

# 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 10 日
学 号	2021223153	姓 名	余思婧
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

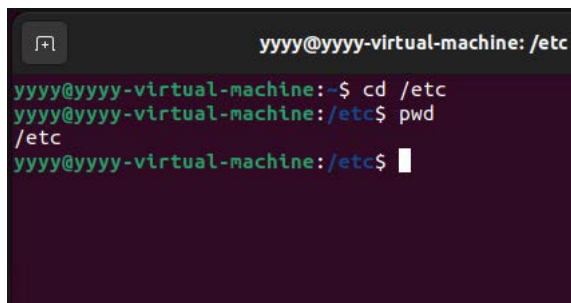
- 1、掌握Linux下文件和目录操作命令：cd、ls、mkdir、rmdir、rm
- 2、掌握Linux下文件信息显示命令：cat、more、head、tail
- 3、掌握Linux下文件复制、删除及移动命令：cp、mv
- 4、掌握 Linux 的文件排序命令：sort

## 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

A terminal window with a dark background. The title bar shows a window icon and the text 'yyyy@yyyy-virtual-machine: /etc'. The terminal content shows the following commands and output:

```
yyyy@yyyy-virtual-machine:~$ cd /etc
yyyy@yyyy-virtual-machine:/etc$ pwd
/etc
yyyy@yyyy-virtual-machine:/etc$
```

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

```
yyyy@yyyy-virtual-machine:~$ cd/home/yyyy
bash: cd/home/yyyy: No such file or directory
yyyy@yyyy-virtual-machine:~$ pwd
/home/yyyy
yyyy@yyyy-virtual-machine:~$ ls -a
.                .bashrc  Documents  Pictures  .sudo_as_admin_successful
..               .cache   Downloads  .profile  Templates
.bash_history    .config  .local     Public    Videos
.bash_logout     Desktop  Music      snap
yyyy@yyyy-virtual-machine:~$
```

3、使用命令创建目录/home/yyyy/linux，然后删除该目录。

```
yyyy@yyyy-virtual-machine:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  snap  Templates  Videos
yyyy@yyyy-virtual-machine:~$ mkdir /home/yyyy/linux
yyyy@yyyy-virtual-machine:~$ ls
Desktop  Downloads  Music  Public  Templates
Documents linux     Pictures snap     Videos
yyyy@yyyy-virtual-machine:~$ pwd
/home/yyyy
yyyy@yyyy-virtual-machine:~$ cd linux
yyyy@yyyy-virtual-machine:~/linux$ rmdir linux
rmdir: failed to remove 'linux': No such file or directory
yyyy@yyyy-virtual-machine:~/linux$ ls
yyyy@yyyy-virtual-machine:~/linux$ cd linux
bash: cd: linux: No such file or directory
yyyy@yyyy-virtual-machine:~/linux$
```

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

```
yyyy@yyyy-virtual-machine:~/linux$ cd /home/yyyy
yyyy@yyyy-virtual-machine:~$ ls
Desktop  Downloads  linux  Pictures  snap  Videos
Documents foo        Music  Public    Templates
yyyy@yyyy-virtual-machine:~$ cat > /home/yyyy/foo << EOF
> "Hello,linux!"
> EOF
yyyy@yyyy-virtual-machine:~$ ls
Desktop  Downloads  linux  Pictures  snap  Videos
Documents foo        Music  Public    Templates
yyyy@yyyy-virtual-machine:~$ cat foo
"Hello,linux!"
yyyy@yyyy-virtual-machine:~$
```

5、使用命令创建目录/home/yyyy/foo.ak，然后将/home/yyyy/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。

```
yyyy@yyyy-virtual-machine:~$ ls
Desktop  Downloads  linux  Pictures  snap      Videos
Documents  foo      Music  Public    Templates
yyyy@yyyy-virtual-machine:~$ mkdir /home/yyyy/foo.bak
yyyy@yyyy-virtual-machine:~$ ls
Desktop  Downloads  foo.bak  Music  Public  Templates
Documents  foo      linux  Pictures  snap      Videos
yyyy@yyyy-virtual-machine:~$ cp /home/yyyy/foo /home/yyyy/foo.bak
yyyy@yyyy-virtual-machine:~$ ls
Desktop  Downloads  foo.bak  Music  Public  Templates
Documents  foo      linux  Pictures  snap      Videos
yyyy@yyyy-virtual-machine:~$ cd /home/yyyy/foo.bak
yyyy@yyyy-virtual-machine:~/foo.bak$ ls -a
.  ..  foo
```

```
yyyy@yyyy-virtual-machine:~/foo.bak$ rm -rf /home/yyyy/foo.bak
yyyy@yyyy-virtual-machine:~/foo.bak$ ls
```

