shell脚本应用（二）

shell脚本编写思路：

1. 按照命令执行顺序编写文件
2. 注意交互式命令（进行转换）
3. 注意命令执行输出

4. 文件上添加执行权限（可选）

```
bash  -x    xxx.sh   方便脚本调试
```

- 单一的顺序结构式脚本过于机械化，不够智能，难以处理复杂的任务
- 使用if语句使脚本具有判断能力，根据不同的条件来完成不同的任务
- 程序执行方式：
    1. 顺序执行
    2. 选择执行
    3. 循环执行

一、条件测试

1、条件测试

根据命令执行的返回值来判断 $?

0	真	执行成功	True
非0	假	执行失败	False

2、测试方法：

test 命令测试的两种形式：

(1) test 条件表达式

```
[root@localhost ~]# test -d /boots
[root@localhost ~]# echo $?
1
[root@localhost ~]# test -d /boot
[root@localhost ~]# echo $?
0
```

(2) [条件表达式]

```
#中括号于字符串中间至少包含一个空格（应用更加普遍）
[root@localhost ~]# [ -d /boot ]
[root@localhost ~]# echo $?
0
```

3、条件测试的分类：

文件测试，在数值比较，字符串比较，多个条件的逻辑测试

(1) 文件测试

根据给定的路径名称，判断对应的是文件还是目录，判断文件是否可读，可写，可执行等。

1）格式：[操作符 文件或目录]

2）常用的测试操作符：

1. -d：测从应用的试是否为目录（Directory）
   从应用的角度去思考，多想想别的用法，不要局限于一种思路
2. -e：测试目录或文件是否存在（Exist）
3. -f：测试是否为文件（File）
4. -r：测试当前用户是否可读（Read）
5. -w：测试当前用户是否可写（Write）
6. -x：测试当前用户是否可执行（eXcute）
7. -s 存在且字节数大于0

示例：

判断光盘是否挂载

```
[root@localhost ~]# [ -d /media/cdrom ]
[root@localhost ~]# echo $?
0
[root@localhost ~]# [ -d /media/cdrom/Packages ]
```

```
[root@localhost ~]# echo $?
0
[root@localhost ~]# ls /media/cdrom
CentOS_BuildTag  GPL       LiveOS    RPM-GPG-KEY-CentOS-7
EFI              images    Packages  RPM-GPG-KEY-CentOS-Testing-7
EULA             isolinux  repodata  TRANS.TBL

[ -d /media/cdrom ] || mkdir /media/cdrom      || 逻辑或
```

配置本地yum仓库

```
[root@localhost ~]# vim create_yum.sh
[root@localhost ~]# bash create_yum.sh
```

本地YUM仓库配置完毕

```
[root@localhost ~]# [ -x create_yum.sh ]     #查看是否有执行权限
[root@localhost ~]# echo $?
1
[root@localhost ~]# [ -x create_yum.sh ] || chmod u+x create_yum.sh  #添加执行权限
[root@localhost ~]# ls -l
总用量 9004
-rw-r--r--  1 root root      40 3月  18 17:43 1
-rw-r--r--  1 root root      68 3月  18 17:48 2
-rw-------. 1 root root    1712 3月  13 23:15 anaconda-ks.cfg
-rw-r--r--  1 root root     926 3月  19 17:01 create_lv.sh
-rwxr--r--          1  root     root            537 3月     19   23:53 create_yum.sh
```

（2）整数值比较

根据给定的两个整数值，判断第一个数是否大于，小于，等于第二个数

1）格式：[整数 1 操作符 整数 2]

2）常用的测试操作符：

1. **-eq：等于（Equal）**
2. **-ne：不等于（Not Equal）**
3. **-gt：大于（Greater Than）**
4. **-lt：小于（Lesser Than）**
5. **-ge：大于或等于（Greater or Equal）**
6. **-le：小于或等于（Lesser or Equal）**

