shell脚本（三）

一、使用for循环语句

1.批量创建用户

2.批量删除用户

3.for另一种语法

1.倒计时：

2.创建用户：

3.随机数---设置密码：

4.查看主机存活状态（面试题）

4.九九乘法表

二、使用while循环语句

1.while输出1-100

2.批量创建/删除用户

1.批量创建用户

2.批量删除用户

3.while/for查看文件

1.for查看文件

2.while查看文件

4.猜商品价格的小游戏

5.石头剪刀布游戏

三、case分支语句

1.管理文件的脚本

shell脚本（三）

用于批量重复操作

一、使用for循环语句

1.批量创建用户

方法一：[root@localhost ~]# vim cu.sh

```
1 #!/bin/bash
2 for i in {1..20}
3 do
4   useradd user$i
5   echo "123456" | passwd --stdin huu$i
6 done
```

```
[root@localhost ~]# for i in ${seq 20}
> do
> echo $i yuyuyuyu....
> sleep 1
> done
```

方法二：

[root@localhost ~]# vim userlist.txt

```
1 zhangsan
2 lisi
```

```
3 wangwu
4 john
5 jenny
6 amy
7 teddy
8 pj
9 gabe
[root@localhost ~]# vim user_add.sh
1 #!/bin/bash
2 for i in $(cat userlist.txt)
3 do
4   useradd $i
5   echo "123456" | passwd --stdin $i
6   chage -d 0 $i
7 done
[root@localhost ~]# vim user_del.sh
1 #!/bin/bash
2 for i in $(cat userlist.txt)
3 do
4   userdel -r $i
7 done
```

2.批量删除用户

```
[root@localhost ~]# vim cu.sh
1 #!/bin/bash
2 for i in user{1..20}
3 do
4   userdel $i
5  # echo "123456" | passwd --stdin huu$i
6 done
```

输出1-100之间的偶数

方法一：[root@localhost ~]# vim cu.sh

```
1 #!/bin/bash
2 #偶数
3 for i in {1..100}
4 do
```

```
5 #  [ $(expr $i % 2) -eq 0 ] && echo "偶数为: $i"
6   [ $[$i%2] -eq 0 ] && echo "偶数为: $i"
7 done
```

方法二：[root@localhost ~]# vim cu.sh

```
13 #输出1-50，乘以2
14 for i in {1..50}
15 do
16   echo "偶数: $(expr $i \* 2)"
17 done
```

方法三：[root@localhost ~]# vim cu.sh

```
19 num=0
20 for i in {1..50}
21 do
22   num=$(expr $num + 2)
23   echo $num
24 done
```

3.for另一种语法

1.倒计时:

```
[root@localhost ~]# for ((i=1;i<=9;i++))
> do
> echo $i
>sleep 1
> done
```

2.创建用户:

```
[root@localhost ~]# for ((i=1;i<=20;i++))
> do
> name=user$i
> useradd $name
> echo "123456" | passwd --stdin $name  &> /dev/null
> done
```

3.随机数---设置密码:

```
[root@localhost ~]# date +%N
```

创建用户及设置密码：

```
[root@localhost ~]# vim users20.sh
1 #!/bin/bash
```

```
 2 for i in user{1..20}
 3 do
 4   ps=$(date +%N | head -c 8)
 5   useradd $i
 6   echo "$ps" | passwd --stdin $i
 7   echo "$i $ps" > /tmp/c
 8 done
```

删除用户：

```
[root@localhost ~]# vim users_del20.sh
 1 #!/bin/bash
 2 for i in user{1..20}
 3 do
 4   userdel -r  $i
 5 done
```

4.查看主机存活状态（面试题）

1.查看一台主机的存活状态

批量操作脚本：（vim编辑，可视块）

Ctrl+v --> 选中多行首字母 --> shift+i --> 输入内容 --> 输入内容

Ctrl+v --> 选中多行的删除部分 --> 按x

```
[root@localhost ~]# vim hostcheck.sh
 1 #!/bin/bash
 2 ping -c 3 -i 0.1  -W 3 $1 &> /dev/null
 3 if [ $? -eq 0 ]
 4 then
 5     echo "host $1 is up"
 6 else
 7     echo "host $1 is down"
 8 fi
[root@localhost ~]# bash hostcheck.sh 192.168.200.110
        host 192.168.200.110 is up
```