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。

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

yyyy@yyyy-virtual-machine:~/foo.bak$ |
bash: syntax error near unexpected token `|'
yyyy@yyyy-virtual-machine:~/foo.bak$
yyyy@yyyy-virtual-machine:~/foo.bak$ 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
yyyy@yyyy-virtual-machine:~/foo.bak$
```

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-pass
wd
# package, may assume that UIDs less than 100 are unallocated.
--More-- (36%)
```

more

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

google 110 5000

baidu 100 5000

guge 50 3000

sohu 100 4500

```
yyyy@yyyy-virtual-machine:~$ cat > /home/yyyy/bar.txt << EOF
> google 110 5000
> baidu 100 5000
> guge 50 3000
> sohu 100 4500
> EOF
yyyy@yyyy-virtual-machine:~$ ls
bar.txt Desktop Documents Downloads foo linux Music Pictures Public snap Templates Videos
yyyy@yyyy-virtual-machine:~$
```

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

```
yyyy@yyyy-virtual-machine:~$ sort bar.txt -k 1,1
baidu 100 5000
google 110 5000
guge 50 3000
sohu 100 4500
```

(2) 按公司人数排序

```
yyyy@yyyy-virtual-machine:~$ sort bar.txt -k 2n,2
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

```
google 110 5000
yyyy@yyyy-virtual-machine:~$ sort -n -t ' ' bar.txt -k 2,2 -k 3,3
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

```
guge 50 3000
yyyy@yyyy-virtual-machine:~$ sort -n -t ' ' bar.txt -k 3r,3 -k 2,2
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。

```
yyyy@yyyy-virtual-machine:~$ sort -t ' ' bar.txt -k 1.2,1.2
baidu 100 5000
google 110 5000
sohu 100 4500
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 4 月 10 日
学 号	2021223153	姓 名	余思婧
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下符号链接和文件比较命令：ln, comm, diff
- 4、掌握 Linux 的文件权限管理命令：chmod

## 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. 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，并写任意几行内容

```
yyyy@yyyy-virtual-machine:~$ mkdir -p baz
yyyy@yyyy-virtual-machine:~$ cd baz
yyyy@yyyy-virtual-machine:~/baz$ cat > qux << EOF
> hello,world
> this is a linux text
> EOF
```

- (2) 在用户目录下查找文件qux，并显示该文件位置信息

```
yyyy@yyyy-virtual-machine:~/baz$ find -name qux
./qux
```

- (3) 统计文件qux种所包含内容的行数，字数和字节数

```
yyyy@yyyy-virtual-machine:~/baz$ wc qux
 2  6 33 qux
```

- (4) 在用户目录下查找文件qux，并删除该文件

```
yyyy@yyyy-virtual-machine:~/baz$ find -name qux
./qux
yyyy@yyyy-virtual-machine:~/baz$ rm ./qux
yyyy@yyyy-virtual-machine:~/baz$ find -name qux
yyyy@yyyy-virtual-machine:~/baz$
```

- (5) 查看文件夹baz内容，看一下是否删除了文件qux

```
yyyy@yyyy-virtual-machine:~/baz$ find baz
find: 'baz': No such file or directory
```

## 2. 文件打包

- (1) 在用户目录下新建目录path1，在path1下新建文件file1和file2

```
yyyy@yyyy-virtual-machine:~$ mkdir path1
yyyy@yyyy-virtual-machine:~$ cd path1
yyyy@yyyy-virtual-machine:~/path1$ touch file1 file2
yyyy@yyyy-virtual-machine:~/path1$ cd
yyyy@yyyy-virtual-machine:~$ find path1
path1
path1/file1
path1/file2
yyyy@yyyy-virtual-machine:~$
```

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

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

- (3) 在用户目录下新建文件file4

```
yyyy@yyyy-virtual-machine:~$ touch file4
yyyy@yyyy-virtual-machine:~$ find file4
file4
```

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

```
yyyy@yyyy-virtual-machine:~$ tar -cvf package.tar file4 path1
file4
path1/
path1/file1
path1/file2
yyyy@yyyy-virtual-machine:~$ ls
bar.txt  Desktop  Downloads  foo  Music  path1  Pictures  snap  Videos
baz      Documents  file4     linux  package.tar  path2  Public  Templates
```

(5) 查看包package.tar的内容

```
yyyy@yyyy-virtual-machine:~$ tar -tvf package.tar
-rw-rw-r-- yyyy/yyyy      0 2023-05-22 19:21 file4
drwxrwxr-x yyyy/yyyy      0 2023-05-22 19:18 path1/
-rw-rw-r-- yyyy/yyyy      0 2023-05-22 19:18 path1/file1
-rw-rw-r-- yyyy/yyyy      0 2023-05-22 19:18 path1/file2
```