例如：

```
[root@localhost ~]# a=99
[root@localhost ~]# [ $a -eq 100 ]
[root@localhost ~]# echo $?
                    1
[root@localhost ~]# unum=$(who | wc -l)
[root@localhost ~]# [ $unum -gt 3 ] && echo "用户登录告警！用户已超出3个用户！"
              用户登录告警！用户已超出3个用户！

[root@localhost ~]# df -Th | awk '/\/$/{print $6}' | awk -F% '{print $1}'
              9
[root@localhost ~]# du=$(df -Th | awk '/\/$/{print $6}' | awk -F% '{print $1}')
[root@localhost ~]# [ $du -gt 8 ] && echo "磁盘报警！"
              磁盘报警！
```

（3）字符串比较

检查用户输入的字符是否符合需求

在shell中一个等号和两个等号一样

格式：

[字符串 1 = 字符串 2] 字符串内容相同

[字符串 1 != 字符串 2] 字符串内容不同

[-z 字符串] 字符串内容为空 一般用于测试变量值

```
[root@localhost ~]# [ -z $HOSTNAME ] || echo $HOSTNAME
                    localhost.localdomain
```

（4）逻辑测试

测试两个条件或多个条件之间的依赖关系

[表达式 1] 操作符 [表达式 2] …

命令 1 操作符 命令 2 …

常用的测试操作符：

1. -a 或&&：逻辑与“而且”的意思
   需要满足所有前提条件，才会执行下一步操作。
2. -o 或||：逻辑或，“或者”的意思
3. !：逻辑否

例：

```
[root@localhost ~]# [ -d /root ]
[root@localhost ~]# echo $?
0
[root@localhost ~]# [ ! -d /root ]
[root@localhost ~]# echo $?
1
[root@localhost ~]#  ! [ -d /root ]
[root@localhost ~]# echo $?
1
```

二、if条件语句

1、if 单分支语句

用的较少，一般用&&连接

```
if [  条件测试操作  ]
then  命令序列
fi
```

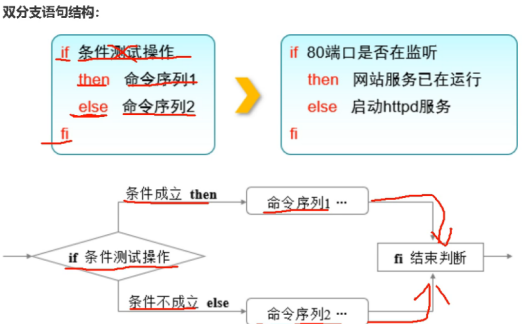
2、if 多分支语句

```
if [  条件测试操作  ]
    then  命令序列 1
    else  命令序列 2
fi
```

示例：

```
[root@localhost ~]# vim calc.sh
#!/bin/bash
X=1
Y=2
Z=$(expr $X + $Y)
if [ $Z -eq 3 ]
then
    echo -n "计算结果等于:"
```

```
    echo $(expr $X + $Y)
fi
[root@localhost ~]# bash calc.sh
计算结果等于:3
```



3.案例

1.示例：去厕所

```
[root@localhost ~]# vim wc.sh
    #! /bin/bash
    while true
    do
    read -p "请输入您的性别(man|woman):" xb
    if [ $xb = "man" ]
    then
        echo "请去左边"
    elif [ $xb = "woman" ]
    then
        echo "请去右边"
    else
    #echo "输入错误，请重新输入"
    /bin/bash ./wc.sh
    fi
    done
[root@localhost ~]# bash wc.sh
    请输入您的性别(man|woman):man
    请去左边
    请输入您的性别(man|woman):woman
    请去右边
    请输入您的性别(man|woman):ss
    请输入您的性别(man|woman):yy
    请输入您的性别(man|woman):man
    请去左边
```

