101个shell脚本

 ZeroOne01 关注

2017-11-30 22:11:55 79811人阅读 2人评论

> 本文用于记录学习和日常中使用过的shell脚本

【脚本1】打印形状

打印等腰三角形、直角三角形、倒直角三角形、菱形

```
#!/bin/bash
# 等腰三角形
read -p "Please input the length: " n
for i in `seq 1 $n`
do
   for ((j=$n;j>i;j--))
   do
      echo -n " "
   done

   for m in `seq 1 $i`
   do
     echo -n "* "
   done
   echo
done

# 倒直角三角形
read -p "Please input the length: " len
for i in `seq 1 $len`
do
    for j in `seq $i $len`
    do
       echo -n "* "
    done
    echo
done

# 直角三角形
read -p "Please input the length: " len
for i in `seq 1 $len`
do
    for((j=1;j<=$i;j++))
    do
       echo -n "* "
    done
    echo
done

# 菱形
read -p "Please input the length: " n

for i in `seq 1 $n`
do
   for ((j=$n;j>i;j--))
   do
      echo -n " "
   done
   for m in `seq 1 $i`
   do
     echo -n "* "
```

```
    done
    echo
done

for i in `seq 1 $n`
do
    for((j=1;j<=$i;j++))
    do
       echo -n " "
    done
    for((k=$i;k<=$len-1;k++))
    do
          echo -n "* "
    done
    echo
done
```

【脚本2】截取字符串

现有一个字符串如下：

> http://www.aaa.com/root/123.htm

请根据以下要求截取出字符串中的字符：

1.取出www.aaa.com/root/123.htm

2.取出123.htm

3.取出http://www.aaa.com/root

4.取出http:

5.取出http://

6.取出www.aaa.com/root/123.htm

7.取出123

8.取出123.htm

```
#!/bin/bash
var="http://www.aaa.com/root/123.htm"
#1.
echo $var |awk -F '//' '{print $2}'
#2.
echo $var |awk -F '/' '{print $5}'
#3.
echo $var |grep -o 'http.*root'
#4.
echo $var |awk -F '/' '{print $1}'
#5.
echo $var |grep -o 'http://'
#6.
echo $var |grep -o 'www.*htm'
#7.
echo $var |grep -o '123'
#8.
echo $var |grep -o '123.htm'
```

【脚本3】tomcat启动脚本

emm。。这个脚本是因为tomcat没有自带的能够给service开机启动的脚本，我就琢磨着自己写了一个简单的启动脚本，如下：

```
#!/bin/bash
# chkconfig:2345 64 36
# description: Tomcat start/stop/restart script.

### BEGIN INIT INFO
# Provides: tomcat
# Required-Start:
# Should-Start:
# Required-Stop:
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: start and stop Tomcat
# Description: Tomcat Service start&restart&stop script
### END INIT INFO

##Written by zero.##
JAVA_HOME=/usr/local/jdk1.8/
JAVA_BIN=/usr/local/jdk1.8/bin
JRE_HOME=/usr/local/jdk1.8/jre
PATH=$PATH:/usr/local/jdk1.8/bin:/usr/local/jdk1.8/jre/bin
CLASSPATH=/usr/local/jdk1.8/jre/lib:/usr/local/jdk1.8/lib:/usr/local/jdk1.8/jre/lib/charsets.jar
TOMCAT_BIN=/usr/local/tomcat/bin
RETVAL=0
prog="Tomcat"

start()
{
    echo  "Starting $prog......"
    /bin/bash  $TOMCAT_BIN/startup.sh
    RETVAL=$?
    return  $RETVAL
}
stop()
{
    echo  "Stopping $prog......"
    /bin/bash  $TOMCAT_BIN/shutdown.sh
    RETVAL=$?
    return  $RETVAL
}
restart(){
    echo  "Restarting $prog......"
    stop
    start
}

case  "$1"  in
    start)
         start
         ;;
    stop)
        stop
        ;;
    restart)
        restart
        ;;
    *)
        echo  $"Usage: $0 {start|stop|restart}"
        RETVAL=1
esac
exit  $RETVAL
```

【脚本4】自定义rm命令

linux系统的rm命令太危险，一不小心就会删除掉系统文件。 写一个shell脚本来替换系统的rm命令，要求当删除一个文件或者目录时，都要做一个备份，然后再删除。下面分两种情况，做练习：

1. 简单的实现：

假设有一个大的分区/data/，每次删除文件或者目录之前，都要先在/data/下面创建一个隐藏目录，以日期/时间命名，比如/data/.201703271012/，然后把所有删除的文件同步到该目录下面，可以使用rsync -R 把文件路径一同同步，示例：

```
#!/bin/bash
fileName=$1
now=`date +%Y%m%d%H%M`
read -p "Are you sure delete the file or directory $1? yes|no: " input

if [ $input == "yes" ] || [ $input == "y" ]
then
    mkdir /data/.$now
    rsync -aR $1/ /data/.$now/$1/
    /bin/rm -rf $1
elif [ $input == "no" ] || [ $input == "n" ]
then
    exit 0
else
    echo "Only input yes or no"
    exit
fi
```

2.复杂的实现：

不知道哪个分区有剩余空间，在删除之前先计算要删除的文件或者目录大小，然后对比系统的磁盘空间，如果够则按照上面的规则创建隐藏目录，并备份，如果没有足够空间，要提醒用户没有足够的空间备份并提示是否放弃备份，如果用户输入yes，则直接删除文件或者目录，如果输入no，则提示未删除，然后退出脚本，示例：

```
#!/bin/bash
fileName=$1
now=`date +%Y%m%d%H%M`
f_size=`du -sk $1 |awk '{print $1}'`
disk_size=`LANG=en; df -k |grep -vi filesystem |awk '{print $4}' |sort -n |tail -n1`
big_filesystem=`LANG=en; df -k |grep -vi filesystem |sort -n -k4 |tail -n1 |awk '{print $NF}'`

if [ $f_size -lt $disk_size ]
then
   read -p "Are you sure delete the file or directory: $1 ? yes|no: " input
   if [ $input == "yes" ] || [ $input == "y" ]
   then
      mkdir -p $big_filesystem/.$now && rsync -aR $1 $big_filesystem/.$now/ && /bin/rm -rf $1
   elif [ $input == "no" ] || [ $input == "n" ]
   then
      exit 0
   else
      echo "Only input 'yes' or 'no'."
   fi
else
   echo "The disk size is not enough to backup the file: $1."
   read -p "Do you want to delete "$1"? yes|no: " input
   if [ $input == "yes" ] || [ $input == "y" ]
   then
       echo "It will delete "$1" after 5 seconds whitout backup."
       for i in `seq 1 5`; do echo -ne "."; sleep 1; done
     echo
       /bin/rm -rf $1
   elif [ $input == "no" ] || [ $input == "n" ]
   then
       echo "It will not delete $1."
       exit 0
   else
       echo "Only input 'yes' or 'no'."
   fi
fi
```

【脚本5】数字求和

编写shell脚本，要求输入一个数字，然后计算出从1到输入数字的和，要求，如果输入的数字小于1，则重新输入，直到输入正确的数字为止，示例：

```
#!/bin/bash
while :
do
    read -p "Please enter a positive integer: " n
    if [ $n -lt 1 ]
    then
        echo "It can't be less than 1"
    else
        break
    fi
done

num=1
for i in `seq 2 $n`
do
    num=$[$num+$i]
done

echo $num
```

【脚本6】拷贝目录

编写shell脚本，把/root/目录下的所有目录（只需要一级）拷贝到/tmp/目录下：

```
#!/bin/bash
cd /root/
list=(`ls`)

for i in ${list[@]}
do
    if [ -d $i ]
    then
        cp -r $i /tmp/
    fi
done
```

【脚本7】批量建立用户

编写shell脚本，批量建立用户user_00, user_01, ... user_100并且所有用户同属于users组：

```
#!/bin/bash
group=`cat /etc/group |grep -o users`
if [ $group == "users" ]
then
     for i in `seq 0 100`
     do
       if [ $i -lt 10 ]
       then
           useradd -g users user_0$i
       else
           useradd -g users user_$i
       fi
     done
else
     echo "users group not found!"
     exit 1
fi
```

删除以上脚本批量添加的用户：

```
#!/bin/bash
for i in `seq 0 100`
do
   if [ $i -lt 10 ]
   then
       userdel -r user_0$i
   else
       userdel -r user_$i
   fi
done
```

【脚本8】每日生成一个文件

要求：请按照这样的日期格式（xxxx-xx-xx）每日生成一个文件，例如今天生成的文件为）2017-07-05.log， 并且把磁盘的使用情况写到到这个文件中，（不用考虑cron，仅仅写脚本即可）

```
#!/bin/bash
fileName=`date +%F`
c=`df -h`
echo "$c" > /root/$fileName.log
```

【脚本9】统计ip

有一个日志文件，日志片段：如下：

> 112.111.12.248 – [25/Sep/2013:16:08:31 +0800]formula-x.haotui.com “/seccode.php?update=0.5593110133088248″ 200″http://formula-x.haotui.com/registerbbs.php” “Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1;)”
> 61.147.76.51 – [25/Sep/2013:16:08:31 +0800]xyzdiy.5d6d.com “/attachment.php?aid=4554&k=9ce51e2c376bc861603c7689d97c04a1&t=1334564048&fid=9&sid=zgohwYoLZq2qPW233ZIRsJiUeu22XqE8f49jY9mouRSoE71″ 301″http://xyzdiy.×××thread-1435-1-23.html” “Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727)”

要求： 统计出每个IP的访问量有多少？

> awk '{print $1}' 1.log |sort -n |uniq -c |sort -n

解释：sort -n会按照数值而不是ASCII码来排序awk截取出来的IP。然后uniq命令用于报告或忽略文件中的重复行，加上-c选项后会在每列旁边显示该行重复出现的次数，在这一步就完成了统计。不过最后还得再让sort -n排序一下uniq -c统计出来的结果。

【脚本10】统计内存使用

写一个脚本计算一下linux系统所有进程占用内存大小的和。

实现代码：

```
#!/bin/bash
count=0
# 这个循环会遍历出每个进程占用的内存大小
for i in `ps aux |awk '{print $6}' |grep -v 'RSS'`
do
   # 将遍历出来的数字进行累加
   count=$[$count+$i]
done
```

```
# 就得到所有进程占用内存大小的和了
echo "$count/kb"
```

也可以使用awk 一条命令计算：

> ps aux |grep -v 'RSS TTY' |awk '{sum=sum+$6};END{print sum}'

解释：grep -v是忽略 'RSS TTY' 所存在的那一行，后面的awk声明了一个变量sum，sum将前面命令列出来的数字进行累加，END之后就将累加后的sum打印出来，就得到所有进程占用内存大小的和了。

【脚本11】简单的监控脚本

设计一个简单的脚本，监控远程的一台机器(假设ip为123.23.11.21)的存活状态，当发现宕机时发一封邮件给你自己。

```
#!/bin/bash
ip="123.23.11.21"
email="user@example"

while 1
do
  ping -c10 $ip > /dev/null 2>/dev/null
  if [ $? != "0" ]
  then
       # 调用一个用于发邮件的脚本
     python /usr/local/sbin/mail.py $email "$ip down" "$ip is down"
  fi
  sleep 30
done
```

mail.py 脚本代码：

```
#!/usr/bin/env python
#-*- coding: UTF-8 -*-
import os,sys
reload(sys)
sys.setdefaultencoding('utf8')
import getopt
import smtplib
from email.MIMEText import MIMEText
from email.MIMEMultipart import MIMEMultipart
from  subprocess import *
def sendqqmail(username,password,mailfrom,mailto,subject,content):
    # 邮箱的服务地址
    gserver = 'smtp.qq.com'
    gport = 25
    try:
        msg = MIMEText(unicode(content).encode('utf-8'))
        msg['from'] = mailfrom
        msg['to'] = mailto
        msg['Reply-To'] = mailfrom
        msg['Subject'] = subject
        smtp = smtplib.SMTP(gserver, gport)
        smtp.set_debuglevel(0)
        smtp.ehlo()
        smtp.login(username,password)
        smtp.sendmail(mailfrom, mailto, msg.as_string())
        smtp.close()
    except Exception,err:
        print "Send mail failed. Error: %s" % err
def main():
    to=sys.argv[1]
    subject=sys.argv[2]
    content=sys.argv[3]
    #定义QQ邮箱的账号和密码，你需要修改成你自己的账号和密码
```

```
    sendqqmail('1234567@qq.com','aaaaaaaaaa','1234567@qq.com',to,subject,content)
if __name__ == "__main__":
    main()

#####脚本使用说明######
#1. 首先定义好脚本中的邮箱账号和密码
#2. 脚本执行命令为：python mail.py 目标邮箱 "邮件主题" "邮件内容"
```