(6) 向包package.tar里添加文件夹path2的内容

```
yyyy@yyyy-virtual-machine:~$ tar -rvf package.tar path2
path2/
path2/file3
yyyy@yyyy-virtual-machine:~$ tar -tvf packages.tar
tar: packages.tar: Cannot open: No such file or directory
tar: Error is not recoverable: exiting now
yyyy@yyyy-virtual-machine:~$ tar -tvf package.tar
-rw-rw-r-- yyyy/yyyy      0 2023-05-22 19:21 file4
drwxrwxr-x yyyy/yyyy      0 2023-05-22 19:18 path1/
-rw-rw-r-- yyyy/yyyy      0 2023-05-22 19:18 path1/file1
-rw-rw-r-- yyyy/yyyy      0 2023-05-22 19:18 path1/file2
drwxrwxr-x yyyy/yyyy      0 2023-05-22 19:20 path2/
-rw-rw-r-- yyyy/yyyy      0 2023-05-22 19:20 path2/file3
```

(7) 将包package.tar复制到用户目录下的新建文件夹path3中

```
yyyy@yyyy-virtual-machine:~$ mkdir path3
yyyy@yyyy-virtual-machine:~$ find path3
path3
yyyy@yyyy-virtual-machine:~$ find -name path3
./path3
yyyy@yyyy-virtual-machine:~$ cp package.tar ./path3
yyyy@yyyy-virtual-machine:~$ find path3
path3
path3/package.tar
yyyy@yyyy-virtual-machine:~$
```

(8) 进入path3文件夹，并还原包package.tar的内容

```

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

```

### 3. 符号链接内容

(1) 新建文件yoo.txt, 内容为123

```

yyyy@yyyy-virtual-machine:~$ cat > yoo.txt << EOF
> 1234
> EOF
yyyy@yyyy-virtual-machine:~$ cat yoo.txt
1234

```

(2) 建立yoo.txt的硬链接文件bar.txt, 并比较bat.txt的内容和yoo.txt是否相同, 要求用comm或diff命令

```

yyyy@yyyy-virtual-machine:~$ ln yoo.txt bar.txt
ln: failed to create hard link 'bar.txt': File exists
yyyy@yyyy-virtual-machine:~$ comm yoo.txt bar.txt
1234
      google 110 5000
comm: file 2 is not in sorted order
      baidu 100 5000
      guge 50 3000
      sohu 100 4500
comm: input is not in sorted order

```

(3) 查看yoo.txt和bar.txt的i节点号(inode)是否相同

```

yyyy@yyyy-virtual-machine:~$ ls -li yoo.txt bar.txt
695902 bar.txt 695916 yoo.txt

```

(4) 修改bar.txt的内容为abc, 然后通过命令判断yoo.txt与bar.txt是否相同



```

5000 100 1500
yyyy@yyyy-virtual-machine:~$ vim bar.txt
yyyy@yyyy-virtual-machine:~$ cat bar.txt
abcgoogle 110 5000
baidu 100 5000
guge 50 3000
sohu 100 4500
yyyy@yyyy-virtual-machine:~$ comm yoo.txt bar.txt
1234
      abcgoogle 110 5000
      baidu 100 5000
      guge 50 3000
      sohu 100 4500
yyyy@yyyy-virtual-machine:~$

```

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

```

yyyy@yyyy-virtual-machine:~$ rm yoo.txt
yyyy@yyyy-virtual-machine:~$ cat bar.txt
abcgoogle 110 5000
baidu 100 5000
guge 50 3000
sohu 100 4500
yyyy@yyyy-virtual-machine:~$ ls -li bar.txt
696674 bar.txt
yyyy@yyyy-virtual-machine:~$

```

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

```

yyyy@yyyy-virtual-machine:~$ ln -s bar.txt baz.txt
yyyy@yyyy-virtual-machine:~$ comm bar.txt baz.txt
      abcgoogle 110 5000
      baidu 100 5000
      guge 50 3000
      sohu 100 4500
yyyy@yyyy-virtual-machine:~$ diff bar.txt baz.txt
yyyy@yyyy-virtual-machine:~$ ls -li bar.txt baz.txt
696674 bar.txt 695916 baz.txt
yyyy@yyyy-virtual-machine:~$

```

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

```
035074 bar.txt 035510 baz.txt
yyyy@yyyy-virtual-machine:~$ rm bar.txt
yyyy@yyyy-virtual-machine:~$ cat baz.txt
cat: baz.txt: No such file or directory
yyyy@yyyy-virtual-machine:~$
```