2.示例：查看网络状态

```
    [root@localhost ~]# vim net.sh
    [root@localhost ~]# bash net.sh
        httpd-2.4.6-90.el7.centos.x86_64
        已安装
        网站服务已停止，尝试启动.....
    ========================================
    [root@localhost ~]# vim net.sh
        #!/bin/bash
        #查看httpd包
```

```
rpm -q httpd &> /dev/null
if [ $? -eq 0 ]
then
        echo "已安装"
else
        yum -y install httpd
fi
#查看网络状态
netstat -lnpt | grep :80
if [ $? -eq 0 ]
then
        echo "网站服务正常运行"
else
        echo "网站服务已停止，尝试启动....."
    systemctl restart httpd
    if [ $? -eq 0 ]
    then
                echo "服务启动成功"
     else
                echo "服务启动失败"
     fi
fi
```

```
[root@localhost ~]# bash net.sh
   已安装
   tcp6       0      0 :::80                   :::*                    LISTEN      2779/httpd
   网站服务正常运行
====================================
[root@localhost ~]# systemctl stop httpd
[root@localhost ~]# bash net.sh
          已安装
          网站服务已停止，尝试启动.....
          服务启动成功
```

注意脚本的层次感，主次语句的缩进（set tabstop=2），整齐美观的脚本更好

永久设置vim编辑：

/etc/vimrc　　是系统范围的初始化配置

~/.vimrc　　个人的vim初始化配置

3.示例：新年压岁钱

```
[root@localhost ~]# vim year.sh
 1 #!/bin/bash
 2 echo "新年压岁钱:有(1),无(2)"
 3 read -p "你有没有收到压岁钱(1/2)?" status
 4 if [ $status -eq 1 ]
 5 then
```

```
     6      echo "看起来有收获啊"
     7      read -p "输入你得到的压岁钱？" m
     8      if [ $m -gt 500 ]
     9      then
    10          echo "收获相当不错了！"
    11      else
    12          echo "有就行啦"
    13      fi
    14 else
    15      echo "让你长大，没了吧"
    16 fi
[root@localhost ~]# bash year.sh
    新年压岁钱:有(1),无(2)
    你有没有收到压岁钱(1/2)?1
    看起来有收获啊
    输入你得到的压岁钱？900
    收获相当不错了！
[root@localhost ~]# bash year.sh
    新年压岁钱:有(1),无(2)
    你有没有收到压岁钱(1/2)?2
    让你长大，没了吧
```

4.示例：考试成绩----只要整数

```
[root@localhost ~]# vim score.sh
     1 #!/bin/bash
     2 while true
     3 do
     4 read -p "请输入您的分数(1-100):" sc
     5 if [ $sc -gt 85 ] && [ $sc -le 100 ]
     6 then
     7     echo "您的成绩优异，请继续保持！"
     8 elif [ $sc -gt 60 ] && [ $sc -le 85 ]
     9 then
    10     echo "您的成绩合格，请再接再厉！"
    11 elif [ $sc -gt 0 ] && [ $sc -le 60 ]
    12 then
    13     echo "您的成绩不合格，请继续努力！"
    14 else
    15     /bin/bash ./score.sh
    16 fi
    17 done
[root@localhost ~]# bash score.sh
    请输入您的分数(1-100):95
    您的成绩优异，请继续保持！
    请输入您的分数(1-100):77
    您的成绩合格，请再接再厉！
    请输入您的分数(1-100):50
    您的成绩不合格，请继续努力！
```