【脚本12】批量更改文件名

需求：

- 找到/123目录下所有后缀名为.txt的文件
- 批量修改.txt为.txt.bak
- 把所有.bak文件打包压缩为123.tar.gz
- 批量还原文件的名字，即把增加的.bak再删除

代码：

```
#!/bin/bash
now=`date +%F_%T`
mkdir /tmp/123_$now
for txt in `ls /123/*.txt`
do
  mv $txt $txt.bak
  for f in $txt
  do
    cp $txt.bak /tmp/123_$now
  done
done

cd /tmp/
tar czf 123.tar.gz 123_$now/

for txt in `ls /123/*.txt.bak`
do
  name=`echo $txt |awk -F '.' '{OFS="."} {print $1,$2}'`
  mv $txt $name
done
```

【脚本13】监控80端口

需求：
写一个脚本，判断本机的80端口（假如服务为httpd）是否开启着，如果开启着什么都不做，如果发现端口不存在，那么重启一下httpd服务，并发邮件通知你自己。脚本写好后，可以每一分钟执行一次，也可以写一个死循环的脚本，30s检测一次。
发邮件的脚本参考【脚本11】的示例代码。
代码：

```
#!/bin/bash
email="user@example.com"
if netstat -lntp |grep ':80' |grep 'httpd'
then
   echo "80 port no problem"
   exit
else
   /usr/local/apache2.4/bin/apachectl restart
   python mail.py $email "check_80port" "The 80 port is down."
   n=`ps aux |grep httpd|grep -cv grep`

   if [ $n -eq 0 ]
   then
      /usr/local/apache2/bin/apachectl start 2>/tmp/apache_start.err
```

```
    fi

    if [ -s /tmp/apache_start.err ]
      then
       python mail.py  $mail 'apache_start_error' `cat /tmp/apache_start.err`
    fi
fi
```

【脚本14】备份数据库

需求：

设计一个shell脚本来备份数据库，首先在本地服务器上保存一份数据，然后再远程拷贝一份，本地保存一周的数据，远程保存一个月。

假定，我们知道mysql root账号的密码，要备份的库为discuz，本地备份目录为/bak/mysql, 远程服务器ip为192.168.123.30，远程提供了一个rsync服务，备份的地址是 192.168.123.30::backup . 写完脚本后，需要加入到cron中，每天凌晨3点执行。
脚本代码：

```
#!/bin/bash
PATH=$PATHi:/usr/local/mysql/bin
week=`date +%w`
today=`date +d`
passwd="123456"
backdir="/data/mysql"
r_backupIP="192.168.123.30::backup"

exec 1>/var/log/mysqlbak.log 2>/var/log/mysqlbak.log
echo "mysql backup begin at `date +%F %T`."

# 本地备份
mysqldump -uroot -p$passwd --default-character-set=utf8 discuz >$backdir/$week.sql
# 同步备份到远程机器
rsync -az $backdir/$week.sql $r_backupIP/$today.sql

echo "mysql backup end at `date +%F %T`."
```

然后加入cron

```
0 3 * * * /bin/bash /usr/local/sbin/mysqlbak.sh
```

【脚本15】自动重启php-fpm服务

服务器上跑的是LNMP环境，近期总是有502现象。502为网站访问的状态码，200正常，502错误是nginx最为普通的错误状态码。由于502只是暂时的，并且只要一重启php-fpm服务则502消失，但不重启的话，则会一直持续很长时间。所以有必要写一个监控脚本，监控访问日志的状态码，一旦发生502，则自动重启一下php-fpm。

我们设定：

1. access_log /data/log/access.log
2. 脚本死循环，每10s检测一次（假设每10s钟的日志条数为300左右）
3. 重启php-fpm的方法是 /etc/init.d/php-fpm restart

脚本代码：

```
#!/bin/bash
access_log="/data/log/access.log"
N=10
while :
do
    # 因为10秒大概产生300条日志记录
    tail -n300 $access_log > /tmp/log
    # 拿出log中包含502的日志行数
    n_502=`grep -c "502" /tmp/log`
```

```
    # 如果行数大于10
    if [ $n_502 -ge $N ]
    then
        # 就记录一下系统状态
        top -bn1 > /tmp/`date +%H%M%S`-top.log
        vmstat 1 5 > /tmp/`date +%H%M%S`-vm.log
        # 然后才重启服务，并把错误信息重定向
        /etc/init.d/php-fpm restart 2> /dev/null
        # 重启php-fpm服务后，应先暂缓1分钟，而后继续每隔10s检测一次
        sleep(60)
    fi
    sleep(10)
done
```

【脚本16】文本替换

将memcached里的数据导出到文件中，然后再导入memcached里却发现数据过期了，这是因为导出的数据是带有一个时间戳的，这个时间戳就是该条数据过期的时间点，如果当前时间已经超过该时间戳，那么是导入不进去的。不过可以修改文件中的时间戳来保证导入时数据的有效期。可以写一个简单的脚本批量替换这些文件中的时间戳：

```
#!/bin/bash
hour=`date -d "+1 hour" +%s`   # 一个小时之后的时间戳
data_time=`cat data.txt |grep add |awk '{print $4}' |sort -n |uniq`
for i in $data_time
do
   sed -i "s/$i/$hour/g" `grep $i -rl /root/data.txt`
done
```

【脚本17】启动容器

docker每次关闭都会连带着将运行中的容器关闭，所以每次启动docker后都需要逐个去启动容器，很麻烦，由于是实验用的虚拟机不是线上的机器，所以就直接写了一个很简单的循环来启动容器：

```
#!/bin/bash
/usr/bin/systemctl start docker

for i in `docker ps -a |grep 'Exited' |awk '{print $1}'`
do
  /usr/bin/docker start $i
done
```

【脚本18】删除文本中的字母

要求：把一个文本文档的前5行中包含字母的行删除掉，同时把6到10行中的全部字母删除掉。

参考答案：假设文本名字叫做1.txt，并且文本行数大于10，脚本如下：

```
#!/bin/bash

## 先获取该文本的行数
rows=`wc -l 1.txt |awk '{print $1}'`

## 对前5行进行处理
for i in `seq 1 5`
do
  ## 使用sed把每一行的内容赋值给变量
  row=`sed -n "$i"p 1.txt`
  ## 用grep 判定是否匹配字母，-v取反，-q不输出内容
  if echo $row |grep -vq '[a-zA-Z]'
  then
```

```
        echo $row
    fi
done

## 对6-10行做删除字母处理
for i in `seq 6 10`
do
    row=`sed -n "$i"p 1.txt`
    echo $row |sed 's/[a-zA-Z]//g'
done

## 剩余的直接输出
for i in `seq 11 $rows`
do
    sed -n "$i"p 1.txt
done

##若想把更改内容写入到1.txt，还需要把以上内容重定向到一个文本中，然后删除1.txt，再把刚刚重定向的文件更名为1.txt
```

【脚本19】查找字母数小于6的单词

用shell打印下面这句话中字母数小于6的单词。

Bash also interprets a number of multi-character options.

脚本如下：

```
#!/bin/bash
for s in Bash also interprets a number of multi-character options
do
    n=`echo $s |wc -c`
    if [ $n -lt 6 ]
    then
        echo $s
    fi
done
```

【脚本20】输入数字执行对应命令

写一个脚本实现如下功能： 输入一个数字，然后运行对应的一个命令。显示命令如下：

> *cmd meau** 1—date 2–ls 3–who 4–pwd

当输入1时，会运行date, 输入2时运行ls, 依此类推。

实现脚本如下：

```
#!/bin/bash
echo "*cmd meau**  1—date 2–ls 3–who 4–pwd"

read -p "please input a number 1-4: " n
case $n in
    1)
        date
    ;;
    2)
        ls
    ;;
    3)
        who
    ;;
    4)
        pwd
```

```
    ;;
    *)
        echo  "Please input a number: 1-4"
    ;;
esac
```

【脚本21】批量创建用户并设置密码

用shell脚本实现如下需求：
添加user_00 – user_09 10个用户，并且给他们设置一个随机密码，密码要求10位包含大小写字母以及数字，注意需要把每个用户的密码记录到一个日志文件里。

提示：

1. 随机密码使用命令 mkpasswd
2. 在脚本中给用户设置密码，可以使用echo 然后管道passwd命令

实现脚本如下：

```
#!/bin/bash
for i in `seq 00 09`
do
   useradd user_$i
   p=`mkpasswd -s 0 -l 10`
   echo "user_$i $p" >> /tmp/user0_9.pw
   echo $p |passwd -stdin user_$i
done
```

【脚本22】监控httpd进程

在服务器上，写一个监控脚本。

1. 每隔10s去检测一次服务器上的httpd进程数，如果大于等于500的时候，就需要自动重启一下apache服务，并检测启动是否成功？
2. 若没有正常启动还需再一次启动，最大不成功数超过5次则需要理解发邮件通知管理员，并且以后不需要再检测！
3. 如果启动成功后，1分钟后再次检测httpd进程数，若正常则重复之前操作（每隔10s检测一次），若还是大于等于500，那放弃重启并需要发邮件给管理员，然后自动退出该脚本。假设其中发邮件脚本为mail.py

实现脚本如下：

```
#!/bin/bash

check_service(){
  n=0
  for i in `seq 1 5`
  do
    # apachectl命令所在路径
    /usr/local/apache2/bin/apachectl restart 2> /tmp/apache.err
    if [$? -ne 0 ]
    then
        n=$[$n-1]
    else
        break
    fi
  done

  if [ $n -eq 5 ]
  then
    ## mail.py的内容参考https://coding.net/u/aminglinux/p/aminglinux-book/git/blob/master/D22Z/mail.py
    python mail.py "123@qq.com" "httpd service down" `cat /tmp/apache.err`
    exit
  fi
}
```

```
while :
do
  t_n=`ps -C httpd --no-heading |wc -l`
  if [ $t_n -ge 500 ]
  then
     /usr/local/apache2/bin/apachectl restart
     if [ $? -ne 0 ]
     then
        check_service
     fi
     sleep 60
     t_n=`ps -C httpd --no-heading |wc -l`
     if [ $t_n -ge 500]
     then
        python mail.py "123@qq.com" "httpd service somth wrong" "the httpd process is budy."
        exit
     fi
  fi
  sleep 10
done
```

【脚本23】封ip

需求： 根据web服务器上的访问日志，把一些请求量非常高的ip给拒绝掉！

分析： 我们要做的，不仅是要找到哪些ip请求量不合法，并且还要每隔一段时间把之前封掉的ip（若不再继续请求了）给解封。 所以该脚本的关键点在于定一个合适的时间段和阈值。

比如， 我们可以每一分钟去查看一下日志，把上一分钟的日志给过滤出来分析，并且只要请求的ip数量超过100次那么就直接封掉。 而解封的时间又规定为每半小时分析一次，把几乎没有请求量的ip给解封！

参考日志文件片段：

```
157.55.39.107 [20/Mar/2015:00:01:24 +0800] www.aminglinux.com “/bbs/thread-5622-3-1.html” 200 “-” “Mozilla/5.0 (compatible; bir
61.240.150.37 [20/Mar/2015:00:01:34 +0800] www.aminglinux.com “/bbs/search.php?mod=forum&srchtxt=LNMP&formhash=8f0c7da9&searc
```