#### 4. 权限管理

(1) 新建文件qux.txt

```
yyyy@yyyy-virtual-machine:~$ touch qux.txt
yyyy@yyyy-virtual-machine:~$ ls -al qux.txt
-rw-rw-r-- 1 yyyy yyyy 0 5月 23 10:02 qux.txt
yyyy@yyyy-virtual-machine:~$ chmod u+x qux.txt
```

(2) 为文件qux.txt增加执行权限(所有用户都可以执行)

```
yyyy@yyyy-virtual-machine:~$ chmod +x qux.txt
yyyy@yyyy-virtual-machine:~$ ls -al qux.txt
-rwxrwxr-x 1 yyyy yyyy 0 5月 23 10:02 qux.txt
```

#### 四、 实验过程分析与讨论

tar -zcvf 对文件打包并压缩 z 是压缩命令，-zxvf 是解压并拿出文件，-cvf 是打包，-xvf 是解包命令 -rvf 是打包 olderpkg newfile，对文本内容更改用 vim [文件名]，无 vim 命令则 sudo apt install vim

## 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 4 月 14 日
学 号	2021223153	姓 名	余思婧
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 vim 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. 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 ；

```
lqzhou@LAPTOP-BOMFC06B:~$ mkdir workspace1
```

- (2) 进入目录 workspace1 ；

```
lqzhou@LAPTOP-BOMFC06B:~$ cd workspace1
lqzhou@LAPTOP-BOMFC06B:~/workspace1$ pwd
/home/lqzhou/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：

```
yyyy@yyyy-virtual-machine:~/workspace1$ touch test.c
yyyy@yyyy-virtual-machine:~/workspace1$ vim test.c
```

```
#include <stdio.h>
int main()
{
    printf("hello,world!\n");
    return 0;
}
~
~
~
```

(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。

```
yyyy@yyyy-virtual-machine:~/workspace1$ gcc test.c -o test
yyyy@yyyy-virtual-machine:~/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；

```
yyyy@yyyy-virtual-machine:~$ mkdir workspace2
yyyy@yyyy-virtual-machine:~$ find workspace
find: 'workspace': No such file or directory
yyyy@yyyy-virtual-machine:~$ find workspace2
workspace2
```

(2) 进入 workspace2 目录；

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

(3) 使用以下命令：

将文件 /etc/gai.conf 的内容复制到当前目录下的新建文件 gai.conf 中；

```
yyyy@yyyy-virtual-machine:~/workspace2$ cp /etc/gai.conf ./gai.conf
yyyy@yyyy-virtual-machine:~/workspace2$ ls
gai.conf
```

(4) 使用 vim 编辑当前目录下的 gai.conf ；

```

# Configuration for getaddrinfo(3).
#
# So far only configuration for the destination address sorting is needed.
# RFC 3484 governs the sorting. But the RFC also says that system
# administrators should be able to overwrite the defaults. This can be
# achieved here.
#
# All lines have an initial identifier specifying the option followed by
# up to two values. Information specified in this file replaces the
# default information. Complete absence of data of one kind causes the
# appropriate default information to be used. The supported commands in
#
# 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

```

(5) 将光标移到第 18 行;

用 ngg 此时 n=18

```

# 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 in
#
# 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
:8

```

(6) 复制该行内容;

用 yy 复制行

(7) 将光标移到最后一行行首；

用 G 直接跳到末行行首

(8) 粘贴复制行的内容；

用 p 命令即可粘贴

```
#scopev4 ::ffff:127.0.0.0/104      2
# All lines have an initial identifier specifying the option followed by
##scopev4 ::ffff:0.0.0.0/96      14
66 1
```

(9) 撤销第 8 步的动作；

按 u 撤销上一步操作

```
##scopev4 ::ffff:0.0.0.0/96      14
1 line less; before #4  10:38:09
```

(10) 存盘但不退出；

用:w 即可

```
##scopev4 ::ffff:0.0.0.0/96      14
"gai.conf" 65L, 2585B written
```

(11) 将光标移到首行；

用 gg 移到首行

```
Configuration for getaddrinfo(3).
#
# So far only configuration for the destination address
# RFC 3484 governs the sorting.  But the RFC also says
# administrators should be able to overwrite the default
# achieved here.
#
# All lines have an initial identifier specifying the
# up to two values.  Information specified in this file
# default information.  Complete absence of data of
# appropriate default information to be used.  The s
#
# reload <yes|no>
```

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

```
#"hello,this is a vim world!"  
# Configuration for getaddrinfo(3).#  
# So far only configuration for the dest  
# RFC 3484 governs the sorting. But the
```

(13) 删除字符串 "this" ;

可以用 x 删除或者在插入模式下删除

```
#"hello, is a vim world!"  
# Configuration for getaddrinfo(3).#  
# So far only configuration for the des  
# RFC 3484 governs the sorting. But th  
# administrators should be able to over  
# achieved here.
```