脚本（if条件嵌套练习）

1.可以事先规定范围（更人性化一点，但又显啰嗦）

2.也可以直接在最后的条件中声明分数不合格

```
[root@localhost ~]# vim score.sh
     1 #!/bin/bash
     2 while true
     3 do
     4 read -p "请输入您的分数(1-100):" sc                    #注意位置哦~
     5 if [ $sc -ge 0 ] && [ $sc -le 100 ]              #判断分数值是否在1~100之间
     6 then
     7     if [ $sc -gt 85 ] && [ $sc -le 100 ]
     8     then
     9         echo "您的成绩优异，请继续保持！"
    10     elif [ $sc -gt 60 ] && [ $sc -le 85 ]
    11     then
    12         echo "您的成绩合格，请再接再厉！"
    13     elif [ $sc -gt 0 ] && [ $sc -le 60 ]
    14     then
    15         echo "您的成绩不合格，请继续努力！"
    16     else
    17         /bin/bash ./score.sh              #数据丢失这里容易出问题，记得修改
    18     fi
    19 else
    20     echo "输入的分数无效，请输入0~100之间的有效分数"
    21 fi
    22 done
[root@localhost ~]# bash score.sh
    请输入您的分数(1-100):200
    输入的分数无效，请输入0~100之间的有效分数
    请输入您的分数(1-100):50
    您的成绩不合格，请继续努力!
    请输入您的分数(1-100):70
    您的成绩合格，请再接再厉!
    请输入您的分数(1-100):90
    您的成绩优异，请继续保持!
```

5.示例：vsftp服务

```
[root@localhost ~]# vim vsftp.sh
    1 #!/bin/bash
    2 rpm -q vsftp &> /dev/null
    3 [ $? -eq 0 ] || yum -y install vsftpd &> /dev/null
    4 #查看网络端口状态
    5 systemctl status vsftpd &> /dev/null
    6 if [ $? -eq 0 ]
    7 then
    8     #echo "监听地址:$(netstat -anpt | grep vsftpd | awk '{print $4}')"
```

```
 9      #echo "进程PID号:$(pgrep -x vsftpd)"
```

注：-x, --exact match exactly 完全匹配

```
10      echo "监听地址:$(netstat -anpt | awk '/vsftpd/{print $4}')"
11      echo "进程PID号:$(netstat -anpt | awk '/vsftpd/{print $7}' | awk -F/ '{print $1}')"
12  else
13      echo "警告：vsftpd服务不可用！"
14      systemctl start vsftpd
15      if [ $? -eq 0 ]
16      then
17          echo "服务运行成功！"
18          /bin/bash ./vsftp.sh
19      else
20          echo "服务启动失败！"
21      fi
22  fi
[root@localhost ~]# systemctl stop vsftpd
[root@localhost ~]# bash vsftp.sh
    警告：vsftpd服务不可用!
    服务运行成功!
    监听地址::::21
    进程PID号:4494
```

6.示例：查看软件包是否安装且实现自动安装

```
[root@localhost ~]# vim pac.sh
 1 #!/bin/bash
 2 read -p "请输入您需要安装的软件包名:" pak
 3 rpm -q $pak &> /dev/null
 4 if [ $? -eq 0 ]
 5 then
 6     echo "已安装$pak包"
 7 else
 8     echo "未安装，尝试自动安装....."
 9     yum -y install $pak &> /dev/null
10     if [ $? -eq 0 ]
11     then
12         echo "安装完成！"
13     else
14         echo "安装失败！"
15     fi
16 fi
[root@localhost ~]# bash pac.sh
    请输入您需要安装的软件包名:rasqal
    未安装，尝试自动安装.....
    安装完成!
```

7.示例：内核版本

```
[root@localhost ~]# vim ker.sh
 1 #!/bin/bash
 2 Mnum=$(uname -r | awk -F. '{print $1}')
```

```
3 Nnum=$(uname -r | awk -F. '{print $2}')
4 if [ $Mnum -gt 2 ] && [ $Nnum -gt 4 ]
5 then
6     echo "$Mnum.$Nnum"
7 else
8     echo "内核版本太低无法继续！"
9 fi
[root@localhost ~]# bash ker.sh
                    3.10
```