2.查看一个网段主机的存活状态

```
[root@localhost ~]# vim hostcheck.sh
#!/bin/bash
for i in 192.168.200.{1..254}
do
```

```
                ping -c 3 -i 0.1        -W 3 $i &> /dev/null
                if [ $? -eq 0 ]
                then
                                echo "host $i is up"
                else
                                echo "host $i is down"
                fi
    done
[root@localhost ~]# bash hostcheck.sh
```

4.九九乘法表

方法一：

```
[root@localhost ~]# vim 99.sh
     1 #!/bin/bash
     2 for i in {1..9}
     3 do
     4   for j in {1..9}
     5   do
     6     z=$(expr $j \* $i)
     7     [ $j -le $i ] && echo -n ""$j"x"$i"="$z "
     8   done
     9   echo -e            #换行
    10 done
[root@localhost ~]# bash 99.sh
    1x1=1
    1x2=2 2x2=4
    1x3=3 2x3=6 3x3=9
    1x4=4 2x4=8 3x4=12 4x4=16
    1x5=5 2x5=10 3x5=15 4x5=20 5x5=25
    1x6=6 2x6=12 3x6=18 4x6=24 5x6=30 6x6=36
    1x7=7 2x7=14 3x7=21 4x7=28 5x7=35 6x7=42 7x7=49
    1x8=8 2x8=16 3x8=24 4x8=32 5x8=40 6x8=48 7x8=56 8x8=64
    1x9=9 2x9=18 3x9=27 4x9=36 5x9=45 6x9=54 7x9=63 8x9=72 9x9=81
```

方法二：

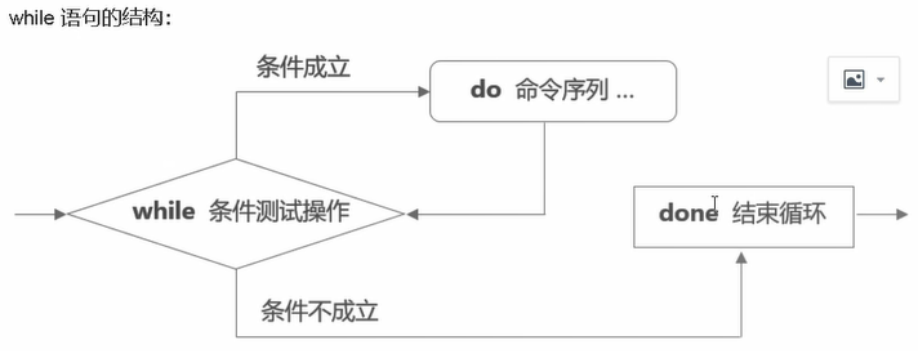
```
[root@localhost ~]# vim 99.sh
```

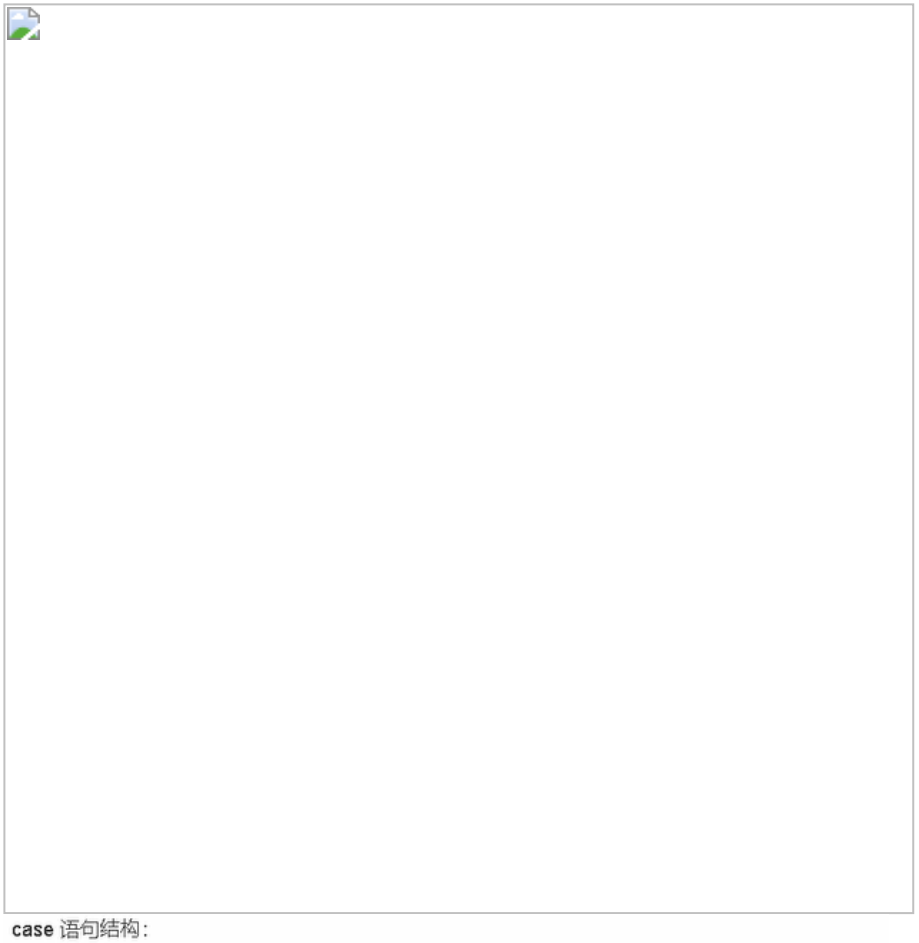
```
 1 #!/bin/bash
12 for i in {1..9}
13 do
14   for j in {1..9}
15   do
16     [ $j -le $i ] && echo -n "${j}x${i}=$(($j*$i)) "
17   done
18   echo -e
19 done
[root@localhost ~]# bash 99.sh
```