(14) 强制退出 vim , 不存盘

```
# Add another  
# RFC 3484.  
#  
#label ::1/128  
:q!
```

#### 四、 实验过程分析与讨论

Vim 命令比较多，各种操作有多种解决方法，需要合理使用较为便捷的快捷方式。

#### 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 4 月 1 日
学 号	2021223153	姓 名	余思婧
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` 。

## 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. 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` ；

```
yyyy@yyyy-virtual-machine:~$ su root
Password:
root@yyyy-virtual-machine:/home/yyyy# cd
root@yyyy-virtual-machine:~# user -c 'bar' -s /bin/sh -d /home/foo -m foo
Command 'user' not found, did you mean:
  command 'iuser' from deb ipmiutil (3.1.8-1)
  command 'fuser' from deb psmisc (23.4-2build3)
  command 'users' from deb coreutils (8.32-4.1ubuntu1)
  command 'userv' from deb userv (1.2.1-beta4)
Try: apt install <deb name>
root@yyyy-virtual-machine:~# useradd -c 'bar' -s /bin/sh -d /home/foo -m foo
root@yyyy-virtual-machine:~# passwd foo
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: password updated successfully
root@yyyy-virtual-machine:~#
```

这里 `useradd` 写错了成了 `user`

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；

```
foo:x:1001:1001:bar:/home/foo:/bin/csh
root@yyyy-virtual-machine:~# usermod -u 2000 -s /bin/csh foo
```

改后：

```
foo:x:2000:1001:bar:/home/foo:/bin/csh
```

3. 从用户 foo 切换到 root；

```
yyyy-virtual-machine:~# su root
```

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@yyyy-virtual-machine:~# userdel -r foo
userdel: foo mail spool (/var/mail/foo) not found
root@yyyy-virtual-machine:~# ls
root  snap
```

5. 使用命令 newusers 批量创建用户，并使用命令 chpasswd 为这些批量创建的用户设置密码（密码也需要批量设置），查看 /etc/passwd 文件检查用户是否创建成功；

```

root@yyyy-virtual-machine:~# cat > user.txt << EOF
> user001::600:100:user:/home/user001:/bin/bash
> user002::601:100:user:/home/user002:/bin/bash
> user003::602:100:user:/home/user003:/bin/bash
> user004::604:100:user:/home/user004:/bin/bash
> cat > user.txt << EOF
user001::600:100:user:/home/user001:/bin/bash
user002::601:100:user:/home/user002:/bin/bash
user003::602:100:user:/home/user003:/bin/bash
user004::603:100:user:/home/user004:/bin/bash
> user005::604:100:user:/home/user005:/bin/bash
> user006::605:100:user:/home/user006:/bin/bash
> EOF
root@yyyy-virtual-machine:~# cat user.txt
user001::600:100:user:/home/user001:/bin/bash
user002::601:100:user:/home/user002:/bin/bash
user003::602:100:user:/home/user003:/bin/bash
user004::604:100:user:/home/user004:/bin/bash
cat > user.txt << EOF
user001::600:100:user:/home/user001:/bin/bash
user002::601:100:user:/home/user002:/bin/bash
user003::602:100:user:/home/user003:/bin/bash
user004::603:100:user:/home/user004:/bin/bash
user005::604:100:user:/home/user005:/bin/bash
user006::605:100:user:/home/user006:/bin/bash
root@yyyy-virtual-machine:~#

```

先建立一个 user.txt 保存要创建的用户

```
root@yyyy-virtual-machine:~# newusers < user.txt
```

取消 shadow password 功能并创建密码文件

```

root@yyyy-virtual-machine:~# pwunconv
root@yyyy-virtual-machine:~# cat > passwd.txt << EOF
> user001:123456
> user002:123456
> user003:123456
> user004:123456
> user005:123456
> user006:123456
> EOF

```

写入并编码

```
root@yyyy-virtual-machine:~# chpasswd < passwd.txt
```

```
root@yyyy-virtual-machine:~# pwconv
```

全都建立完毕，并且密码隐藏为 x

6. 创建用户组 group1 ，并在创建时设置其 GID 为 3000 ；

7. 在用户组 group1 中添加两个之前批量创建的用户；

```
root@yyyy-virtual-machine:~# groupadd -g 3000 group1
root@yyyy-virtual-machine:~# usermod -g 3000 user001
root@yyyy-virtual-machine:~# usermod -g 3000 user002
```

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。

```
yyyy@yyyy-virtual-machine:~$ su root
Password:
root@yyyy-virtual-machine:/home/yyyy# cd
root@yyyy-virtual-machine:~# su user001
$ sudo vim /etc/shadow
[sudo] password for user001:
user001 is not in the sudoers file. This incident will be reported.
```

