aliases
## Groups of machines. You may prefer to use hostnames (perhaps using
## wildcards for entire domains) or IP addresses instead.
# Host_Alias      FILESERVERS = fs1, fs2
# Host_Alias      MAILSERVERS = smtp, smtp2

## User Aliases
## These aren't often necessary, as you can use regular groups
## (ie, from files, LDAP, NIS, etc) in this file - just use %groupname
## rather than USERALIAS
# User_Alias      ADMINS = jsmith, mikem

## Command Aliases
## These are groups of related commands...

## Networking
@
@
"/etc/sudoers" [readonly] 120L, 4328C

命令输入 (按ALT键提示历史,TAB键路径,ESC键) 历史 选项
```


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

/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 4 月 5 日
学 号	2021213189	姓 名	贺腾艺
专业班级	计算机科学与技术 06 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

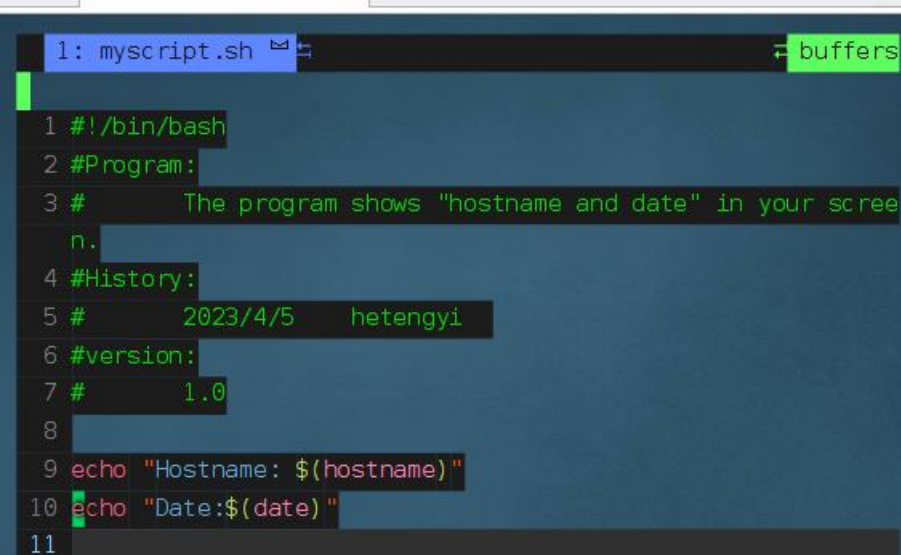
1. 定义变量 foo 的值为 200，并将其显示在屏幕上（终端上执行）；

```
[thaddeus7@hty ~]$ foo=200
[thaddeus7@hty ~]$ echo $foo
200
```

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）；

```
[thaddeus7@hty ~]$ bar=100
[thaddeus7@hty ~]$ test $bar -gt 150
[thaddeus7@hty ~]$ echo $?
1
```

3. 创建一个 Shell 程序，其功能为显示计算机主机名（hostname）和系统时间（date）；



```
1: myscript.sh buffers
1 #!/bin/bash
2 #Program:
3 #    The program shows "hostname and date" in your screen.
4 #History:
5 #    2023/4/5    hetengyi
6 #version:
7 #    1.0
8
9 echo "Hostname: $(hostname)"
10 echo "Date:$(date)"
11
```

```
[thaddeus7@hty ~]$ vim myscript.sh
[thaddeus7@hty ~]$ sh myscript.sh
Hostname: hty
Date:2023年 04月 27日 星期四 14:02:41 CST
```

4.创建一个 Shell 程序，要求可以处理一个输入参数，判断该输入参数是否为水仙花数；