脚本实现如下：

```
#!/bin/bash
## 日志文件路径
log_file="/home/logs/client/access.log"
## 当前时间减一分钟的时间
d1=`date -d "-1 minute" +%H:%M`
## 当前时间的分钟段
d2=`date +%M`
## iptables命令所在的路径
ipt="/sbin/iptables"
## 用于存储访问日志里的ip
ips="/tmp/ips.txt"

## 封ip
block(){
   ## 把日志文件中的ip过滤出来，去掉重复的ip，并统计ip的重复次数以及对ip进行排序，最后将结果写到一个文件中
   grep "$d1:" $log_file |awk '{print $1}' |sort -n |uniq -c |sort -n > $ips
   ## 将文件里重复次数大于100的ip迭代出来
   for ip in `awk '$1 > 100 {print $2}' $ips`
   do
      ## 通过防火墙规则对这些ip进行封禁
      $ipt -I INPUT -p -tcp --dport 80 -s $ip -j REJECT
      ## 将已经封禁的ip输出到一个文件里存储
      echo "`date +%F-%T` $ip" >> /tmp/badip.txt
   done
}

## 解封ip
```

```
unblock(){
    ## 将流量小于15的规则索引过滤出来
    for i in `$ipt -nvL --line-number |grep '0.0.0.0/0' |awk '$2 < 15 {print $1}' |sort -nr`
    do
        ## 通过索引来删除规则
        $ipt -D INPUT $i
    done
    ## 清空规则中的数据包计算器和字节计数器
    $ipt -Z
}

## 为整点或30分钟就是过了半个小时，就需要再进行分析
if [ $d2 == "00" ] || [ $d2 == "30" ]
then
    unblock
    block
else
    block
fi
```

【脚本24】部署前端项目

最近做了一个web前端的项目，需要编写一个脚本完成项目的上线。

脚本实现如下：

```
#!/bin/bash

#
# 使用方法：
# mmall：front_deploy.sh mmall-fe
# admin：front_deploy.sh admin-fe
#

GIT_HOME=/developer/git-repository/    # 从git仓库拉取下来的源码的存放路径
DEST_PATH=/product/frontend/     # 项目打包后的发布路径

# cd dir
if [ ! -n "$1" ]
then
     echo -e "请输入要发布的项目！"
     exit
fi

if [ $1 = "mmall-fe" ]
then
     echo -e "===========Enter mall-fe============="
     cd $GIT_HOME$1
elif [ $1 = "admin-fe" ]
then
     echo -e "===========Enter mall-fe============="
     cd $GIT_HOME$1
else
     echo -e "输入的项目名没有找到！"
     exit
fi

# clear git dist
echo -e "===========Clear Git Dist============="
rm -rf ./dist

# git操作
echo -e "===========git checkout master============="
git checkout master

echo -e "===========git pull============="
git pull
```

```
# npm install
echo -e "===========npm install============="
npm install --registry=https://registry.npm.taobao.org

# npm run dist
echo -e "===========npm run dist============="
npm run dist

if [ -d "./dist" ]
then
    # backup dest
    echo -e "===========dest backup============="
    mv $DEST_PATH$1/dist $DEST_PATH$1/dist.bak

    # copy
    echo -e "===========copy============="
    cp -R ./dist $DEST_PATH$1

    # echo result
    echo -e "===========Deploy Success============="
else
    echo -e "===========Deploy Error============="
fi
```

【脚本25】找规律打印数字

请详细查看如下几个数字的规律，并使用shell脚本输出后面的十个数字。

> 10 31 53 77 105 141 .....

试题解析：
我想大多数人都会去比较这些数字的差值：

```
10  31  53  77  105  141
21   22   24   28   36
```

但是这个差值看，并没有什么规律，而我们再仔细看的时候，发现这个差值的差值是有规律的：

```
10  31  53  77  105  141
21   22   24   28   36
1      2     4     8
```

脚本实现：

```
#! /bin/bash
x=21
m=10
echo $m
for i in `seq 0 14`; do
 j=$[2**$i]
 m=$[$m+$x]
 echo $m
 x=$[$x+$j]
done
```

【脚本26】统计普通用户

写个shell，看看你的Linux系统中是否有自定义用户（普通用户），若是有，一共有几个？

假设所有普通用户都是uid大于1000的

脚本实现：

```
#!/bin/bash
 n=`awk -F ':' '$3>1000' /etc/passwd|wc -l`
 if [ $n -gt 0 ]
 then
      echo "There are $n common users."
 else
      echo "No common users."
 fi
```

【脚本27】监控磁盘使用率

写一个shell脚本，检测所有磁盘分区使用率和inode使用率并记录到以当天日期为命名的日志文件里，当发现某个分区容量或者inode使用量大于85%时，发邮件通知你自己。

思路：就是先df -h 然后过滤出已使用的那一列，然后再想办法过滤出百分比的整数部分，然后和85去比较，同理，inode也是一样的思路。

实现代码：

```
#!/bin/bash
## This script is for record Filesystem Use%,IUse% everyday and send alert mail when % is more than 85%.

log=/var/log/disk/`date +%F`.log
date +'%F %T' > $log
df -h >> $log
echo >> $log
df -i >> $log

for i in `df -h|grep -v 'Use%'|sed 's/%//'|awk '{print $5}'`; do
    if [ $i -gt 85 ]; then
        use=`df -h|grep -v 'Use%'|sed 's/%//'|awk '$5=='$i' {print $1,$5}'`
        echo "$use" >> use
    fi
done
if [ -e use ]; then

   ##这里可以使用咱们之前介绍的mail.py发邮件
    mail -s "Filesystem Use% check" root@localhost < use
    rm -rf use
fi

for j in `df -i|grep -v 'IUse%'|sed 's/%//'|awk '{print $5}'`; do
    if [ $j -gt 85 ]; then
        iuse=`df -i|grep -v 'IUse%'|sed 's/%//'|awk '$5=='$j' {print $1,$5}'`
        echo "$iuse" >> iuse
    fi
done
if [ -e iuse ]; then
    mail -s "Filesystem IUse% check" root@localhost < iuse
    rm -rf iuse
fi
```

思路：

1. df -h、df -i 记录磁盘分区使用率和inode使用率，date +%F 日志名格式
2. 取出使用率(第5列)百分比序列，for循环逐一与85比较，大于85则记录到新文件里，当for循环结束后，汇总超过85的一并发送邮件(邮箱服务因未搭建，发送本地root账户)。

此脚本正确运行前提：

- 该系统没有逻辑卷的情况下使用，因为逻辑卷df -h、df -i 时，使用率百分比是在第4列，而不是第5列。如有逻辑卷，则会漏统计逻辑卷使用情况。

【脚本28】获取文件列表

有一台服务器作为web应用，有一个目录（/data/web/attachment）不定时地会被用户上传新的文件，但是不知道什么时候会上传。所以，需要我们每5分钟做一次检测是否有新文件生成。

请写一个shell脚本去完成检测。检测完成后若是有新文件，还需要将新文件的列表输出到一个按年、月、日、时、分为名字的日志里。请不要想的太复杂，核心命令只有一个 find /data/web/attachment -mmin -5

思路： 每5分钟检测一次，那肯定需要有一个计划任务，每5分钟去执行一次。脚本检测的时候，就是使用find命令查找5分钟内有过更新的文件，若是有更新，那这个命令会输出东西，否则是没有输出的。固，我们可以把输出结果的行数作为比较对象，看看它是否大于0。

实现代码：

```
#!/bin/bash
d=`date -d "-5 min" +%Y%m%d%H%M`
basedir=/data/web/attachment
find $basedir/ -type f  -mmin -5 > /tmp/newf.txt
n=`wc -l /tmp/newf.txt`
if [ $n -gt 0 ]; then
    /bin/mv /tmp/newf.txt /tmp/$d

fi
```

【脚本29】统计常用命令

写一个shell脚本来看看你使用最多的命令是哪些，列出你最常用的命令top10。

思路：我们要用到一个文件就是.bash_history，然后再去sort、uniq，剩下的就不用我多说了吧。很简单一个shell。

一条命令即可：

```
sort /root/.bash_history |uniq -c |sort -nr |head
```

【脚本30】统计日志大小

假如我们需要每小时都去执行你写的脚本。在脚本中实现这样的功能，当时间是0点和12点时，需要将目录/data/log/下的文件全部清空，注意只能清空文件内容而不能删除文件。而其他时间只需要统计一下每个文件的大小，一个文件一行，输出到一个按日期和时间为名字的日志里。需要考虑/data/log/目录下的二级、三级、… 等子目录里面的文件。

实现代码：

```
#!/bin/bash

logdir="/data/log"
t=`date +%H`
d=`date +%F-%H`
[ -d /tmp/log_size ] || mkdir /tmp/log_size
for log in `find $logdir -type f`
do
    if [ $t == "0" ] || [ $t == "12" ]
    then
    true > $log
    else
    du -sh $log >>/tmp/log_size/$d
    fi
done
```

【脚本31】统计数字并求和

计算文档a.txt中每一行中出现的数字个数并且要计算一下整个文档中一共出现了几个数字。例如a.txt内容如下：

```
12aa*lkjskdj
alskdflkskdjflkjj
```

我们脚本名字为 ncount.sh, 运行它时：

```
bash ncount.sh a.txt
```

输入结果应该为：

```
2
0
sum:2
```

实现代码：

```
#!/bin/bash

n=`wc -l a.txt|awk '{print $1}'`
sum=0
for i in `seq 1 $n`
do
    line=`sed -n "$i"p a.txt`
    n_n=`echo -n $line|sed 's/[^0-9]//g'|wc -c`
    echo line $i number: $n_n
    sum=$[$sum+$n_n]
done

echo sum is $sum
```

【脚本32】检测文件改动

有两台Linux服务器A和B，假如A可以直接ssh到B，不用输入密码。A和B都有一个目录叫做/data/web/ 这下面有很多文件，当然我们不知道具体有几层子目录，假若之前A和B上该目录下的文件都是一模一样的。但现在不确定是否一致了。固需要我们写一个脚本实现这样的功能，检测A机器和B机器/data/web/目录下文件的异同，我们以A机器上的文件作为标准。比如，假若B机器少了一个a.txt文件，那我们应该能够检测出来，或者B机器上的b.txt文件有过改动，我们也应该能够检测出来（B机器上多了文件我们不用考虑）。

提示： 使用核心命令 md5sum a.txt 算出md5值，去和B机器上的比较。

实现代码：

```
#!/bin/bash
#假设A机器到B机器已经做了无密码登录设置
dir=/data/web
##假设B机器的IP为192.168.0.100
B_ip=192.168.0.100
find $dir -type f |xargs md5sum >/tmp/md5.txt
ssh $B_ip "find $dir -type f |xargs md5sum >/tmp/md5_b.txt"
scp $B_ip:/tmp/md5_b.txt /tmp
for f in `awk '{print $2}' /tmp/md5.txt`
do
    if grep -q "$f" /tmp/md5_b.txt
    then
        md5_a=`grep $f /tmp/md5.txt|awk '{print $1}'`
        md5_b=`grep $f /tmp/md5_b.txt|awk '{print $1}'`
        if [ $md5_a != $md5_b ]
        then
            echo "$f changed."
        fi
    else
        echo "$f deleted. "
    fi
done
```

【脚本33】统计网卡流量

写一个脚本,检测你的网络流量，并记录到一个日志里。需要按照如下格式，并且一分钟统计一次（只需要统计外网网卡，假设网卡名字为eth0）：

```
2017-08-04 01:11
eth0 input: 1000bps
eth0 output : 200000bps
################
2017-08-04 01:12
eth0 input: 1000bps
eth0 output : 200000bps
```

提示：使用sar -n DEV 1 59 这样可以统计一分钟的平均网卡流量，只需要最后面的平均值。另外，注意换算一下，1byt=8bit

实现代码：

```
#!/bin/bash

while :
do
    LANG=en
    DATE=`date +"%Y-%m-%d %H:%M"`
    LOG_PATH=/tmp/traffic_check/`date +%Y%m`
    LOG_FILE=$LOG_PATH/traffic_check_`date +%d`.log
    [ -d $LOG_PATH ] || mkdir -p $LOG_PATH
    echo " $DATE" >> $LOG_FILE
    sar -n DEV 1 59|grep Average|grep eth0 \
    |awk '{print "\n",$2,"\t","input:",$5*1000*8,"bps", \
  "\t","\n",$2,"\t","output:",$6*1000*8,"bps" }' \
    >> $LOG_FILE
    echo "####################" >> $LOG_FILE
done
```