```
# User privilege specification
root    ALL=(ALL:ALL) ALL
user001 ALL=(ALL:ALL) ALL
# Members of the admin group may gain root privileges
%admin   ALL=(ALL) ALL
```

#### 四、 实验过程分析与讨论

对于用户权限的了解有一定缺失。

`sudo` 命令的使用需要在 `sudoers` 添加权限。

#### 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 22 日
学 号	2021223153	姓 名	余思婧
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。

## 二、 实验环境

- (1) 计算机的硬件配置 PC 系列微机。
- (2) 计算机的软件配置通过 wsl 安装的 ubuntu 虚拟机。

## 三、 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）；

```
yyyy@yyyy-virtual-machine:~$ mkdir scripts
yyyy@yyyy-virtual-machine:~$ cd scripts
yyyy@yyyy-virtual-machine:~/scripts$ foo=200
yyyy@yyyy-virtual-machine:~/scripts$ 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）；

```
yyyy@yyyy-virtual-machine:~/scripts$ bar=100
yyyy@yyyy-virtual-machine:~/scripts$ test $bar -gt 150
yyyy@yyyy-virtual-machine:~/scripts$ 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 ）；

```
#!/bin/bash
hn=$(hostname)
da=$(date)
echo $hn
echo $da
~
```

```
yyyy@yyyy-virtual-machine:~/scripts$ vim l3.sh
```

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

```
#!/bin/bash
read -p "Input number:" num
num1=`expr $num % 10`
#echo $num1
num2=`expr $num / 10 % 10`
#echo $num2
num3=`expr $num / 100 % 10`
#echo $num3
num13=`expr $num1 \* $num1 \* $num1`
num23=`expr $num2 \* $num2 \* $num2`
num33=`expr $num3 \* $num3 \* $num3`
num4=`expr $num13 + $num23 + $num33`
if (( $num4 == $num))
then
    echo "true"
else
    echo "false"
fi
~
~
~
```

```
yyyy@yyyy-virtual-machine:~/scripts$ bash l4.sh
Input number:153
true
yyyy@yyyy-virtual-machine:~/scripts$ bash.l4.sh
bash.l4.sh: command not found
yyyy@yyyy-virtual-machine:~/scripts$ bash l4.sh
Input number:124
false
yyyy@yyyy-virtual-machine:~/scripts$ bash l4.sh
Input number:370
true
```

5. 创建一个Shell程序，输入 3 个参数，计算 3 个输入变量的



和并输出；

```
#!/bin/bash
echo 'sum = ' `expr $1 + $2 + $3`
```

```
yyyy@yyyy-virtual-machine:~/scripts$ bash l5.sh 1 2 3
sum = 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 B ， 70-80为 C ， 60-70 为 D ，小于 60 分为 E 。

```
#!/bin/bash
read -p "Please input your grade:" grade
if (($grade < 60))
then
    echo "level is E"
elif (($grade < 70))
then
    echo "level is D"
elif (($grade < 80))
then
    echo "level is C"
elif (($grade < 90))
then
    echo "B"
else echo "level is A"
fi
~
~
```

```
yyyy@yyyy-virtual-machine:~/scripts$ bash l6.sh
Please input your grade:88
B
yyyy@yyyy-virtual-machine:~/scripts$ bash l6.sh
Please input your grade:66
level is D
yyyy@yyyy-virtual-machine:~/scripts$ bash l6.sh
Please input your grade:55
level is E
```



#### 四、 实验过程分析与讨论

发现每个脚本文件都要写一个`#!/bin/bash` 我觉得太累了，写一个 shell 生成 shell 脚本的脚本，感觉速度要快一点，体验到自动化的快乐。

原本以为可以直接通过运算符计算，但是搜索后发现，需要 `expr`

#### 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 23 日
学 号	2021223153	姓 名	余思婧
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. 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、 实验环境

- (1) 计算机的硬件配置 PC 系列微机。
- (2) 计算机的软件配置通过 wsl 安装的 ubuntu 虚拟机。

## 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 ）；

```
#!/bin/bash
for file in `find *.c`
do
    mv $file ./workspace
done
```

```
yyyy@yyyy-virtual-machine:~/scripts$ vim 11.c
yyyy@yyyy-virtual-machine:~/scripts$ vim 22.c
yyyy@yyyy-virtual-machine:~/scripts$ vim 33.c
yyyy@yyyy-virtual-machine:~/scripts$ bash l.sh
yyyy@yyyy-virtual-machine:~/scripts$ ls
l1.sh l3.sh l4.sh l5.sh l6.sh l.sh workspace
yyyy@yyyy-virtual-machine:~/scripts$ ls workspace/
11.c 1.c 22.c 2.c 33.c 3.c
```