```
1: check1.sh  buf
1 #!/bin/bash
2 #Function:输入一个参数，判断是不是水仙花数
3
4 echo "Total paramter are: $#"
```

5 test \$# -eq 0 && echo "You don't give one paramter at least" && exit 0

6

7 for var in \$@

8 do

9 echo "The num is: \$var"

10 var0=\$var

11 var1=\$((var0/100))

12 var0=\$((var0%100))

13

14 var2=\$((var0/10))

15 var3=\$((var0%10))

16 if [\$(((\$var1*\$var1*\$var1+\$var2*\$var2*\$var2+\$var3*\$var3*\$var3)) -eq \$var)];then

17 echo "\$var is 水仙花数"

18 else

19 echo "\$var 不是水仙花数"

20 fi

21 done

22

```
[thaddeus7@hty ~]$ vim check1.sh
[thaddeus7@hty ~]$ sh check1.sh 156
Total paramter are: 1
The num is: 156
156 不是水仙花数
```

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；

```
1: check2.sh  buf
1 #!/bin/bash
2 #Program: The program calculates sum.
3 #History: 2023/4/5 hetengyi release3
4
5 sum=$(( $1 + $2 + $3 ))
6 echo "三个输入变量的和为: $sum"
7
```



```
[thaddeus7@hty ~]$ vim check2.sh
[thaddeus7@hty ~]$ sh check2.sh 1 2 3
三个输入变量的和为：6
```

6. 创建一个 Shell 程序，输入学生成绩，给出该成绩对应的等级：90 分以上为 A，80-90 为

```
if
elif
else
fi
```

B，70-80 为 C，60-70 为 D，小于 60 分为 E。要求使用 **实现**

```
[thaddeus7@hty ~]$ vim myscript2.sh
[thaddeus7@hty ~]$ sh myscript2.sh 65
请输入学生成绩：65
```



```
1: myscript2.sh buffers
1 #!/bin/bash
2 #Program: 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。
3
4 read -p "请输入学生成绩：" score
5
6 if [ $score -ge 90 ]; then
7     echo "该成绩为 A 等级"
8 elif [ $score -ge 80 ]; then
9     echo "该成绩 为 B 等级"
10 elif [ $score -ge 70 ]; then
11     echo "该成绩为 C 等级"
12 elif [ $score -ge 60 ]; then
13     echo "该成绩为 D 等级"
14 else
15     echo "该成绩为 E 等级"
16 fi
```

二十、 实验过程分析与讨论

if 条件判断[]内的语法格式:

常用参数:

文件/目录判断:

[-a FILE] 如果 FILE 存在则为真。

[-b FILE] 如果 FILE 存在且是一个块文件则返回为真。

[-c FILE] 如果 FILE 存在且是一个字符文件则返回为真。

[-d FILE] 如果 FILE 存在且是一个目录则返回为真。

[-e FILE] 如果 指定的文件或目录存在时返回为真。

[-f FILE] 如果 FILE 存在且是一个普通文件则返回为真。

数值判断

[INT1 -eq INT2] INT1 和 INT2 两数相等返回为真 ,=

[INT1 -ne INT2] INT1 和 INT2 两数不等返回为真 ,<>

[INT1 -gt INT2] INT1 大于 INT2 返回为真 ,>

[INT1 -ge INT2] INT1 大于等于 INT2 返回为真,>=

[INT1 -lt INT2] INT1 小于 INT2 返回为真 ,<

[INT1 -le INT2] INT1 小于等于 INT2 返回为真,<=

逻辑判断

[!EXPR] 逻辑非, 如果 EXPR 是 false 则返回为真。

[EXPR1 -a EXPR2] 逻辑与, 如果 EXPR1 and EXPR2 全真则返回为真。

[EXPR1 -o EXPR2] 逻辑或, 如果 EXPR1 或者 EXPR2 为真则

返回为真。

[] || [] 用 OR 来合并两个条件

[] && [] 用 AND 来合并两个条件

其他判断

[-t FD] 如果文件描述符 FD （默认值为 1）打开且指向一个终端
则返回为真

[-o optionname] 如果 shell 选项 optionname 开启则返回为

IF 高级特性：

双圆括号(())：表示数学表达式

在判断命令中只允许在比较中进行简单的算术操作，而双圆括号提供更多的数学符号，而且在双圆括号里面的 '>', '<' 号不需要转意。

双方括号[[]]：表示高级字符串处理函数

双方括号中判断命令使用标准的字符串比较，还可以使用匹配模式，从而定义与字符串相匹配的正则表达式。

双括号的作用：

在 shell 中，[\$a != 1 || \$b = 2] 是不允许出，要用 [\$a != 1] || [\$b = 2]，而双括号就可以解决这个问题的，[[\$a != 1 || \$b = 2]]。又比如这个 ["\$a" -lt "\$b"]，也可以改成双括号的形式((" \$a" < "\$b"))

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 循环控制语句		
实验教室	丹青 922	实验日期	2023 年 4 月 12 日
学 号	2021213189	姓 名	贺腾艺
专业班级	计算机科学与技术 06 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*.c 文件复制到指定的目录中(如~/workspace)；