8.ping检查主机是否存活！！！！！

- **-c：表示的是ping的次数（linux系统下并不会像windows一样ping四次后停止），后面的3为ping三次后终止。**
- **-i：表示的是两次ping访问之间的时间间隔，0.2参数表示的是间隔0.2s**
- **-W：表示的是定义等待超时的时间，3表示的是超过三秒钟就定义为ping不通**
- **$1：为输入的参数**
- **&> /dev/null：表示的是用完的参数自动存入一个没有回收功能的垃圾箱**
- **$?：参数表示的是若前面的语句执行成功，则会返回0，若执行不成功，则会返回非0数据。**
- **-eq ： 为前者是否等于后者**

```
[root@localhost ~]# vim ping.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 ping.sh 192.168.200.110
          host 192.168.200.110 is up
```

测试网段的脚本：

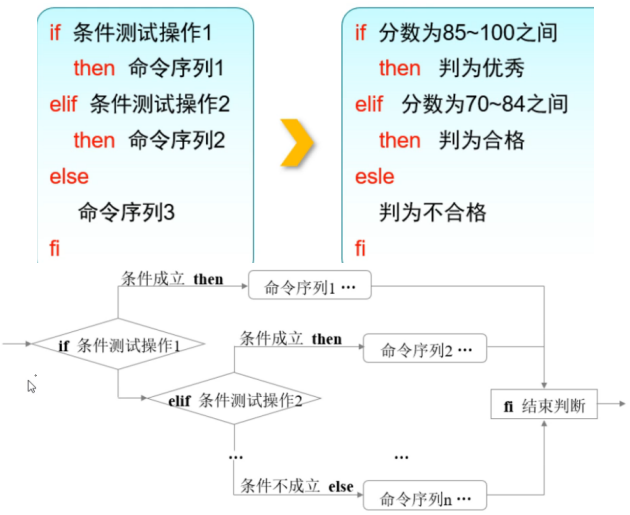
```
vim test.txt
#!/bin/bash
#
#by skyfans
#seq命令用于产生从某个数到另外一个数之间的所有整数。
#分解这个组合：“>/dev/null 2>&1” 为五部分。
#1：> 代表重定向到哪里，例如：echo "123" > /home/123.txt
#2：/dev/null 代表空设备文件
#3：2> 表示stderr标准错误
#4：& 表示等同于的意思，2>&1，表示2的输出重定向等同于1
#5：1 表示stdout标准输出，系统默认值是1，所以">/dev/null"等同于 "1>/dev/null"
#
#1>/dev/null ：首先表示标准输出重定向到空设备文件，也就是不输出任何信息到终端，
说白了就是不显示任何信息。
```

#2>&1 ：接着，标准错误输出重定向 到 标准输出，因为之前标准输出已经重定向到了空>设备文件，所以标准错误输出也重定向到空设备文件。

```
#
for ip in `seq 1 255`
  do
   {
  ping -c 1 172.17.99.$ip >/dev/null 2>&1
    if [ $? -eq 0 ];then
     echo 172.17.99.$ip UP
    else
     echo 172.17.99.$ip DOWN
fi
}&
done
wait
```

4、if 多分支语句结构

```
if    [   条件测试操作 1   ]
then  命令序列 1
elif  条件测试操作 2
then  命令序列 2
else
命令序列 3
fi
```



5.DNS综合管理脚本

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

```
#!/bin/bash

cat << EOF
/==============================\\
|         DNS服务管理脚本         |
|        1、首次部署DNS服务       |
|        2、更换DNS域名           |
|        3、添加DNS域名           |
|        4、测试DNS解析           |
|        5、退出此脚本            |
--------------------------------

EOF
while true
do
#提示用户输入选项
read -p "请输入您的选项(1~5):" num

#判断具体的选项操作
if [ $num -eq 1 ]
then
#安装软件
#pm -q bind bind-utils bind-libs &> /dev/null
[ $? -eq 0 ] || yum -y install bind bind-utils bind-libs

#提示用户输入域名及设置IP变量
read -p "请输入您的域名(example:red.com):" dn
IP=$(ifconfig ens33 | awk '/inet /{print $2}')

#修改主配置文件
cat << EOF > /etc/named.conf
options {
    directory "/var/named";
};
zone "$dn"  IN  {
    type master;
    file "$dn.zheng";
};
EOF