2. 编写 Shell 脚本，利用 while 循环求前 10 个偶数之和，

并输出结果；

```
#!/bin/bash
count=0
sum=0
num=2
while (($count<10))
do
    sum=`expr $sum + $num`
    num=`expr $num + 2`
    let "count++"
done
echo $sum
~
~
~
~
```

```
yyyy@yyyy-virtual-machine:~/scripts$ vim l1.sh
yyyy@yyyy-virtual-machine:~/scripts$ bash l1.sh
110
```

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

```
#!/bin/bash
count=1
sum=0
until (($count == 11))
do
    sum=`expr $sum + $count \* $count`
    let "count++"
done
echo $sum
~
~
```

```
110
yyyy@yyyy-virtual-machine:~/scripts$ vim l3.sh
yyyy@yyyy-virtual-machine:~/scripts$ bash l3.sh
385
```

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 、 continue 2 ，并观察四种情况下的实验结果。

break 时

```
yyyy@yyyy-virtual-machine:~/scripts$ vim l7.sh
yyyy@yyyy-virtual-machine:~/scripts$ bash l7.sh
a1234
b1234
c1234
d1234
```

Break 2 时

```
yyyy@yyyy-virtual-machine:~/scripts$ bash l7.sh
a1234yyyy@yyyy-virtual-machine:~/scripts$
```

直接跳过 2 个循环了

Continue

```
a1234yyyy@yyyy-virtual-machine:~/scripts$ vim l7.sh
yyyy@yyyy-virtual-machine:~/scripts$ bash l7.sh
a1234678910
b1234678910
c1234678910
d1234678910
```

Continue 2

```
yyyy@yyyy-virtual-machine:~/scripts$ bash l7.sh  
a1234b1234c1234d1234yyyy@yyyy-virtual-machine:~/scripts$
```

#### 四、 实验过程分析与讨论

对 until 和 while 有一定基础了解了。

Break continue 后面 2 的意思是应该是执行 2 遍？

## 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 26 日
学 号	2021223153	姓 名	余思婧
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、 实验环境

- (1) 计算机的硬件配置 PC 系列微机。
- (2) 计算机的软件配置通过 wsl 安装的 ubuntu 虚拟机。

## 三、 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；

```
yyyy@yyyy-virtual-machine:~/scripts$ bash l1.sh
Please input first number:1230
Please input second number:4312
Sum of two nubern is 5542
```

```
#!/bin/bash
function AddTwoNum()
{
    s=`expr $1 + $2`
    echo "Sum of two nubern is $s"
}
read -p "Please input first number:" num1
read -p "Please input second number:" num1
AddTwoNum $num1 $num2
```

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；

```
#!/bin/bash
f()
{
    if [ $k -eq 0 ]
    then
        echo 1
    else
        let sum=$1*f `f ["$1"-1]`
        echo $sum
    fi
}
read a
f $a
```

3. 试运行该程序，并观察程序运行结果，理解函数嵌套的含义。

```
#!/bin/bash

function first() {
    function second() {
        function third() {
            echo "-3- here is in the third func."
        }
        echo "-2- here is in the second func."
        third
    }
    echo "-1- here is in the first func."
    second
}
echo "starting..."
first
```

```
yyyy@yyy-virtual-machine:~/scripts$ bash l8.sh
starting...
-1- here is in the first func.
-2- here is in the second func.
-3- here is in the third func.
```

```
#!/bin/bash
function first(){
    function second(){
        function third(){
            echo "-3- here is in the third func."
        }
        echo "-2- here is in the second func."
        third
    }
    echo "-1- here is in the first func."
    second
}
echo "string..."
first
```

#### 四、 实验过程分析与讨论

理解一些 shell 和子 shell 的关系，可以让 echo 不显示在当前 shell 的屏幕上。这对于只能返回 0-255 的 return 来说，帮助很大，能过通过递归返回很大的数值。

## 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	2021223153	姓 名	余思婧
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 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。

## 二、 实验环境

- (1) 计算机的硬件配置 PC 系列微机。
- (2) 计算机的软件配置通过 wsl 安装的 ubuntu 虚拟机。

## 三、 实验内容及结果

1. 文件 quote.txt 的内容如下所示：

试使用 sed 命令实现如下功能：

```
yyyy@yyyy-virtual-machine:~/scripts$ cat >quote .txt << EOF
cat >quote .txt << EOF
The honeysuckle band played all night long for only $90.
It as 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.
> EOF
```

- (1) 删除 \$ 符号；

```
yyyy@yyyy-virtual-machine:~/scripts$ cat quote.txt | sed 's/\$/g'
The honeysuckle band playednall niaght long for only 0.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；