```
[thaddeus7@hty ~]$ vim myscripts3.sh
[thaddeus7@hty ~]$ sh myscripts3.sh
所有.c文件已经成功复制到/home/thaddeus7/workspace/
[thaddeus7@hty ~]$ ls ~/workspace
make-1 test1.c test2.c test3.c test4.c
```

```
1: myscripts3.sh buffers
1 #!/bin/bash#Program: 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(如~/workspace); #History:2023/4/12 hetengyi release1
~
2 target_dir=~/workspace/
3 for file in *.c;do
4     cp "$file" "$target_dir"
5 done
6
7 echo "所有.c文件已经成功复制到$target_dir"
8
```

- 2.编写 Shell 脚本，利用 while 循环求前十个偶数之和，并输出结果

```
[thaddeus7@hty ~]$ vim myscript4.sh
[thaddeus7@hty ~]$ sh myscript4.sh
前10个偶数之和为：110
```

```

1: myscript4.sh
1 #!/bin/bash
2 #Program:编写Shell脚本，利用while循环求前10个偶数之和，并输出结果
3 #History: 2023/4/12 hetengyi release2
4
5 n=1
6 sum=0
7
8 while((n<=20))
9 do
10     if ((n%2 == 0))
11     then
12         sum=$((sum + n))
13     fi
14
15     n=$((n+1))
16 done
17
18 echo "前10个偶数之和为: $sum"

```

3.编写 Shell 脚本，利用 until 循环求 1 到 10 的平方和，并输出结果

```

[thaddeus7@hty ~]$ vim myscript5.sh
[thaddeus7@hty ~]$ sh myscript5.sh
1到10的平方和为： 385

```

```

1: myscript5.sh
1 #!/bin/bash
2 #Program:编写Shell脚本，利用until循环求1-10的平方和，并输出结果
3 #History: 2023/4/12 hetengyi release1
4
5 n=1
6 sum=0
7
8 until ((n>10))
9 do
10     square=$((n*n))
11     sum=$((sum+square))
12
13     n=$((n+1))
14 done
15
16 echo "1到10的平方和为: $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、break2、continue、continue2，并观察四种情况下的实验结果

```
[thaddeus7@hty ~]$ sh myscript6.sh  
a1234  
b1234  
c1234  
d1234
```

```
a1234myscript6.sh:行7: break2: 未找到命令  
5678910  
b1234myscript6.sh:行7: break2: 未找到命令  
5678910  
c1234myscript6.sh:行7: break2: 未找到命令  
5678910  
d1234myscript6.sh:行7: break2: 未找到命令  
5678910
```

```
[thaddeus7@hty ~]$ sh myscript6.sh  
a1234678910  
b1234678910  
c1234678910  
d1234678910
```

```
[thaddeus7@hty ~]$ sh myscript6.sh  
a1234myscript6.sh:行7: continue2: 未找到命令  
5678910  
b1234myscript6.sh:行7: continue2: 未找到命令  
5678910  
c1234myscript6.sh:行7: continue2: 未找到命令  
5678910  
d1234myscript6.sh:行7: continue2: 未找到命令  
5678910
```