方法三：

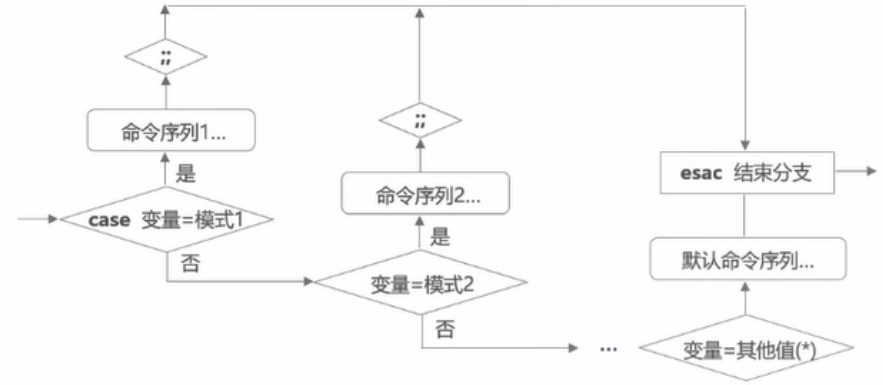
```
[root@localhost ~]# vim 99.sh
 1 #!/bin/bash
22 for ((i=1;i<=9;i++))
23 do
24   for ((j=1;j<=$i;j++))
25   do
26     echo -n "${j}x${i}=$(($j*$i)) "
27   done
28   echo
[root@localhost ~]# bash 99.sh
```

二、使用while循环语句





case 语句结构：



true　　永真

fault　　假

1.while输出1-100

```
[root@localhost ~]# vim while.sh
      1 #!/bin/bash
      2 i=1
      3 while [ $i -le  100 ]
      4 do
      5   echo "$i"
```

```
      6   let i++
      7 done
[root@localhost ~]# bash while.sh
```

let i++ 等同于 i='expr $i + 1' 避免死循环

2.批量创建/删除用户

1.批量创建用户

```
[root@localhost ~]# vim while.sh
      1 #!/bin/bash
      2 i=1
      3 while [ $i -le  20 ]
      4 do
      5   useradd user$i
      6   echo "123456" | passwd --stdin user$i
      7   let i++
      8 done
```

2.批量删除用户

```
[root@localhost ~]# vim while.sh
      1 #!/bin/bash
      2 i=1
      3 while [ $i -le  20 ]
      4 do
      5   userdel -r user$i
      6   #userdel -r stu$i
      7   let i++
      8 done
```

3.while/for查看文件

1.for查看文件

```
[root@localhost ~]# for i in $(cat userlist.txt)
> do
> echo $i
> sleep 1
> done
zhangsan
```

```
lisi
wangwu
john
jenny
amy
teddy
pj
gabe
```

2.while查看文件

```
[root@localhost ~]# cat userlist.txt | while read line
> do
> echo $line
> done
zhangsan
lisi
wangwu
john
jenny
amy
teddy
pj
gabe
```