```
the local nurse Miss P.Neave was in attendance.
yyyy@yyyy-virtual-machine:~/scripts$ nl quote.txt | sed -n '/music/p'
2 It was an evening of splendid music and company.
```

- (3) 在第 4 行后面追加内容： "hello world!" ；

```
yyyy@yyyy-virtual-machine:~/scripts$ sed -e 4a'hello,world!' quote.txt
The honeysuckle band played all night long for only 0.
It was an evening of splendid music and company.
Too bad the disco floor fell through at23:10.
The local nurse Miss P.Neave was in attendance.
hello,world!
yyyy@yyyy-virtual-machine:~/scripts$
```

(4) 将文本 “The” 替换为 “Quod” ；

```
yyyy@yyyy-virtual-machine:~/scripts$ sed 's/The/Quod/g' quote.txt
Quod honeysuckle band played all night long for only 0.
It was an evening of splendid music and company.
Too bad the disco floor fell through at23:10.
Quod local nurse Miss P.Neave was in attendance.
```

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

```
yyyy@yyyy-virtual-machine:~/scripts$ sed '3c This is the third line\.' quote.txt
This is the third line.
The honeysuckle band played all night long for only 0.
It was an evening of splendid music and company.
The local nurse Miss P.Neave was in attendance.
```

(6) 删除第 2 行内容；

```
yyyy@yyyy-virtual-machine:~/scripts$ nl quote.txt | sed '2d'
1 The honeysuckle band played all night long for only 0.
3 Too bad the disco floor fell through at23:10.
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。

```
yyyy@yyyy-virtual-machine:~/scripts$ var='evening'
yyyy@yyyy-virtual-machine:~/scripts$ echo $var
evening
```

```
yyyy@yyyy-virtual-machine:~/scripts$ set | nl - | sed -n '/evening/p'
90 _=evening
95 var=evening
```

2. 文件 numbers.txt 的内容如下所示：



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

注：每个冒号前后都有空格。

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

```
yyyy@yyyy-virtual-machine:~/scripts$ cat > numbers.txt << EOF
> one : two : three
> four : five : six
> EOF
```

```
yyyy@yyyy-virtual-machine:~/scripts$ awk -F ' ' '{print $2}' numbers.txt
:
:
```

```
yyyy@yyyy-virtual-machine:~/scripts$ awk -F ':' '{print $2}' numbers.txt
two
five
```

3. 已知文件 `foo.txt` 中存储的都是数字，且每行都包含 3 个数字，数字之前以空格作为分隔符。试找出 `foo.txt` 中的所有偶数进行打印，并输出偶数的个数

要求：判断每行的 3 个数字是否为偶数时用循环结果，即要求程序里包含循环和分支结构。

```

read -p "please input filename:" filename
echo "even:"
export num=0
while read line
do
    a1=`echo $line | awk '{print $1}'`
    a2=`echo $line | awk '{print $2}'`
    a3=`echo $line | awk '{print $3}'`
    if ($a1%2 == 0)
    then
        let num=$num+1
        echo $a1
    fi
    if (($a2%2 == 0))
    then
        let num=$num+1
        echo $a2
    fi
    if (($a3%2 == 0))
    then
        let num=$num+1
        echo $a3
    fi
done < $filename
echo "numbers:"
echo $num

```

```

yyyy@yyyy-virtual-machine:~/scripts$ bash l9.sh
please input filename:foo.txt
even:
2
4
46
l9.sh: line 15: ((: %2 == 0: syntax error: operand expected (error token is
"%2 == 0")
l9.sh: line 20: ((: %2 == 0: syntax error: operand expected (error token is
"%2 == 0")
cat
l9.sh: line 15: ((: >%2 == 0: syntax error: operand expected (error token is
">%2 == 0")
l9.sh: line 20: ((: foo.txt%2 == 0: syntax error: invalid arithmetic operato
r (error token is ".txt%2 == 0")
2
4
46
6
numbers:
8

```

#### 4. 脚本的内容如下所示:

```

#!/bin/bash

read -p "enter search pattern: " pattern

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

```

试运行该脚本，并理解该脚本实现的功能。

```
yyyy@yyyy-virtual-machine:~/scripts$ cat > info.txt << EOF
> this is a test
> in my lib
> nefu 123
> test over
> EOF
```

```
yyyy@yyyy-virtual-machine:~/scripts$ bash ll.sh
enter serach pattern:test
found.
```

#### 四、 实验过程分析与讨论

理解一些 shell 和子 shell 的关系，可以让 echo 不显示在当前 shell 的屏幕上。这对于只能返回 0-255 的 return 来说，帮助很大，能过通过递归返回很大的数值。

## 五、指导教师意见

指导教师签字：卢洋