```

```
#修改正向区域解析文件
cat << EOF > /var/named/$dn.zheng
\$TTL 86400
@    IN    SOA    $dn.    admin.$dn.    (
            2020320
            3H
            15M
            1W
            1D
)
     IN    NS    ns.$dn.
ns   IN    A     $IP
www  IN    A     $IP
EOF

#改组
chgrp named /var/named/$dn.zheng

#修改/etc/resolv.conf
echo "nameserver $IP" > /etc/resolv.conf

#启动服务
systemctl enable named
systemctl restart named
systemctl status named
if [ $? -eq 0 ]
then echo "服务正常启动!"
else
    echo "服务部署失败!"
fi

#更换域名
elif [ $num -eq 2 ]
then
read -p "请输入需更换域名:" dn

#过滤原域名及IP
old_dn=$(awk -F\" '/zone/{print $2}' /etc/named.conf)
IP=$(ifconfig ens33 | awk 'NR==2{print $2}')

```

```
#修改主配置文件替换旧域名
sed -i "1,7 s#$old_dn#$dn#" /etc/named.conf

#删除冗余文件
cd /var/named/
rm -rf $(ls *.zheng |grep -v "$dn.zheng")

#修改正向解析区域文件
cat << EOF > /var/named/$dn.zheng
\$TTL 86400
@    IN    SOA    $dn.    admin.$dn.    (
            2020320
            3H
            15M
            1W
            1D
)
      IN    NS    ns.$dn.
ns    IN    A     $IP
www   IN    A     $IP
EOF
chgrp named /var/named/$dn.zheng
#启动服务
systemctl restart named
if [ $? -eq 0 ]
then
    echo "服务正常启动"
else
    echo "服务部署失败"
fi

#添加域名
elif [ $num -eq 3 ]
then
read -p "请输入您需要添加的域名:" dn
IP=$(ifconfig ens33 | awk 'NR==2{print $2}')

#修改主配置文件
cat << EOF >> /etc/named.conf
zone "$dn" IN {
```

```
122     type master;
123     file "$dn.zheng";
124 };
125 EOF
126 #修改正向解析区域文件
127 cat << EOF > /var/named/$dn.zheng
128 \$TTL 86400
129 @    IN   SOA   $dn.    admin.$dn.    (
130           2020320
131           3H
132           15M
133           1W
134           1D
135 )
136      IN    NS   ns.$dn.
137 ns   IN    A    $IP
138 www  IN    A    $IP
139 EOF
140
141 #启动服务
142 systemctl restart named
143 if [ $? -eq 0 ]
144 then echo "服务正常启动!"
145 else echo "服务部署失败!"
146 fi
147
148 #测试
149 elif [ $num -eq 4 ]
150 then
151 read -p "请输入您需要测试的域名:" dn
152 IP=$(ifconfig ens33 | awk 'NR==2{print $2}')
153 echo "nameserver $IP" >  /etc/resolv.conf
154 echo
155 echo "开始测试 $dn 域名解析"
156 echo "-----------------------------------"
157 echo
158 nslookup www.$dn
159 else
160   read -p "尊敬的用户~您确定要放弃选择，退出本脚本吗?(yes/no):" set
161   if [ $set = "yes" ] || [ $set = "y" ]
```

```
162   then
163       echo
164       echo "friends,goodbye,see you again!"
165       echo
166       exit
167   else
168       echo "欢迎回来~"
169   fi
170 fi
171 done
```

exit [number] exit 0 捕获的返回值 即 echo $? 的返回值

6.邮件告警

需求描述

- 编写监控脚本sysmon.sh，存在异常时邮件告警
- 监控CPU使用率、内存使用率、根分区的占用率

  mpstat、free、
- 百分比精确到个位，如7%、12%
- 出现以下情况时告警：磁盘占用率超过90%、CPU使用率超过80%、内存使用率超过90%
- 结合计划任务，每半小时检查一次