二十四、 实验过程分析与讨论

for 循环，while 循环的使用

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

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

for 循环对字符串进行操作，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。

(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。

2、while 循环

也称为前测试循环语句，重复次数是利用一个条件来控制是否继续重复执行这个语句。为了避免死循环，必须保证循环体中包含循环出口条件即表达式存在退出状态为非 0 的情况。

(1) 计数器控制的 while 循环

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

```
sum=0
```

```
i=1
```

```
while(( i <= 100 ))
```

```
do
```

```
    let "sum+=i"
```

```
    let "i += 2"
```

```
done
```

```
echo "sum=$sum"
```

指定了循环的次数 500，初始化计数器值为 1，不断测试循环条件 i 是否小于等于 100。在循环条件中设置了计数器加 2 来计算 1~100 内所有的奇数之和。

(2) 结束标记控制的 while 循环

设置一个特殊的数据值（结束标记）来结束 while 循环。

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

```
echo "Please input the num(1-10) "
```

```
read num
```

```
while [[ "$num" != 4 ]]
```

```
do
```

```
    if [ "$num" -lt 4 ]
```

```
    then
```

```
        echo "Too small. Try again!"
```

```
        read num
```

```
    elif [ "$num" -gt 4 ]
```

```
    then
```

```
        echo "To high. Try again"
```

```
        read num
```

```
    else
```

```
        exit 0
```

```
    fi
```

```
done
```

```
echo "Congratulation, you are right! "
```

(4) 命令行控制的 while 循环

使用命令行来指定输出参数和参数个数，通常与 shift 结合使用，shift 命令使位置变量下移一位（\$2 代替\$1、\$3 代替\$2，并使\$#变量递减），当最后一个参数显示给用户，\$#会等于 0，\$*也等于空。

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

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

```
echo "What you input is: "
```

```
while [[ "$*" != "" ]]  
do  
    echo "$1"  
    shift  
done
```

循环条件可以改写为 `while[["$#" -ne 0]]`，等于 0 时退出 `while` 循环

3、until 循环

`until` 命令和 `while` 命令类似，`while` 能实现的脚本 `until` 同样也可以实现，但区别是 `until` 循环的退出状态是不为 0，退出状态是为 0（与 `while` 刚好相反），即 `while` 循环在条件为真时继续执行循环而 `until` 则在条件为假时执行循环。

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

```
i=0
```

```
until [[ "$i" -gt 5 ]]    #大于 5  
do  
    let "square=i*i"  
    echo "$i * $i = $square"  
    let "i++"  
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 4 月 19 日
学 号	2021213189	姓 名	贺腾艺
专业班级	计算机科学与技术 06 班		
指导教师	卢洋		

东北林业大学
信息与计算机科学技术实验中心

二十五、 实验目的

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

二十六、 实验环境

- (1) 计算机的硬件配置 PC 系列微机。
- (2) 计算机的软件配置 VMware 虚拟机软件及 Ubuntu 虚拟机。

二十七、 实验内容及结果

1. 编写 Shell 脚本，实现一个函数，对两个数的和进行求解，并输出结果

```
[thaddeus7@hty ~]$ vim myscript7.sh
[thaddeus7@hty ~]$ sh myscript7.sh
The sum of 3 and 4 is 7
```

```
#!/bin/bash
#Program:实现一个函数，对两个数的和进行求解
3
4 function addition(){
5     sum=$(( $1 + $2 ))
6     echo "The sum of $1 and $2 is $sum"
7 }
8
9 addition 3 4
~
```

2. 编写 Shell 脚本，在脚本中定义一个递归函数，实现 n 的阶乘的求解

```
1: myscript7.sh
1 #!/bin/bash
2 #Program:定义一个递归函数，实现n的阶乘的求解
3 function factorial() {
4     if [ $1 -eq 1 ]; then
5         # 如果n等于1，则直接返回1
6         echo 1
7     else # 否则，将n与n-1的阶乘相乘
8         n_minus_one=$(( $1 - 1 ))
9         n_minus_one_factorial=$(factorial $n_minus_one)
10        echo $(( $1 * $n_minus_one_factorial ))
11    fi
12 }
13 # 测试递归函数 factorial
14 n=$1
15 [
16 echo "$n的阶乘为: $(factorial $n)"