【脚本34】系统-批量杀进程

今天发现网站访问超级慢，top看如下：

```
[root@j[redacted]]# top

top - 10:34:23 up 1266 days, 22:41, 10 users,  load average: 205.97, 206.37, 207.43
Tasks: 855 total, 207 running, 642 sleeping,   4 stopped,   2 zombie
Cpu(s):  4.0%us, 95.7%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.2%si,  0.0%st
Mem:  16410752k total,  4161544k used, 12249208k free,   107888k buffers
Swap:  4096564k total,   169880k used,  3926684k free,   206804k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
19829 root      20   0 10.1g 1.8g  15m S 50.5 11.5  16:04.85 java
  715 root      25   0  8704  704  552 R  6.5  0.0 927:36.25 sh
  835 root      25   0  8704  700  552 R  6.5  0.0 366:27.08 sh
  843 root      25   0  8704  700  552 R  6.5  0.0 196:43.25 sh
  891 root      25   0  8704  700  552 R  6.5  0.0 695:46.66 sh
 1216 root      25   0  8704  696  552 R  6.5  0.0   1944:43 sh
 1894 root      25   0  8704  700  552 R  6.5  0.0 509:39.17 sh
 1928 root      25   0  8704  696  552 R  6.5  0.0   1302:37 sh
 2061 root      25   0  8704  704  552 R  6.5  0.0 359:34.72 sh
 2110 root      25   0  8704  696  552 R  6.5  0.0 686:45.64 sh
 2220 root      25   0  8704  700  552 R  6.5  0.0 912:37.27 sh
 2290 root      25   0  8704  700  552 R  6.5  0.0 297:13.79 sh
 2418 root      25   0  8704  704  552 R  6.5  0.0 191:05.08 sh
 2741 root      25   0  8704  704  552 R  6.5  0.0   1903:05 sh
 3412 root      25   0  8704  700  552 R  6.5  0.0   1281:56 sh
 3635 root      25   0  8704  704  552 R  6.5  0.0 903:00.87 sh
 4072 root      25   0  8704  704  552 R  6.5  0.0 103:00.38 sh
 4352 root      25   0  8704  700  552 R  6.5  0.0   1850:52 sh
 4835 root      25   0  8704  700  552 R  6.5  0.0 667:08.53 sh
 5099 root      25   0  8704  704  552 R  6.5  0.0 891:31.97 sh
 5384 root      25   0  8704  704  552 R  6.5  0.0 182:20.23 sh
 5516 root      25   0  8704  704  552 R  6.5  0.0 100:22.39 sh
 5619 root      25   0  8704  700  552 R  6.5  0.0  30:33.37 sh
 5658 root      25   0  8704  704  552 R  6.5  0.0 488:30.54 sh
 5937 root      25   0  8704  696  552 R  6.5  0.0 355:39.16 sh
 6205 root      25   0  8704  700  552 R  6.5  0.0   1238:58 sh
 6300 root      25   0  8704  700  552 R  6.5  0.0 655:33.23 sh
 6570 root      25   0  8704  696  552 R  6.5  0.0 881:20.06 sh
 6856 root      25   0  8704  700  552 R  6.5  0.0 481:23.82 sh
 7044 root      25   0  8704  700  552 R  6.5  0.0  96:26.46 sh
 7522 root      25   0  8704  700  552 R  6.5  0.0   1221:03 sh
 7707 root      25   0  8704  704  552 R  6.5  0.0 648:30.19 sh
 7976 root      25   0  8704  704  552 R  6.5  0.0   1730:36 sh
 8071 root      25   0  8704  700  552 R  6.5  0.0 865:25.53 sh
 8189 root      25   0  8704  700  552 R  6.5  0.0 474:55.37 sh
 8514 root      25   0  8704  700  552 R  6.5  0.0 280:53.66 sh
```


有很多sh进程，再ps查看：

```
root      7040  0.0  0.0  8700   948 ?        Ss   Oct07   0:00 /bin/sh -c /bin/clearmen.sh 1>>/bin/clearmen.log 2>>/bin/clear
men.log
root      7044  6.3  0.0  8704   700 ?        R    Oct07  96:46 /bin/sh -c /bin/clearmen.sh 1>>/bin/clearmen.log 2>>/bin/clear
men.log
root      7248  0.0  0.0  8700   952 ?        Ss   Oct05   0:00 /bin/sh -c /bin/clearmen.sh 1>>/bin/clearmen.log 2>>/bin/clear
men.log
root      7249  7.2  0.0  8704   704 ?        R    Oct05 288:15 /bin/sh -c /bin/clearmen.sh 1>>/bin/clearmen.log 2>>/bin/clear
men.log
root      7521  0.0  0.0  8700   948 ?        Ss   Oct01   0:00 /bin/sh -c /bin/clearmen.sh 1>>/bin/clearmen.log 2>>/bin/clear
men.log
root      7522 12.4  0.0  8704   700 ?        R    Oct01 1221:22 /bin/sh -c /bin/clearmen.sh 1>>/bin/clearmen.log 2>>/bin/clea
rmen.log
root      7706  0.0  0.0  8700   952 ?        Ss   Oct03   0:00 /bin/sh -c /bin/clearmen.sh 1>>/bin/clearmen.log 2>>/bin/clear
men.log
root      7707  9.0  0.0  8704   704 ?        R    Oct03 648:49 /bin/sh -c /bin/clearmen.sh 1>>/bin/clearmen.log 2>>/bin/clear
men.log
root      7975  0.0  0.0  8700   952 ?        Ss   Sep30   0:00 /bin/sh -c /bin/clearmen.sh 1>>/bin/clearmen.log 2>>/bin/clear
men.log
root      7976 15.9  0.0  8704   704 ?        R    Sep30 1730:56 /bin/sh -c /bin/clearmen.sh 1>>/bin/clearmen.log 2>>/bin/clea
rmen.log
root      8070  0.0  0.0  8700   948 ?        Ss   Oct02   0:00 /bin/sh -c /bin/clearmen.sh 1>>/bin/clearmen.log 2>>/bin/clear
men.log
root      8071 10.2  0.0  8704   700 ?        R    Oct02 865:45 /bin/sh -c /bin/clearmen.sh 1>>/bin/clearmen.log 2>>/bin/clear
men.log
root      8188  0.0  0.0  8700   948 ?        Ss   Oct04   0:00 /bin/sh -c /bin/clearmen.sh 1>>/bin/clearmen.log 2>>/bin/clear
men.log
root      8189  8.1  0.0  8704   700 ?        R    Oct04 475:15 /bin/sh -c /bin/clearmen.sh 1>>/bin/clearmen.log 2>>/bin/clear
men.log
root      8409  0.0  0.0  8700   948 ?        Ss   Oct06   0:00 /bin/sh -c /bin/clearmen.sh 1>>/bin/clearmen.log 2>>/bin/clear
men.log
root      8410  6.7  0.0  8704   700 ?        R    Oct06 175:02 /bin/sh -c /bin/clearmen.sh 1>>/bin/clearmen.log 2>>/bin/clear
men.log
root      8513  0.0  0.0  8700   948 ?        Ss   Oct05   0:00 /bin/sh -c /bin/clearmen.sh 1>>/bin/clearmen.log 2>>/bin/clear
men.log
root      8514  7.1  0.0  8704   700 ?        R    Oct05 281:13 /bin/sh -c /bin/clearmen.sh 1>>/bin/clearmen.log 2>>/bin/clear
men.log
root      8641  0.0  0.0  8700   952 ?        Ss   Oct07   0:00 /bin/sh -c /bin/clearmen.sh 1>>/bin/clearmen.log 2>>/bin/clear
men.log
root      8643  6.3  0.0  8704   704 ?        R    Oct07  93:06 /bin/sh -c /bin/clearmen.sh 1>>/bin/clearmen.log 2>>/bin/clear
men.log
root      8770  0.0  0.0  8700   944 ?        Ss   Oct01   0:00 /bin/sh -c /bin/clearmen.sh 1>>/bin/clearmen.log 2>>/bin/clear
men.log
root      8771 12.3  0.0  8704   696 ?        R    Oct01 1201:02 /bin/sh -c /bin/clearmen.sh 1>>/bin/clearmen.log 2>>/bin/clea
rmen.log
root      8960  0.0  0.0  8700   952 ?        Ss   Oct03   0:00 /bin/sh -c /bin/clearmen.sh 1>>/bin/clearmen.log 2>>/bin/clear
men.log
root      8961  8.9  0.0  8704   704 ?        R    Oct03 638:29 /bin/sh -c /bin/clearmen.sh 1>>/bin/clearmen.log 2>>/bin/clear
men.log
root      9139  0.0  0.0  8700   944 ?        Ss   Sep30   0:00 /bin/sh -c /bin/clearmen.sh 1>>/bin/clearmen.log 2>>/bin/clear
men.log
root      9140 15.5  0.0  8704   696 ?        R    Sep30 1680:35 /bin/sh -c /bin/clearmen.sh 1>>/bin/clearmen.log 2>>/bin/clea
rmen.log
root      9416  0.0  0.0  8700   948 ?        Ss   Oct02   0:00 /bin/sh -c /bin/clearmen.sh 1>>/bin/clearmen.log 2>>/bin/clear
```


这个脚本，运行很慢，因为制定了cron，上一次还没有运行完，又有了新的运行任务。太多肯定会导致系统负载升高。当务之急就是先把这些在跑的给kill掉。那么我们可以使用一条命令，直接杀死所有的sh。

命令如下：

```
ps aux |grep clearmem.sh |grep -v grep|awk '{print $2}'|xargs kill
```

【脚本35】判断是否开启80端口

写一个脚本判断你的Linux服务器里是否开启web服务？（监听80端口）如果开启了，请判断出跑的是什么服务，是httpd呢还是nginx又或者是其他的什么？

实现代码：

```
#!/bin/bash
 port=`netstat -lnp | grep 80`
 if [ -z "port" ]; then
      echo "not start service.";
      exit;
 fi
 web_server=`echo $port | awk -F'/' '{print $2}'|awk -F : '{print $1}'`
case $web_server in
   httpd )
        echo "apache server."
   ;;
   nginx )
        echo "nginx server."
   ;;
   * )
        echo "other server."
   ;;
esac
```

【脚本36】监控mysql服务

假设，当前MySQL服务的root密码为123456，写脚本检测MySQL服务是否正常（比如，可以正常进入mysql执行show processlist），并检测一下当前的MySQL服务是主还是从，如果是从，请判断它的主从服务是否异常。如果是主，则不需要做什么。

实现代码：

```
#!/bin/bash
Mysql_c="mysql -uroot -p123456"
$Mysql_c -e "show processlist" >/tmp/mysql_pro.log 2>/tmp/mysql_log.err
n=`wc -l /tmp/mysql_log.err|awk '{print $1}'`

if [ $n -gt 0 ]
then
     echo "mysql service sth wrong."
else

     $Mysql_c -e "show slave status\G" >/tmp/mysql_s.log
     n1=`wc -l /tmp/mysql_s.log|awk '{print $1}'`

     if [ $n1 -gt 0 ]
     then
          y1=`grep 'Slave_IO_Running:' /tmp/mysql_s.log|awk -F : '{print $2}'|sed 's/ //g'`
          y2=`grep 'Slave_SQL_Running:' /tmp/mysql_s.log|awk -F : '{print $2}'|sed 's/ //g'`

          if [ $y1 == "Yes" ] && [ $y2 == "Yes" ]
          then
               echo "slave status good."
          else
               echo "slave down."
          fi
     fi
fi
```

【脚本37】带选项的用户脚本

要求如下：

1. 只支持三个选项 '–del' '–add' –help输入其他选项报错。
2. 使用'–add'需要验证用户名是否存在，存在则反馈存在。且不添加。 不存在则创建该用户，切>添加与该用户名相同的密码。并且反馈。
3. 使用'–del' 需要验证用户名是否存在，存在则删除用户及其家目录。不存在则反馈该用户不存>在。
4. –help 选项反馈出使用方法
5. 支持以，分隔 一次删除多个或者添加多个用户。
6. 能用echo $? 检测脚本执行情况 成功删除或者添加为0,报错信息为其他数字。
7. 能以，分割。一次性添加或者 删除多个用户。 例如 adddel.sh –add user1,user2,user3…….
8. 不允许存在明显bug。