实现思路

- 使用df、mpstat、free等命令提取各种监控指标
- 将各指标与正常值进行比较，保存异常情况
- 检查异常记录，若存在则发送告警邮件
- 设置crontab任务，定期调用sysmon.sh脚本
- [root@localhost ~]# watch -n 1 'date'

```
[root@localhost ~]# vim  sysmon.sh
    #!/bin/bash
    #  提取性能监控指标（磁盘占用、CPU使用、内存使用）
        DUG=$(df  -h  |  grep  "/$"  |  awk  '{print  $5}'  |  awk  -F%  '{print  $1}')
        CUG=$(expr  100  -  $(mpstat  |  tail  -1  |  awk  '{print  $NF}'  |  awk  F.  '{print  $1}'))
        #CUG=$(expr 100 - $(mpstat | tail -1 | awk '{print $NF}' | awk -F. '{print $1}'))
        MUG=$(expr  $(expr  $(free    |  awk  '/Mem:/{print  $3}')  \*  100  /  $(free    |  awk  '/Mem:/{print  $2}')))
    #  设置告警日志文件、告警邮箱
        ALOG="/tmp/alert.txt"
    #  判断是否记录告警
        if  [  $DUG  -gt  90  ]
        then
        echo  "磁盘占用率：$DUG  %"  >>  $ALOG
        fi
        if  [  $CUG  -gt  80  ]
        then
        echo  "CPU使用率：$CUG  %"  >>  $ALOG
```

```
fi
if  [  $MUG  -gt  90  ]
then
echo  "内存使用率：$MUG  %"  >>  $ALOG
fi
```

三、网易邮箱配置

1.电脑登录网易邮箱配置

安装配置邮件服务器

授权码：MYSBSFFHVIGQIMAD

```
[root@server  ~]#  yum  install  -y  mailx  dos2unix
                                linux的邮件客户端
[root@server  ~]#  mailx  -V
              12.5  7/5/10
[root@server  ~]#  vim  /etc/mail.rc
```

//在此文件末尾添加，
指定接收邮件邮箱地址，
指定邮箱服务器地址，
指定接收邮件邮箱地址的授权码，并非163邮箱的密码，而是授权码，
登陆163网易邮箱地址后，
--设置--开启smtp和pop3--点击客户端生成授权码--使用此授权码进行指定到此配置文件；

```
set  from=xxxx@163.com  smtp=smtp.163.com
set  smtp-auth-user=xxxx@163.com  smtp-auth-password=上面生成的授权码
set  smtp-auth=login

[root@localhost ~]# route add default gw 192.168.200.110
[root@localhost ~]# vim /etc/resolv.conf
                                1 nameserver 202.106.0.20
[root@server  ~]#  echo  "test  mail  from  rainy lrf"|  mails"test  mail"  liruifangtote@163.com
  //发送测试内容
```