注意：使用while的时候，一行里有空格的时候需要定义两个变量

有几个空格设置几个变量

例如：[root@localhost ~]# x x | while read a b

```
[root@localhost ~]# cat /root/file.txt | while read a b
> do
> echo $a $b
> done
```

4.猜商品价格的小游戏

```
[root@localhost ~]# vim game.sh
     1 #!/bin/bash
     2 #猜测商品价格的小游戏
     3 echo "这是一个猜商品价格的小游戏，猜猜看"
```

```
 4
 5 #电脑出价
 6 jiage=$(expr $RANDOM % 1000 + 1)
 7 cishu=0
 8
 9 echo "$jiage" > /tmp/b                    #bug 游戏外挂，哈哈
10 while true
11 do
12     read -p "请输入你猜测的价格（1-1000）：" cai
13     let cishu++
14     if [ $cai -eq $jiage ]
15     then
16         echo "恭喜你猜对了，商品价格就是：$jiage"
17         echo "共猜了$jiage次"
18     elif [ $cai -gt $jiage ]
19     then
20         echo "给高了，太有钱了"
21     else
22         echo "太抠了，多出点"
23     fi
24 done
```

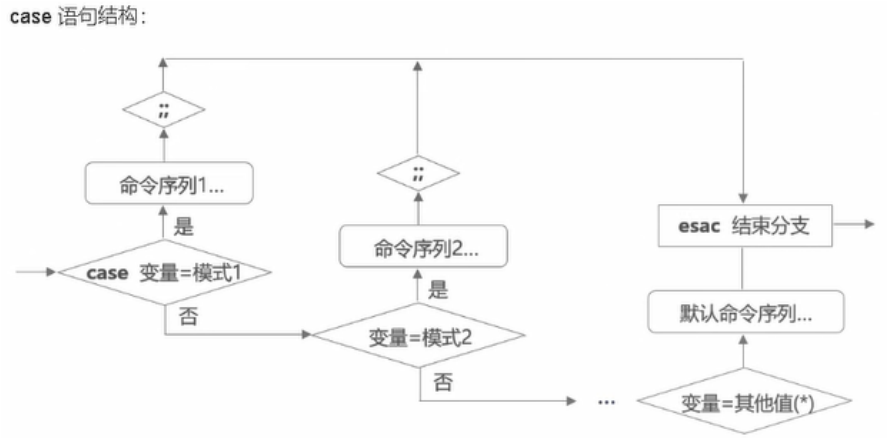
5.石头剪刀布游戏

```
[root@localhost ~]# vim game2.sh
 1 #!/bin/bash
 2 #石头剪刀布游戏
 3 echo "这是一个石头剪刀布的小游戏，小朋友可以来玩呀~"
 4 echo "石头0 剪刀1 布2"
 5
 6 pc=$(expr $RANDOM % 3)
 7 echo "$pc" >> /tmp/a                #bug在这里哟，啊哈哈
 8 while :
 9 do
10     #pc=$(expr $RANDOM % 3)
11     read -p "输入你的选择：" cai
```

```
 12     if [ $pc -eq 0 -a $cai -eq 2 ] || [ $pc -eq 1 -a $cai -eq 0 ] || [
$pc -eq  2 -a $cai -eq 1 ]
 13     then
 14         echo "恭喜你，电脑都猜不过你~"
 15     elif [ $pc -eq $cai ]
 16     then
 17         echo "平局"
 18     else
 19         echo "很遗憾，你猜错了，祝你下次好运~"
 20     fi
 21 done
[root@localhost ~]# bash game2.sh
```

三、case分支语句

多分支时使用case



1.管理文件的脚本

```
[root@localhost ~]# vim file.sh
 1 #!/bin/bash
 2 case $1 in
 3 create)
 4       touch /tmp/{1..10}.txt
 5 ;;
 6 remove)
 7       rm -rf /tmp/{1..10}.txt
 8 ;;
 9 list)
```

```
     10     ls -l /tmp/{1..10}.txt
     11 ;;
     12 *)
     13     echo "用法：$0 {create|remove|list}"
     14 esac
[root@localhost ~]# bash file.sh create
[root@localhost ~]# bash file.sh remove
[root@localhost ~]# bash file.sh list
```