代码参考：

```
#!/bin/bash
#written by aming.

if [ $# -eq 0 -o $# -gt 2 ]
then
    echo "use $0 --add username or $0 --del username or $0 --help."
    exit 1
fi

case $1 in
    --add)
        n=0
        for u in `echo $2|sed 's/,/ /g'`; do
            if awk -F: '{print $1}' /etc/passwd |grep -qw "$u"
            then
                echo "The user $u exist."
            else
                useradd $u
                echo -e "$u\n$u"|passwd $u >/dev/null 2>&1
                echo "The user $u added successfully."
                n=$[$n+1]
            fi
        done

        if [ $n -eq 0 ]; then
            exit 2
        fi
        ;;

    --del)
        n=0
        for u in `echo $2|sed 's/,/ /g'`; do
            if awk -F: '{print $1}' /etc/passwd|grep -qw "$u"
            then
               userdel -r $u
                echo "The user $u deleted successfully."
                n=$[$n+1]
            else
                echo "The user $u not exist."
            fi
        done

        if [ $n -eq 0 ]; then
                exit 3
        fi
        ;;

    --help)
        echo -e "--add can add user,and the passwd is the same as username.
    It can add multiuser such as --add user1,user2,user3..."
```

```
        echo "--del cat delete user.It can delete user such as --del user1,user2,user3..."
        ;;

    *)
        echo "use $0 --add username or $0 --del username or $0 --help."
        exit 1
        ;;
esac
```

【脚本38】被3整除

写一个脚本： 计算100以内所有能被3整除的正整数的和

代码参考：

```
#!/bin/bash
sum=0
for i in {1..100};do
    if [ $[$i%3] -eq 0 ];then
    sum=$[$i+$sum]
     fi
done

echo "sum:$sum"
```

【脚本39】脚本传参

使用传参的方法写个脚本，实现加减乘除的功能。例如： sh a.sh 1 2，这样会分别计算加、减、乘、除的结果。

要求：

1. 脚本需判断提供的两个数字必须为整数
2. 当做减法或者除法时，需要判断哪个数字大
3. 减法时需要用大的数字减小的数字
4. 除法时需要用大的数字除以小的数字，并且结果需要保留两个小数点。

参考代码：

```
#!/bin/bash

if [ $# -ne 2 ]
then
    echo "The number of parameter is not 2, Please useage: ./$0 1 2"
    exit 1
fi

is_int()
{
    if echo "$1"|grep -q '[^0-9]'
    then
    echo "$1 is not integer number."
    exit 1
    fi
}

max()
{
    if [ $1 -ge $2 ]
    then
    echo $1
    else
    echo $2
    fi
}
```

```
min()
{
    if [ $1 -lt $2 ]
    then
    echo $1
    else
    echo $2
    fi
}

sum()
{
    echo "$1 + $2 = $[$1+$2]"
}

minus()
{
    big=`max $1 $2`
    small=`min $1 $2`
    echo "$big - $small = $[$big-$small]"
}

mult()
{
    echo "$1 * $2 = $[$1*$2]"
}

div()
{
    big=`max $1 $2`
    small=`min $1 $2`
    d=`echo "scale =2; $big / $small"|bc`
    echo "$big / $small = $d"
}

is_int $1
is_int $2
sum $1 $2
minus $1 $2
mult $1 $2
div $1 $2
```

【脚本40】用户交互脚本

写一个脚本，执行后，打印一行提示“Please input a number:”，要求用户输入数值，然后打印出该数值，然后再次要求用户输入数值。直到用户输入”end”停止。

代码参考：

```
#!/bin/bash

while :
do
 read -p "Please input a number:(end for exit) " n
 num=` echo $n |sed -r 's/[0-9]//g'|wc -c `
 if [ $n == "end" ]
 then
      exit
 elif [ $num -ne 1 ]
 then
      echo "what you input is not a number!Try again!"
 else
      echo "your input number is: $n"
 fi
done
```

【脚本41】获取ip

提示用户输入网卡的名字，然后我们用脚本输出网卡的ip。 看似简单，但是需要考虑多个方面，比如我们输入的不符合网卡名字的规范，怎么应对。名字符合规范，但是根本就没有这个网卡有怎么应对。

代码参考：

```
#!/bin/bash
while :
do
        read -p "请输入网卡名: " e
        e1=`echo "$e" | sed 's/[-0-9]//g'`
        e2=`echo "$e" | sed 's/[a-zA-Z]//g'`
        if [ -z $e ]
        then
            echo "你没有输入任何东西"
            continue
        elif [ -z $e1 ]
        then
            echo "不要输入纯数字在centos中网卡名是以eth开头后面加数字"
            continue
        elif [ -z $e2 ]
        then
            echo "不要输入纯字母在centos中网卡名是以eth开头后面加数字"
            continue
        else
            break
        fi
done

ip() {
        ifconfig | grep -A1 "$1 " |tail -1 | awk '{print $2}' | awk -F ":" '{print $2}'
}

myip=`ip $e`
if [ -z $myip ]
then
    echo "抱歉，没有这个网卡。"
else
    echo "你的网卡IP地址是$myip"
fi
```

【脚本42】列出子目录

脚本的功能：
脚本可以带参数也可以不带，参数可以有多个，每个参数必须是一个目录，脚本检查参数个数，若等于0，则列出当前目录本身；否则，显示每个参数包含的子目录。

参考代码：

```
#!/bin/bash
if [ $# == 0 ]
then
    ls -ld `pwd`
else
    for i in `seq 1 $#`
    do
        a=$i
        echo "ls ${!a}"
        ls -l ${!a} |grep '^d'
    done
fi
```

标注：

你可能会对${!a}有疑问，这里是一个特殊用法，在shell中，$1为第一个参数，$2为第二个参数，以此类推，那么这里的数字要是一个变量如何表示呢？比

【脚本43】下载文件

创建一个函数，能接受两个参数：

1. 第一个参数为URL，即可下载的文件；第二个参数为目录，即下载后保存的位置；
2. 如果用户给的目录不存在，则提示用户是否创建；如果创建就继续执行，否则，函数返回一个51的错误值给调用脚本；
3. 如果给的目录存在，则下载文件；下载命令执行结束后测试文件下载成功与否；如果成功，则>返回0给调用脚本，否则，返回52给调用脚本；

提示，在函数中返回错误值给调用脚本，使用return

参考代码：

```
#!/bin/bash

if [ ! -d $2 ]
then
    echo "please make directory"
    exit 51
fi

cd $2
wget $1

n=`echo $?`
if [ $n -eq 0 ];then
    exit 0
else
    exit 52
fi
```

【脚本44】猜数字

写一个猜数字脚本，当用户输入的数字和预设数字（随机生成一个小于100的数字）一样时，直接退出，否则让用户一直输入，并且提示用户的数字比预设数字大或者小。

参考代码：

```
#!/bin/bash
m=`echo $RANDOM`
n1=$[$m%100]
while :
do
    read -p "Please input a number: " n
    if [ $n == $n1 ]
    then
        break
    elif [ $n -gt $n1 ]
    then
        echo "bigger"
        continue
    else
        echo "smaller"
        continue
    fi
done
echo "You are right."
```

【脚本45】抽签脚本

1、写一个脚本执行后，输入名字，产生随机数01-99之间的数字。
2、如果相同的名字重复输入，抓到的数字还是第一次抓取的结果，
3、前面已经抓到的数字，下次不能在出现相同数字。
4、第一个输入名字后，屏幕输出信息，并将名字和数字记录到文件里，程序不能退出继续等待别的学生输入。

参考代码：

```
while :
do
    read -p  "Please input a name:" name
    if [ -f /work/test/1.log ];then
    bb=`cat /work/test/1.log | awk -F: '{print $1}' | grep "$name"`

    if [ "$bb" != "$name" ];then   #名字不重复情况下
        aa=`echo $RANDOM | awk -F "" '{print $2 $3}'`
        while :
        do
          dd=`cat  /work/test/1.log |  awk -F: '{print $2}'  | grep "$aa"`
          if [ "$aa"  ==  "$dd" ];then    #数字已经存在情况下
              echo "数字已存在."
              aa=`echo $RANDOM | awk -F "" '{print $2 $3}'`
          else
              break
          fi
        done
        echo "$name:$aa" | tee -a /work/test/1.log
     else
     aa=`cat /work/test/1.log |  grep "$name" | awk -F: '{print $2}'` #名字重复
     echo $aa
     echo "重复名字."
     fi
  else
      aa=`echo $RANDOM | awk -F "" '{print $2 $3}'`
      echo "$name:$aa" | tee -a  /work/test/1.log
  fi
done
```

【脚本46】打印只有一个数字的行

如题，把一个文本文档中只有一个数字的行给打印出来。

参考代码：

```
#!/bin/bash

f=/etc/passwd
line=`wc -l $f|awk '{print $1}'`
for l in `seq 1 $line`; do
     n=`sed -n "$l"p $f|grep -o '[0-9]'|wc -l`;
     if [ $n -eq 1 ]; then
        sed -n "$l"p $f
     fi
done
```

【脚本47】日志归档

类似于日志切割，系统有个logrotate程序，可以完成归档。但现在我们要自己写一个shell脚本实现归档。

举例： 假如服务的输出日志是1.log，我要求每天归档一个，1.log第二天就变成1.log.1，第三天1.log.2, 第四天 1.log.3 一直到1.log.5

参考答案：

```
#!/bin/bash

function e_df()
{
    [ -f $1 ] && rm -f $1
}

for i in `seq 5 -1 2`
do
    i2=$[$i-1]
    e_df /data/1.log.$i
    if [ -f /data/1.log.$i2 ]
    then
        mv /data/1.log.$i2 /data/1.log.$i
    fi
done

e_df /data/1.log.1
mv /data/1.log  /data/1.log.1
```

【脚本48】找出活动ip

写一个shell脚本，把192.168.0.0/24网段在线的ip列出来。
思路： for循环， 0.1 — 0.254 依次去ping，能通说明在线。

参考代码：

```
#!/bin/bash

ips="192.168.1."
for i in `seq 1 254`
do

ping -c 2 $ips$i >/dev/null 2>/dev/null
if [ $? == 0 ]
then
    echo "echo $ips$i is online"
else
    echo "echo $ips$i is not online"
fi
done
```

【脚本49】检查错误

写一个shell脚本，检查指定的shell脚本是否有语法错误，若有错误，首先显示错误信息，然后提示用户输入q或者Q退出脚本，输入其他内容则直接用vim打开该shell脚本。

提醒： 检查shell脚本有没有语法错误的命令是 sh -n xxx.sh

参考代码：

```
#!/bin/bash
sh -n $1 2>/tmp/err
if [ $? -eq "0" ]
then
    echo "The script is OK."
else
    cat /tmp/err
    read -p "Please inpupt Q/q to exit, or others to edit it by vim. " n
```

```
        if [ -z $n ]
        then
            vim $1
            exit
        fi
        if [ $n == "q" -o $n == "Q" ]
        then
            exit
        else
            vim $1
            exit
        fi

    fi
```

【脚本50】格式化输出

输入一串随机数字，然后按千分位输出。

比如输入数字串为“123456789”，输出为123,456,789

代码参考：

```
#!/bin/bash

read -p "输入一串数字：" num
v=`echo $num|sed 's/[0-9]//g'`
if [ -n "$v" ]
then
    echo "请输入纯数字."
    exit
fi
length=${#num}
len=0
sum=''
for i in $(seq 1 $length)
do
        len=$[$len+1]
        if [[ $len == 3 ]]
        then
                sum=','${num:$[0-$i]:1}$sum
                len=0
        else
                sum=${num:$[0-$i]:1}$sum
        fi
done

if [[ -n $(echo $sum | grep '^,' ) ]]
then
        echo ${sum:1}
else
        echo $sum
fi
```

上面这个实现比较复杂，下面再来一个sed的：

```
#!/bin/bash
read -p "输入一串数字：" num
v=`echo $num|sed 's/[0-9]//g'`
if [ -n "$v" ]
then
    echo "请输入纯数字."
    exit
fi

echo $num|sed -r '{:number;s/([0-9]+)([0-9]{3})/\1,\2/;t number}'
```

【脚本51】

1 编写一个名为iffile程序，它执行时判断/bin目录下date文件是否存在？

参考代码：

```
#!/bin/bash
if [ -f /bin/date  ]
then
    echo "/bin/date file exist."
else
    echo "/bin/date not exist."
fi
```