[thaddeus7@hty ~]$ vim myscript7.sh
[thaddeus7@hty ~]$ sh myscript7.sh 6
6的阶乘为：720
```

3. 运行该程序，并观察程序运行结果， 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
```

```
[thaddeus7@hty ~]$ vim myscript8.sh
[thaddeus7@hty ~]$ sh myscript8.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() {
```

<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> <pre>name</pre> <p>如果传递参数，那么多个参数之间以空格分隔：</p> <pre>name param1 param2 param3</pre> <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>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	丹青 922	实验日期	2023 年 4 月 26 日
学 号	2021213189	姓 名	贺腾艺
专业班级	计算机科学与技术 06 班		
指导教师	卢洋		

东北林业大学
信息与计算机科学技术实验中心

二十九、 实验目的

1. 掌握 sed 基本编辑命令的使用方法；
2. 掌握 sed 与 Shell 变量的交互方法；
3. 掌握 awk 命令的使用方法；
4. 掌握 awk 与 Shell 变量的交互方法；

三十、 实验环境

- (1) 计算机的硬件配置 PC 系列微机。
- (2) 计算机的软件配置 VMware 虚拟机软件及 Ubuntu 虚拟机。

三十一、 实验内容及结果

1、已知 quote.txt 文件内容如下

The honeysuckle band played all night long for only \$90.

It was an evening of splendid music and company.

Too bad the disco floor fell through at 23:10.

The local nurse Miss P.Neave was in attendance.

试编写 sed 命令实现如下功能：

(1.)删除\$符号；

```
[thaddeus7@hty ~]$ 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

```
[thaddeus7@hty ~]$ cat quote.txt | sed -n '/music/p'
It was an evening of splendid music and company.
```

(3.) 在第 4 行后面追加内容：“hello world!”;

```
[thaddeus7@hty ~]$ 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”替换为“Quod”;

```
[thaddeus7@hty ~]$ 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 at 23:10.
Quod local nurse Miss P.Neave was in attendance.
```

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

```
[thaddeus7@hty ~]$ cat quote.txt | sed '3c This is the third line.'
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;

```
[thaddeus7@hty ~]$ 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 r 变量值的行。

```
[thaddeus7@hty ~]$ var=evening
[thaddeus7@hty ~]$ cat quote.txt | sed -n "/$var/p"
It was an evening of splendid music and company.
```

2. 文件 number.txt 的内容如下所示:

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

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

```
[thaddeus7@hty ~]$ cat number.txt | awk '{FS=":"}{print $2}'
:
five
[thaddeus7@hty ~]$ cat number.txt | awk '{FS=" "}{print $2}'
:
:
```

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。

例如: `foo.txt` 内容为:

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

则输出为:

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

```
[thaddeus7@hty ~]$ awk '{for(i=1;i<=NF;i++) if($i%2==0) {print $i; count++}} END{print "The total number of even numbers is: " count}' foo.txt
2
4
46
```

4.脚本的内容如下所示, 试运行该脚本吧, 并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
```

```
[thaddeus7@hty ~]$ cat scripts.sh
#!/bin/bash
read -p "enter search pattern: " pattern
awk "/$pattern/" '{ nmatches++; print } END { print nmatches, "found." }'
info.txt
```

`awk` 中 `"/$pattern/"` 这一部分用双引号括起来, 是为了允许引号内的 `Shell` 变量进行替换
此脚本的作用用于匹配字符串

首先输入你要匹配的字符串, 脚本中指定的文件为 `info.txt`

并在 `info.txt` 文件中查找相应的字符串, 如果能匹配到, 则 `nmatches` 变量就加一, 并在最后输出要匹配字符串出现的位置, 以及出现的次数

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