2.判断字符（字母、数字、其他）

```
[root@localhost ~]# vim key.sh
      1 #!/bin/bash
      2 while true
      3 do
      4  read -p "输入一个字符：" k
      5  case $k in
      6  [0-9])
      7       echo "数字"
      8  ;;
      9  [a-zA-Z])
     10       echo "字母"
     11  ;;
     12  *)
     13       echo "特殊符号"
     14  ;;
     15  esac
     16 done
[root@localhost ~]# bash key.sh
```

？？？？？怎么退出脚本呢，，，，，待解决；有了*，就很难识别别的了，可能不行

系统脚本，是最好的学习教材

```
[root@localhost ~]# /etc/init.d/network
用法：/etc/init.d/network {start|stop|status|restart|reload|force-reload}
[root@localhost ~]# vim /etc/init.d/network
```

rsync远程同步服务----脚本

```
[root@localhost ~]# vim /etc/init.d/rsyncd
    #!/bin/bash
    #chkconfig:2345 80 20        #如果系统启动级别是2345，启动服务是第80个启动，第20个关闭
    #description:rsync server

    start () {
    netstat -lnpt | grep -q :873              #看看有没有端口
    [ $? -ne 0 ] && rsync --daemon             #启动服务
    }

    stop () {
    netstat -lnpt | grep -q :873
    [ $? -eq 0 ] && kill $(cat /var/run/rsyncd.pid) && rm -rf /var/run/rsyncd.pid
    }
    #查看PID号所在文件，杀进程并删文件

    status () {
    netstat -lnpt | grep -q :873
    [ $? -eq 0 ] && echo "Rsync Daemon is running."
    }


    case $1 in
    start)
    start
    ;;
    stop)
    stop
    ;;
    restart|reload)
    $0 stop
    $0 start
```

```
    ;;
    status)
    status
    ;;
    *)
    echo  "用法:  $0  {start|stop|restart|reload|status}"
    esac

[root@localhost  ~]#  chmod  +x  /etc/init.d/rsyncd
[root@localhost  ~]#  chkconfig  --add  rsyncd
[root@localhost  ~]#  systemctl  start  rsyncd
```

四、正则表达式

awk sed grep ----- 三剑客

文本处理工具:

grep 匹配，查询

sed 编辑（增加，删除，修改）

awk 文本格式化（字符串提取）

注意：

linux中正则一般以行为单位处理文件

alias grep='grep --color=auto'

注意字符集，LANG=C

通配符:

正则表达式与命令行中使用的通配符有本质区别

* 任意长度任意字符串

? 单个任意字符串

grep命令参数

-v或--revert-match　　反转查找

grep -v "nologin" /etc/passwd

-E或--extended-regexp　　将范本样式为延伸的普通表示法来使用

grep -E "qemu|ntp" /etc/passwd

-i或--ignore-case　　忽略字符大小写的差别

-o　　只输出匹配的内容 only

[root@localhost ~]# grep -Eo "qemu|ntp" /etc/passwd

-n或--line-number　　在显示符合范本样式的那一列之前，标示出该列的列数编号

[root@localhost ~]# grep -En "qemu|ntp" /etc/passwd

-q或--quiet或--silent　　不显示任何信息(静默输出)

[root@localhost ~]# netstat -lnpt | grep -q :22

-w或--word-regexp　　将过滤条件当成单词匹配

grep "bin" /etc/passwd

grep -w "bin" /etc/passwd

-c或--count　　计算匹配行的数

grep -wc "bin" /etc/passwd

grep -w "bin" /etc/passwd | wc -l

基础正则表达式

- **^：匹配行首位置，注意匹配的是位置，不是字符**
- \:转义字符
- **$：匹配行尾位置，注意匹配的是位置，不是字符**
- **^$：它表示匹配空行**
- **.*：匹配任意长度的任意字符，但不能匹配换行符**
- **. ：匹配任意单个字符，但不能匹配换行符\n**
- ***：匹配前面那个字符0或多次**