2 编写一个名为greet的问候程序，它执行时能根据系统当前的时间向用户输出问候信息。设从半夜到中午为早晨，中午到下午六点为下午，下午六点到半夜为晚上。

参考代码：

```
#!/bin/bash
h=`date +%H`
if [ $h -ge 0 ] && [ $h -lt 12 ]
then
    echo "Good morning."
elif [ $h -ge 12 ] && [ $h -lt 18 ]
then
    echo "Good afternoon."
else
    echo "Good evening."
fi
```

【脚本52】判断用户登录

1 编写一个名为ifuser的程序，它执行时带用户名作为命令行参数，判断该用户是否已经在系统中登录，并给出相关信息。

参考代码：

```
#!/bin/bash
read -p "Please input the username: " user
if who | grep -qw $user
then
    echo $user is online.
else
    echo $user not online.
fi
```

2 编写一个名为menu的程序，实现简单的弹出式菜单功能，用户能根据显示的菜单项从键盘选择执行对应的命令。

参考代码：

```
#!/bin/bash
function message()
{
    echo "0. w"
    echo "1. ls"
    echo "2.quit"
    read -p "Please input parameter: " Par
}
message
while [ $Par -ne '2' ] ; do
    case $Par in
```

```
        0)
            w
            ;;
        1)
            ls
            ;;
        2)
            exit
            ;;
        *)
            echo "Unkown command"
            ;;
    esac
    message
done
```

【脚本53】更改后缀名

1 编写一个名为chname的程序，将当前目录下所有的.txt文件更名为.doc文件。

参考代码：

```
#!/bin/bash
find . -type f -name "*.txt" > /tmp/txt.list
for f in `cat /tmp/txt.list`
do
    n=`echo $f|sed -r 's/(.*)\.txt/\1/'`
    echo "mv $f $n.doc"
done
```

2 编写一个名为chuser的程序，执行中每隔5分钟检查指定的用户是否登录系统，用户名从命令行输入；如果指定的用户已经登录，则显示相关信息。

参考代码：

```
#!/bin/bash
read -p "Please input the username: " user
while :
do
    if who | grep -qw $user
    then
        echo $user login.
    else
        echo $user not login.
    fi
    sleep 300
done
```

【脚本54】判断pid是否一致

先普及一小段知识,我们用ps aux可以查看到进程的PID，而每个PID都会在/proc内产生。如果查看到的pid而proc内是没有的，则是进程被人修改了，这就代表你的系统很有可能已经被***过了。

请大家用上面知识编写一个shell，定期检查下自己的系统是否被人***过。

参考代码：

```
#!/bin/bash

ps aux|awk '/[0-9]/ {print $2}'|while read pid
do
    result=`find /proc/ -maxdepth 1 -type d -name "$pid"`
    if [ -z $result ]; then
```

```
        echo "$pid abnormal!"
    fi
done
```

【脚本55】一列变三行

比如1.txt内容：

```
1
2
3
4
5
6
7
```

处理后应该是：

```
1 2 3
4 5 6
7
```

可使用sed命令完成：

```
sed 'N;N;s/\n/ /g' 1.txt
```

【脚本56】shell的getops

写一个getinterface.sh 脚本可以接受选项[i，I]，完成下面任务：

1. 使用一下形式：getinterface.sh [-i interface | -I ip]
2. 当用户使用-i选项时，显示指定网卡的IP地址；当用户使用-I选项时，显示其指定ip所属的网卡。
   - 例：sh getinterface.sh -i eth0
   - sh getinterface.sh -I 192.168.0.1
3. 当用户使用除[-i | -I]选项时，显示[-i interface | -I ip]此信息。
4. 当用户指定信息不符合时，显示错误。（比如指定的eth0没有，而是eth1时）

参考代码：

```
#!/bin/bash

ip add |awk -F ":" '$1 ~ /^[1-9]/ {print $2}'|sed 's/ //g' > /tmp/eths.txt
[ -f /tmp/eth_ip.log ] && rm -f /tmp/eth_ip.log
for eth in `cat /tmp/eths.txt`
do
    ip=`ip add |grep -A2 ": $eth" |grep inet |awk '{print $2}' |cut -d '/' -f 1`
    echo "$eth:$ip" >> /tmp/eth_ip.log
done
useage()
{
    echo "Please useage: $0 -i 网卡名字 or $0 -I ip地址"
}

wrong_eth()
{
    if ! awk -F ':' '{print $1}' /tmp/eth_ip.log | grep -qw "^$1$"
    then
        echo "请指定正确的网卡名字"
        exit
    fi
}
```

```
wrong_ip()
{
    if ! awk -F ':' '{print $2}' /tmp/eth_ip.log | grep -qw "^$1$"
    then
        echo "请指定正确的ip地址"
        exit
    fi
}

if [ $# -ne 2 ]
then
    useage
    exit
fi

case $1 in
    -i)
    wrong_eth $2
    grep -w $2 /tmp/eth_ip.log |awk -F ':' '{print $2}'
    ;;

    -I)
    wrong_ip $2
    grep -w $2 /tmp/eth_ip.log |awk -F ':' '{print $1}'
    ;;

    *)
    useage
    exit
esac
```

【脚本57】3位随机数字

写一个脚本产生随机3位的数字，并且可以根据用户的输入参数来判断输出几组。 比如，脚本名字为 number3.sh。
执行方法：
bash number3.sh
直接产生一组3位数字。
bash number3.sh 10
插上10组3位数字。

思路： 可以使用echo $RANDOM获取一个随机数字，然后再除以10，取余获取0-9随机数字，三次运算获得一组。

参考代码：

```
#!/bin/bash

get_a_num() {
    n=$[$RANDOM%10]
    echo $n
}

get_numbers() {
    for i in 1 2 3; do
        a[$i]=`get_a_num`
    done
    echo ${a[@]}
}

if [ -n "$1" ]; then
    m=`echo $1|sed 's/[0-9]//g'`
    if [ -n "$m" ]; then
        echo "Useage bash $0 n, n is a number, example: bash $0 5"
        exit
    else
        for i in `seq 1 $1`
```

```
        do
            get_numbers
        done
    fi
else
    get_numbers

fi
```

【脚本58】检查服务

先判断是否安装http和mysql，没有安装进行安装，安装了检查是否启动服务，若没有启动则需要启动服务。

说明：操作系统为centos6，httpd和mysql全部为rpm包安装。

参考代码：

```
#!/bin/bash

if_install()
{
    n=`rpm -qa|grep -cw "$1"`
    if [ $n -eq 0 ]
    then
    echo "$1 not install."
    yum install -y $1
    else
    echo "$1 installed."
    fi
}

if_install httpd
if_install mysql-server

chk_ser()
{
    p_n=`ps -C "$1" --no-heading |wc -l`
    if [ $p_n -eq 0 ]
    then
    echo "$1 not start."
    /etc/init.d/$1 start
    else
    echo "$1 started."
    fi
}

chk_httpd
chk_mysqld
```

【脚本59】判断日期是否合法

用shell脚本判断输入的日期是否合法。就是判断日期是都是真实的日期，比如20170110就是合法日期，20171332就不合法

参考代码：

```
#!/bin/bash

#check date

if [ $# -ne 1 ] || [ ${#1} -ne 8 ]
then
    echo "Usage: bash $0 yyyymmdd"
    exit 1
fi
```

```
datem=$1
year=${datem:0:4}
month=${datem:4:2}
day=${datem:6:2}

if echo $day|grep -q '^0'
then
    day=`echo $day |sed 's/^0//'`
fi

if cal $month $year >/dev/null 2>/dev/null
then
    daym=`cal $month $year|egrep -v "$year|Su"|grep -w "$day"`
    if [ "$daym" != "" ]
    then
        echo ok
    else
        echo "Error: Please input a wright date."
    exit 1
    fi
else
    echo "Error: Please input a wright date."
    exit 1
fi
```

【脚本60】监控网卡

1.每10分钟检测一次指定网卡的流量
2.如果流量为0，则重启网卡

参考代码：

```
#!/bin/bash

LANG=en
n1=`sar -n DEV 1 60 |grep eth0 |grep -i average|awk '{print $5}'|sed 's/\.//g'`
n2=`sar -n DEV 1 60 |grep eth0 |grep -i average|awk '{print $6}'|sed 's/\.//g'`
if [ $n1 == "000" ] && [ $n2 == "000" ]
then
    ifdown eth0
    ifup eth0
fi
```

然后写个cron，10分钟执行一次

【脚本61】监控web可用性

写一个shell脚本，通过curl -I 返回的状态码来判定所访问的网站是否正常。比如，当状态码为200时，才算正常。

参考代码：

```
#/bin/bash

url="http://www.apelearn.com/index.php"
sta=`curl -I $url 2>/dev/null |head -1 |awk '{print $2}'`

if [ $sta != "200" ]
then
    python /usr/local/sbin/mail.py xxx@qq.com "$url down." "$url down"
fi
```

【脚本62】文件打包

需求：将用户家目录（考虑到执行脚本的用户可能是普通用户也可能是root）下面小于5KB的文件打包成tar.gz的压缩包，并以当前日期为文件名前缀，例如今天打包的文件为2017-09-15.tar.gz。

参考代码：

```
#!/bin/bash

t=`date +%F`
cd $HOME
tar czf $t.tar.gz `find . -type f -size -5k`
```

【脚本63】端口解封

一个小伙伴提到一个问题，他不小心用iptables规则把sshd端口22给封掉了，结果不能远程登陆，要想解决这问题，还要去机房，登陆真机去删除这规则。 问题来了，要写个监控脚本，监控iptables规则是否封掉了22端口，如果封掉了，给打开。 写好脚本，放到任务计划里，每分钟执行一次。

参考代码：

```
#!/bin/bash
# check sshd port drop

/sbin/iptables -nvL --line-number|grep "dpt:22"|awk -F ' ' '{print $4}' > /tmp/drop.txt
i=`cat /tmp/drop.txt|head -n 1|egrep -iE "DROP|REJECT"|wc -l`
if [ $i -gt 0 ]
then
    /sbin/iptables -I INPUT 1 -p tcp --dport 22 -j ACCEPT
fi
```

【脚本64】统计分析日志

已知nginx访问的日志文件在/usr/local/nginx/logs/access.log内

请统计下早上10点到12点 来访ip最多的是哪个?

日志样例：

```
111.199.186.68 - [15/Sep/2017:09:58:37 +0800]  “//plugin.php?id=security:job” 200 “POST //plugin.php?id=security:job HTTP/1

203.208.60.208 - [15/Sep/2017:09:58:46 +0800] “/misc.php?mod=patch&action=ipnotice&_r=0.05560809863330207&inajax=1&ajaxtarge
```

实现这个需求使用如下命令即可：

```
grep '15/Sep/2017:1[0-2]:[0-5][0-9]:' /usr/local/nginx/logs/access.log|awk '{print $1}'|sort -n|uniq -c |sort -n|tail -n1
```

【脚本65】打印数字

写一个shell脚本。提示你输入一个暂停的数字，然后从1打印到该数字。然后询问是否继续。继续的话在输入个在数字 接着打印。不继续退出。

例：如果输入的是5，打印1 2 3 4 5 然后继续 输入15 然后打印 6 7 …14 15 依此类推。

参考代码：

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

```
read -p "请输入您想要暂停的数字：" number_1

for i in `seq 1 $number_1`;
do
        echo $i
done

read -p "是否继续输入数字？" a
if [ $a == "yes" ];then
        read -p "请继续输入您想要暂停的数字：" number_2
        number_3=$[$number_1+1]
        if [ $number_2 -gt $number_1 ];then
                for h in `seq $number_3 $number_2`;
                do
                        echo $h
                done
        else
                echo "输入数字错误，请输入大于的数字!"
        fi
else
        exit
fi
```

【脚本66】给文档增加内容

在文本文档1.txt第5行（假设文件行数大于5）后面增加如下内容：

```
# This is a test file.

# Test insert line into this file.
```

参考命令：

```
sed -i "5a # This is a test file.\n# Test insert line into this file." 1.txt
```

【脚本67】备份etc下面文件

设计一个shell程序，在每月第一天备份并压缩/etc目录的所有内容，存放在/root/bak目录里，且文件名为如下形式”yymmdd_etc.tar.gz”，yy为年，mm为月，dd为日。