邮箱查看

编写报警邮件脚本，

调整mail命令语法位置，

使用dos2unix命令转换字符，

避免收到邮件不显示正常邮件内容而出现ATT00001.bin的错误；

2.改进的脚本：可发送邮件

```
[root@localhost ~]# vim rate.sh
 1 #!/bin/bash
 2 #export.UTF-8
 3 #CUG=$(expr 100 - $(mpstat | tail -1 | awk '{print $NF}' | awk -F. '{print $1}'))
 4 CUG=$(expr 100 - $(mpstat | awk '/all/{print $NF}' | awk -F. '{print $1}'))
 5 MUG=$(expr $(free | awk '/Mem:/{print $3}') \* 100 / $(free | awk '/Mem:/{print $2}'))
 6 DUG=$(df -Th | awk '/\/$/{print $6}' | awk -F% '{print $1}')
 7
 8 #设置告警日志文件、告警邮箱
 9 ALOG="/tmp/alert.txt"
10 > $ALOG
11 #判断是否记录告警
12 if [ $CUG -gt 0 ]
13 then
14     echo "CPU使用率:$DUG%" >> $ALOG
15 fi
16 if [ $MUG -gt 1 ]
17 then
18     echo "内存使用率:$MUG%" >> $ALOG
19 fi
20 if [ $DUG -gt 1 ]
21 then
22     echo "磁盘占用率:$DUG%" >> $ALOG
23 fi
24
25
26 #发送告警文件
27 #标题=ip地址
28 bt=$(ifconfig ens33 | awk 'NR==2{print $2}')
27 #标题=ip地址
28 bt=$(ifconfig ens33 | awk 'NR==2{print $2}')
29 #收件人
30 sjr="liruifangtote@163.com"
31 #判断文件是否存在；转格式，防止产生乱码
32 [ -f $ALOG ] && /usr/bin/dos2unix -k $ALOG
```

```
 33 #-s 指定标题（IP地址），收件人，文件信息
 34 /bin/mail -s "$bt" "$sjr" < $ALOG
[root@localhost ~]# bash +x rate.sh
```

3.写入日志执行

```
[root@localhost ~]# crontab -l
[root@localhost ~]# crontab -e
       * *   * * * /bin/bash /root/rate.sh
[root@localhost ~]# watch -n 1 'date'              #每隔一秒执行一次
[root@localhost ~]# systemctl start crond
[root@localhost ~]# tail /var/log/cron
 26 #发送告警文件
 27 #标题=ip地址
 28 export bt=$(ifconfig ens33 | awk 'NR==2{print $2}')
 29 #收件人
 30 export sjr="liruifangtote@163.com"
 31 #判断文件是否存在；转格式，防止产生乱码
 32 [ -f $ALOG ] && /usr/bin/dos2unix -k $ALOG
 33 #-s 指定标题（IP地址），收件人，文件信息
 34 /bin/mail -s "主机报警" "$sjr" < $ALOG
```

```
[root@master ~]# cat check_hard.sh
#!/bin/bash
#export.UTF-8
    #  提取性能监控指标（磁盘占用、CPU使用、内存使用）
    DUG=$(df -h | grep "/$" | awk '{print $5}' | awk -F% '{print $1}')
    CUG=$(expr 100 - $(mpstat | tail -1 | awk '{print $NF}' | awk -F. '{print $1}'))
    MUG=$(expr $(expr $(free | awk '/Mem:/{print $3}') \* 100 / $(free | awk '/Mem:/{print $2}')))
#  设置告警日志文件、告警邮箱
    ALOG="/tmp/alert.txt"
    > $ALOG
#  判断是否记录告警
    if [ $DUG -gt 5 ]
    then
    echo "磁盘占用率：$DUG%" >> $ALOG
    fi
    if [ $CUG -gt 5 ]
    then
    echo "CPU使用率：$CUG%" >> $ALOG
    fi
    if [ $MUG -gt 5 ]
    then
    echo "内存使用率：$MUG%" >> $ALOG
    fi
#  发送告警邮件
```

```
bt=$(ifconfig  ens32  |  awk  'NR==2  {print  $2}')
sjr="xxxx@163.com"
[  -f  $ALOG  ]  &&  /usr/bin/dos2unix  -k  $ALOG
/bin/mail  -s  "$bt"  "$sjr"  <$ALOG

[root@server  ~]#  chmod  777  check_

hard.sh                                          //为脚本加权
[root@server  ~]#  ./check_hard.sh
```

bt，sjr　定义发件标题和收件人信息

```
/usr/bin/dos2unix  -k  $ALOG
```

使用dos2unix命令转换字符，避免收到邮件不显示正常邮件内容而出现ATT00001.bin的错误；