- [list] ：
  - `[abcd...]`：匹配中括号内的任意一个字符
  - `[^abcd...]`：拒绝匹配中括号内的任意字符
  - `[a-z]`：匹配字母a到z
  - `[A-Z]`：匹配字母A到Z
  - `[0-9]`：匹配0-9，也就是匹配数字
- **\<**：匹配单词开头处的空字符
- **\>**：匹配单词结尾处的空字符
- **\{M,N\}**：匹配前面那个字符至少M，最多N次
- **\{M,\}**：匹配前面那个字符至少M次，最多无限制
- **\{,N\}**：匹配前面那个字符最多N次(最少当然是0次)。注意，perl正则不支持这种方式
- **\{M\}**：匹配前面那个字符正好M次

```
[root@localhost ~]# vim test.txt
[root@localhost ~]# cat test.txt
    aabcc
    aa.cc
    aa cc
[root@localhost ~]# grep "a.c" test.txt
    aabcc
    aa.cc
    aa cc
[root@localhost ~]# grep "a\.c" test.txt
    aa.cc
[root@localhost ~]# grep "a..c" test.txt
    aabcc
    aa.cc
    aa cc
```

```
[root@localhost ~]# grep "$" test1.txt
        gd
        god
        good
        goood
        gooood
        gold
        glad
        gaad
        abcDfg
        food
        601151272
        HELLO
        010-6666888
        0666-5666888
        IP 192.168.200.108
        IP 173.16.16.1
        pay $180
[root@localhost ~]# grep '\$' test1.txt
        pay $180
[root@localhost ~]# awk '/\$/{print $0}' test1.txt
pay $180
[root@localhost ~]# sed -n '/\$/p' test1.txt
pay $180

[root@localhost ~]# grep "^[a-z]" test1.txt
        gd
        god
        good
        goood
        gooood
        gold
        glad
        gaad
        abcDfg
```

```
        food
        pay $180

[root@localhost ~]# grep "[0-9]$" test1.txt
        601151272
        010-6666888
        0666-5666888
        IP 192.168.200.108
        IP 173.16.16.1
        pay $180


[root@localhost ~]# grep "go.d" test1.txt
    good
    gold
[root@localhost ~]# grep "go[a-z]d" test1.txt
    good
    gold
[root@localhost ~]# grep "go[a-zA-Z]d" test1.txt
    good
    gold
[root@localhost ~]# grep "go..d" test1.txt
    goood
[root@localhost ~]# grep "go*d" test1.txt
    gd
    god
    good
    goood
    gooood
[root@localhost ~]# grep "go.*d" test1.txt
    god
    good
    goood
    gooood
    gold
```

```
[root@localhost ~]# grep "go[a-z]d" test1.txt
    good
    gold
[root@localhost ~]# grep "go[^a-z]d" test1.txt

go{3}d = gooood
go\{3\}d

[root@localhost ~]# grep "g[la]ad" test1.txt
    glad
    gaad
[root@localhost ~]# grep "[a-z]" test1.txt
    gd
    god
    good
    goood
    gooood
    gold
    glad
    gaad
    abcDfg
    food
    pay $180
[root@localhost ~]# grep "[^a-z]" test1.txt
    abcDfg
    601151272
    HELLO
    010-6666888
    0666-5666888
    IP 192.168.200.108
    IP 173.16.16.1
    pay $180
```

这个有点疑问。。。。。

```
[root@localhost ~]# grep "[0-9]\{3,4\}-[0-9]\{7,8\}" test1.txt
```

```
    010-6666888
    0666-5666888
[root@localhost ~]# grep -E "[0-9]{3,4}-[0-9]{7,8}" test1.txt
    010-6666888
    0666-5666888

[root@localhost ~]# grep -E "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}"
test1.txt
    IP 192.168.200.108
    IP 173.16.16.1
[root@localhost ~]# grep -Eo "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}"
test1.txt
    192.168.200.108
    173.16.16.1
[root@localhost ~]# ifconfig | grep -E "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-
9]{1,3}"
        inet 192.168.200.110  netmask 255.255.255.0  broadcast 192.168.200.255
        inet 127.0.0.1  netmask 255.0.0.0
        inet 192.168.122.1  netmask 255.255.255.0  broadcast 192.168.122.255
[root@localhost ~]# ifconfig | grep -Eo "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-
9]{1,3}"
    192.168.200.110
    255.255.255.0
    192.168.200.255
    127.0.0.1
    255.0.0.0
    192.168.122.1
    255.255.255.0
    192.168.122.255

   grep "bin" /etc/passwd              #过滤bin
   grep "\<bin" /etc/passwd           #过滤以bin开头的
   grep "\<bin>\" /etc/passwd         #过滤只有bin
   grep "bin\>" /etc/passwd            #过滤以bin结尾的
```