参考代码：

```
#!/bin/sh

if [ ! -d /root/bak ]
then
    mkdir /root/bak
fi
prefix=`date +%y%m%d`
d=`date +%d`
if [ $d == "01" ]
then
    cd /etc/
    tar czf  /root/bak/$prefix_etc.tar.gz ./
fi
```

【脚本68】计算单词重复次数

将文件内所有的单词的重复次数计算出来，只需要列出重复次数最多的10个单词。

假设文档名字叫做a.txt，使用如下命令即可：

```
sed 's/[^a-zA-Z]/ /g' a.txt|xargs -n1 |sort |uniq -c |sort -nr |head
```

【脚本69】成员分组

需求是，把所有的成员平均得分成若干个小组。这里，我会提供一个人员列表，比如成员有50人，需要分成7个小组，要求随机性，每次和每次分组的结构应该不一致。

假设成员列表文件为members.txt

参考代码：

```
#!/bin/bash
f=members.txt
n=`wc -l $f|awk '{print $1}'`
get_n()

{
    l=`echo $1|wc -c`
    n1=$RANDOM
    n2=$[$n1+$l]
    g_id=$[$n1%7]
    if [ $g_id -eq 0 ]
    then
        g_id=7
    fi
    echo $g_id
}

for i in `seq 1 7`
do
    [ -f n_$i.txt ] && rm -f n_$i.txt
done
for i in `seq 1 $n`
do
    name=`sed -n "$i"p $f`
    g=`get_n $name`
    echo $name >> n_$g.txt
done

nu(){
    wc -l $1|awk '{print $1}'
}

max(){
    ma=0
    for i in `seq 1 7`
    do
        n=`nu n_$i.txt`
        if [ $n -gt $ma ]
        then
            ma=$n
       fi
    done
    echo $ma
}

min(){
    mi=50
    for i in `seq 1 7`
    do
       n=`nu n_$i.txt`
       if [ $n -lt $mi ]
       then
           mi=$n
```

```
        fi
    done
    echo $mi
}

ini_min=1
while [ $ini_min -le 7 ]
do
    m1=`max`
    m2=`min`
    ini_min=m2
    for i in `seq 1 7`
    do
        n=`nu n_$i.txt`
        if [ $n -eq $m1 ]
        then
            f1=n_$i.txt
        elif [ $n -eq $m2 ]
        then
          f2=n_$i.txt
        fi
    done
    name=`tail -n1 $f1`
    echo $name >> $f2
    sed -i "/$name/d" $f1
    ini_min=$[$ini_min+1]
done

for i in `seq 1 7`
do
    echo "$i 组成员有："
    cat n_$i.txt
    echo
done
```

【脚本70】shell中的小数

有一组式子如下：

```
a=0.5
b=3
c=a*b
```

求c的值，参考代码：

```
#!/bin/bash
a=0.5
b=3
c=`echo "scale=1;$a*$b"|bc`
echo $c
```

【脚本71】a.txt有b.txt没有

有两个文件a.txt和b.txt，需求是，把a.txt中有的并且b.txt中没有的行找出来，并写入到c.txt，然后计算c.txt文件的行数。

参考代码：

```
#!/bin/bash

n=`wc -l a.txt|awk '{print $1}'`
[ -f c.txt ] && rm -f c.txt
for i in `seq 1 $n`
do
```

```
    l=`sed -n "$i"p a.txt`
    if ! grep -q "^$l$" b.txt
    then
    echo $l >>c.txt
    fi
done
wc -l c.txt
```

或者用grep实现

```
grep -vwf b.txt a.txt > c.txt;  wc -l c.txt
```

【脚本72】杀死进程

把当前用户下所有进程名字中含有"java"的进程关闭。

参考答案：

```
ps -u $USER |awk '$NF ~ /java/ {print $1}'|xargs kill
```

【脚本73】备份数据表

用shell实现，以并发进程的形式将mysql数据库所有的表备份到当前目录，并把所有的表压缩到一个压缩包文件里。

假设数据库名字为mydb，用户名为zero，密码为passwd。

提示： 在shell中加上&可以将命令丢到后台，从而可以同时执行多条命令达到并发的效果。

参考代码：

```
#!/bin/bash
pre=`date +%F`
for d in `mysql -uaming -ppasswd mydb -e "show tables"|grep -v 'Tables_in_'`
do
    mysqldump -uaming -ppasswd mydb $d > $d.sql &
done
tar czf $pre.tar.gz *.sql
rm -f *.sql
```

【脚本74】监控节点

一个网站，使用了cdn，全国各地有几十个节点。需要你写一个shell脚本来监控各个节点是否正常。

假如：

1. 监控的url为www.xxx.com/index.php
2. 源站ip为88.88.88.88

参考代码：

```
#!/bin/bash

url="www.xxx.com/index.php"
s_ip="88.88.88.88"
curl -x $s_ip:80 $url > /tmp/source.html 2>/dev/null

for ip in `cat /tmp/ip.txt`
do
    curl -x $ip:80 $url 2>/dev/null >/tmp/$ip.html
    [ -f /tmp/$ip.diff ] && rm -f /tmp/$ip.diff
    touch /tmp/$ip.diff
    diff /tmp/source.html /tmp/$ip.html > /tmp/$ip.diff 2>/dev/null
```

```
    n=`wc -l /tmp/$ip.diff|awk '{print $1}'`
    if [ $n -lt 0 ]
    then
        echo "node $ip sth wrong."
    fi
done
```

【脚本75】破解字符串

已知下面的字符串是通过RANDOM随机数变量md5sum|cut-c 1-8截取后的结果，请破解这些字符串对应的md5sum前的RANDOM对应数字？

```
21029299
00205d1c
a3da1677
1f6d12dd
890684ba
```

解题思路：通过每次传递一个参数的方式，来实现依次破解，$RANDOM的范围为0-32767。

参考代码：

```
#!/bin/bash

for n in {0..32767}
do
        MD5=`echo $n | md5sum | cut -c 1-8`
        if [ "$MD5" == "$1" ];then
            echo "$n $1 "
            break
        fi
done
```

【脚本76】判断cpu厂商

写一个脚本：

1. 判断当前主机的CPU生产商，其信息在/proc/cpuinfo文件中vendor id一行中。
2. 如果其生产商为AuthenticAMD，就显示其为AMD公司；
3. 如果其生产商为GenuineIntel，就显示其为Intel公司；
4. 否则，就说其为非主流公司。

参考代码：

```
#!/bin/bash

m=`cat /proc/cpuinfo |grep vendor_id|awk  -F":" '{print $2}'|tail -1`
if [ $m == "GenuineIntel" ]
then
      echo "cpu is 英特尔"
elif [ $m == "AuthenticAMD" ]
then
      echo "cpu is AMD"
else
      echo "cpu is 非主流"
fi
```

【脚本77】监控cpu使用率

用shell写一个监控服务器cpu使用率的监控脚本。

思路：用top -bn1 命令，取当前空闲cpu百份比值（只取整数部分），然后用100去剑这个数值。

参考代码：

```
#!/bin/bash

while :
do
    idle=`top -bn1 |sed -n '3p' |awk '{print $5}'|cut -d . -f1`
    use=$[100-$idle]
    if [ $use -gt 90 ]
    then
        echo "cpu use percent too high."
        #发邮件省略
    fi
    sleep 10
done
```

【脚本78】获取子进程

说明：本shell题目是一个网友在公众号中提问的，正好利用这个每日习题的机会拿出来让大家一起做一做。

给出一个进程PID，打印出该进程下面的子进程以及子进程下面的所有子进程。（只需要考虑子进程的子进程，再往深层次则不考虑）

参考代码：

```
#!/bin/bash

read -p "please input a pid number: " p
ps -elf > /tmp/ps.log

is_ppid(){
    awk '{print $5}' /tmp/ps.log > /tmp/ps1.log
    if ! grep -qw "$1" /tmp/ps1.log
    then
        echo "PID $1 不是系统进程号，或者它不是父进程"
    return 1
    fi
}

is_ppid $p

if [ $? -eq "1" ]
then
    exit
fi

print_cpid(){
    p=$1
    awk -v p1=$p '$5 == p1 {print $4}' /tmp/ps.log |sort -n |uniq >/tmp/p1.log
    n=`wc -l /tmp/p1.log|awk '{print $1}'`
    if [ $n -ne 0 ]
    then
        echo "PID $p 子进程 pid 如下:"
        cat /tmp/p1.log
    else
    echo "PID $p 没有子进程"
    fi
}

print_cpid $p

for cp in `cat /tmp/p1.log`
do
    print_cpid $cp
done
```

另外，一条命令查询的方法是：

```
pstree -p pid
```

【脚本79】自动添加项目

需求背景：
服务器上，跑的lamp环境，上面有很多客户的项目，每个项目就是一个网站。 由于客户在不断增加，每次增加一个客户，就需要配置相应的mysql、ftp以及httpd. 这种工作是重复性非常强的，所以用脚本实现非常合适。

mysql增加的是对应客户项目的数据库、用户、密码，ftp增加的是对应项目的用户、密码（使用vsftpd，虚拟用户模式），httpd就是要增加虚拟主机配置段。

参考代码：

```
#!/bin/bash

webdir=/home/wwwroot
ftpudir=/etc/vsftpd/vuuser
mysqlc="/usr/bin/mysql -uroot -xxxxxx"
httpd_config_f="/usr/local/apache2/conf/extra/httpd-vhosts.conf"

add_mysql_user()
{
        mysql_p=`mkpasswd -s 0 -l 12`
        echo "$pro $mysql_p" >/tmp/$pro.txt
        $mysqlc <<EOF
        grant all on $p.* to "$pro"@'127.0.0.1' identified by "$mysql_p";
EOF
}

add_ftp_user()

{
        ftp_p=`mkpasswd -s 0 -l 12`
        echo "$pro" >> /root/login.txt
        echo "$ftp_p" >> /root/login.txt
        db_load -T -t hash -f /root/login.txt  /etc/vsftpd/vsftpd_login.db
        cd $ftpudir
        cp aaa $pro    //这里的aaa是一个文件，是之前的一个项目，可以作为配置模板
        sed -i "s/aaa/$pro/" $pro   //把里面的aaa改为新的项目名字
        /etc/init.d/vsftpd restart
}

config_httpd()

{
        mkdir $webdir/$pro
        chown vsftpd:vsftpd $webdir/$pro
        echo -e "<VirtualHost *:80> \n    DocumentRoot "/home/internet/www/$pro/" \n    ServerName $dom \n    #ServerAlias \n</VirtualHost> " >
        /usr/local/apache2/bin/apachectl graceful
}

read -p "input the project name: " pro
read -p "input the domain: " dom

add_mysql_user
add_ftp_user
config_httpd
```

【脚本80】计算器

用shell写一个简易计算器，可以实现加、减、乘、除运算，假如脚本名字为1.sh，执行示例：./1.sh 1 + 2

参考代码：

```
#!/bin/bash

if [ $# -ne 3 ]
then
    echo "参数个数不为3"
    echo "当使用乘法时，需要加上脱义符号，例如 $0 1 \* 2"
    exit 1;
fi

num1=`echo $1|sed 's/[0-9.]//g'` ;
if [ -n "$num1" ]
then
    echo "$1 不是数字" ;
    exit 1
fi

num3=`echo $3|sed 's/[0-9.]//g'` ;
if [ -n "$num3" ]
then
    echo "$3 不是数字" ;
    exit 1
fi

case $2 in
  +)
    echo "scale=2;$1+$3" | bc
    ;;

  -)
    echo "scale=2;$1-$3" | bc
    ;;

  \*)
    echo "scale=2;$1*$3" | bc
    ;;

  /)
    echo "scale=2;$1/$3" | bc
    ;;

  *)
   echo  "$2 不是运算符"
   ;;
esac
```

【脚本81】判断没有文件

判断所给目录内哪些二级目录下没有text.txt文件。

有text.txt文件的二级目录，根据文件计算选项中单词数最大的值（选项间以|分割，单词间以空格分隔）。

假如脚本名字为1.sh， 运行脚本的格式为 ./1.sh 123 root，其中123为目录名字，而root为要计算数量的单词。

说明： 这个shell脚本题目出的有点歧义。 原题给的描述不是很清楚，我另外又改了一下需求，依然不是很清晰。在这里我再做一个补充： 对于有test.txt的目录，计算出该test.txt文件里面所给出单词的次数。不用找最大。

参考代码：

```
#!/bin/bash

if [ $# -ne 2 ]
then
    echo "useage $0 dir word"
```

```
    exit 1
fi

if [ -d $1 ]
then
    cd $1
else
    echo "$1目录不存在"
    exit 1
fi

for f in `ls $1`
do
    if [ -d $f ]
    then
    if [ -f $f/test.txt ]
    then
        n=`grep -cw "$2" $f/test.txt`
        echo "$1/$f/test.txt 里面有$n个$2"
    else
        echo "$1/$f 下面没有test.txt"
        fi
    fi
done
```

【脚本82】打印正方形

交互式脚本，根据提示，需要用户输入一个数字作为参数，最终打印出一个正方形。

在这里我提供一个linux下面的特殊字符■，可以直接打印出来。

示例： 如果用户输入数字为5，则最终显示的效果为：

■ ■ ■ ■ ■

■ ■ ■ ■ ■

■ ■ ■ ■ ■

■ ■ ■ ■ ■

■ ■ ■ ■ ■

参考代码：

```
#!/bin/bash

read -p "please input a number:" sum
a=`echo $sum |sed 's/[0-9]//g'`
if [ -n "$a" ]
then
    echo "请输入一个纯数字。"
    exit 1
fi

for n in `seq $sum`
do
    for m in `seq $sum`
    do
        if [ $m -lt $sum ]
        then
            echo -n "■ "
        else
            echo "■"
        fi
```

```
    done
done
```

【脚本83】问候用户

写一个脚本，依次向/etc/passwd中的每个用户问好，并且说出对方的ID是什么：

```
Hello,root, your UID is 0.
```

参考命令：

```
awk -F ':' '{print "Hello,"$1",your uid is "$3.}' /etc/passwd
```

【脚本84】按要求处理文本

linux系统 /home目录下有一个文件test.xml，内容如下：

```
<configuration>
    <artifactItems>
        <artifactItem>
        <groupId>zzz</groupId>
        <artifactId>aaa</artifactId>
    </artifactItem>
    <artifactItem>
        <groupId>xxx</groupId>
        <artifactId>yyy</artifactId>
    </artifactItem>
    <!-- </artifactItem><groupId>some groupId</groupId>
    <version>1.0.1.2.333.555</version> </artifactItem>-->
    </artifactItems>
</configuration>
```

请写出shell脚本删除文件中的注释部分内容，获取文件中所有artifactItem的内容，并用如下格式逐行输出 artifactItem：groupId：artifactId

分析：这个文件比较特殊，但是却很有规律。注释部分内容其实就是<!– –>中间的内容，所以我们想办法把这些内容删除掉就ok了。而artifactItem的内容，其实就是获取<artifactItem></artifactItem>中间的内容。然后想办法用提到的格式输出即可。

参考代码：

```
#!/bin/bash

egrep -v '<!--|-->' 1.txt |tee 2.txt  //这行就是删除掉注释的行
grep -n 'artifactItem>' 2.txt |awk '{print $1}' |sed 's/://' > /tmp/line_number.txt
n=`wc -l /tmp/line_number.txt|awk '{print $1}'`

get_value(){
    sed -n "$1,$2"p 2.txt|awk -F '<' '{print $2}'|awk -F '>' '{print $1,$2}' > /tmp/value.txt
    nu=`wc -l /tmp/value.txt|awk '{print $1}'`
    for i in `seq 1 $nu`
    do
        x=`sed -n "$i"p /tmp/value.txt|awk '{print $1}'`
        y=`sed -n "$i"p /tmp/value.txt|awk '{print $2}'`
        echo artifactItem:$x:$y
    done
}

n2=$[$n/2]

for j in `seq 1 $n2`
do
    m1=$[$j*2-1]
    m2=$[$j*2]
```

```
    nu1=`sed -n "$m1"p /tmp/line_number.txt`
    nu2=`sed -n "$m2"p /tmp/line_number.txt`
    nu3=$[$nu1+1]
    nu4=$[$nu2-1]
    get_value $nu3 $nu4
done
```

【脚本85】判断函数

请使用条件函数if撰写一个shell函数 函数名为 f_judge，实现以下功能：

1. 当/home/log 目录存在时 将/home目录下所有tmp开头的文件或目录移/home/log 目录。
2. 当/home/log目录不存在时，创建该目录，然后退出。

参考代码：

```
#!/bin/bash

f_judge (){
    if [ -d /home/log ]
    then
        mv /home/tmp* /home/log/
    else
        mkdir -p /home/log
        exit
    fi
}
```

【脚本86】批量杀进程

linux系统中，根目录/root/下有一个文件ip-pwd.ini，内容如下：

```
10.111.11.1,root,xyxyxy

10.111.11.1,root,xzxzxz

10.111.11.1,root,123456

10.111.11.1,root,xxxxxx

……
```

文件中每一行的格式都为linux服务器的ip,root用户名,root密码，请用一个shell批量将这些服务器中的所有tomcat进程kill掉。

讲解： 有了ip，用户名和密码，剩下的就是登录机器，然后执行命令了。批量登录机器，并执行命令，咱们课程当中有讲过一个expect脚本。所以本题就是需要这个东西来完成。

首先编辑expect脚本 kill_tomcat.expect：

```
#!/usr/bin/expect
set passwd [lindex $argv 0]
set host [lindex $argv 1]
spawn ssh root@$host

expect {
    "yes/no" { send "yes\r"; exp_continue}
    "password:" { send "$passwd\r" }
}

expect "]*"
send "killall java\r"
expect "]*"
send "exit\r"
```

编辑完后需要给这个文件执行权限：

```
chmod a+x kill_tomcat.expect
```

然后编辑shell脚本：

```
#!/bin/bash
n=`wc -l ip-pwd.ini`
for i in `seq 1 $n`
do
    ip=`sed -n "$n"p ip-pwd.ini |awk -F ',' '{print $1}'`
    pw=`sed -n "$n"p ip-pwd.ini |awk -F ',' '{print $3}'`
    ./kill_tomcat.expect $pw $ip
done
```

【脚本87】处理日志

写一个脚本查找/data/log目录下，最后创建时间是3天前，后缀是*.log的文件，打包后发送至192.168.1.2服务上的/data/log下，并删除原始.log文件，仅保留打包后的文件

参考代码：

```
#!/bin/bash

find /data/log -name “*.log” -mtime +3 > /tmp/file.list
cd /data/log
tar czvf log.tar.gz `cat /tmp/file.list|xargs`
rsync -a log.tar.gz  192.168.1.2:/data/log  # 这一步需要提前做一个免密码登录
for f in `cat /tmp/file.list`
do
    rm -f $f
done
```

【脚本88】处理文本

有如下文本，其中前5行内容为

```
1111111:13443253456
2222222:13211222122
1111111:13643543544
3333333:12341243123
2222222:12123123123
```

用shell脚本处理后，按下面格式输出：

```
[1111111]
13443253456
13643543544
[2222222]
13211222122
12123123123
[3333333]
12341243123
```

参考代码：

```
#! /bin/bash

sort -n filename |awk -F ':' '{print $1}'|uniq >id.txt
```

```
for id in `cat id.txt`; do
        echo "[$id]"
        awk -v id2=$id -F ':' '$1==id2 {print $2}' filename
        #另外的方式为: awk -F ':' '$1=="$id"' {print $2}' filename
done
```

【脚本89】清理日志

要求：两类机器一共300多台，写个脚本自动清理这两类机器里面的日志文件。在堡垒机批量发布，也要批量发布到crontab里面。

A类机器日志存放路径很统一，B类机器日志存放路径需要用*匹配（因为这个目录里除了日志外，还有其他文件，不能删除。匹配的时候可用*.log）

A类：/opt/cloud/log/ 删除7天前的
B类: /opt/cloud/instances/ 删除15天前的

要求写在一个脚本里面。不用考虑堡垒机上的操作，只需要写出shell脚本。

参考代码：

```
#!/bin/bash

dir1=/opt/cloud/instances/
dir2=/opt/cloud/log/

  if [ -d $dir1 ];then
      find $dir1 -type f -name "*.log" -mtime +15 |xargs rm -f
  elif [ -d $dir2 ];then
      find $dir2 -type f -mtime +7 |xargs rm -f
  fi
```

【脚本90】

贷款有两种还款的方式：等额本金法和等额本息法

简单说明一下等额本息法与等额本金法的主要区别：

等额本息法的特点是：每月的还款额相同，在月供中“本金与利息”的分配比例中，前半段时期所还的利息比例大、本金比例小，还款期限过半后逐步转为本金比例大、利息比例小。所支出的总利息比等额本金法多，而且贷款期限越长，利息相差越大。

等额本金法的特点是：每月的还款额不同，它是将贷款额按还款的总月数均分（等额本金），再加上上期剩余本金的月利息，形成一个月还款额，所以等额本金法第一个月的还款额最多 ，尔后逐月减少，越还越少。所支出的总利息比等额本息法少。

两种还款方式的比较不是我们今天的讨论范围，我们的任务就是做一个贷款计算器。

其中：等额本息每月还款额的计算公式是：
[贷款本金×月利率×（1+月利率）^还款月数]÷[（1+月利率）^还款月数 - 1]

参考代码：

```
#!/bin/bash

read -p "请输入贷款总额（单位：万元）：" dkzewy
read -p "请输入贷款年利率（如年利率为6.5%，直接输入6.5）：" dknll
read -p "请输入贷款年限（单位：年）：" dknx
echo "贷款计算方式："
echo "1)等额本金计算法"
echo "2)等额本息计算法"
read -p "请选择贷款方式（1|2）" dkfs
dkze=`echo "scale=2;$dkzewy*10000 " | bc -l`
dkll=`echo "scale=6;$dknll/100 " | bc -l`
dkyll=`echo "scale=6;$dkll/12 " | bc -l`
dkqc=$[$dknx*12]
```

```
echo "期次 本月还款额 本月利息 未还款额"

debjjsf()
{
    yhbj=`echo "scale=2;($dkze/$dkqc)/1 " | bc -l`
    whbj=$dkze
    for((i=1;i<=$dkqc;i++))
    do
        bylx=`echo "scale=2;($whbj*$dkyll)/1 " | bc -l`
        bybx=`echo "scale=2;($yhbj+$bylx)/1 " | bc -l`
        yhke=`echo "scale=2;($yhbj*$i)/1 " | bc -l`
        whbj=`echo "$dkze-$yhke " | bc -l`
        if [ $i -eq $dkqc ]
        then
            yhbj=`echo "scale=2;($yhbj+$whbj)/1 " | bc -l`
            whbj="0.00"
            bybx=`echo "scale=2;($yhbj+$bylx)/1 " | bc -l`
        fi
        echo "$i $bybx $bylx $whbj"
    done
}

debxjsf()
{
    bybx=`echo "scale=2;(($dkze*$dkyll*((1+$dkyll)^$dkqc))/(((1+$dkyll)^$dkqc)-1))/1 " | bc -l`
    whbj=$dkze
    for((i=1;i<=$dkqc;i++))
    do
        bylx=`echo "scale=2;($whbj*$dkyll)/1 " | bc -l`
        yhbj=`echo "scale=2;($bybx-$bylx)/1 " | bc -l`
        whbj=`echo "scale=2;($whbj-$yhbj)/1 " | bc -l`
        if [ $i -eq $dkqc ]
        then
            bybx=`echo "scale=2;($yhbj+$whbj)/1 " | bc -l`
            whbj="0.00"
        fi
        echo "$i $bybx $bylx $whbj"
    done
}

case $dkfs in
    1) debjjsf
       ;;
    2) debxjsf
       ;;
    *) exit 1
       ;;
esac
```

【脚本91】监控磁盘io

阿里云的机器，今天收到客服来的电话，说服务器的磁盘io很重。于是登录到服务器查看，并没有发现问题，所以怀疑是间歇性地。

正要考虑写个脚本的时候，幸运的抓到了一个线索，造成磁盘io很高的幕后黑手是mysql。此时去show processlist，但未发现队列。原来只是一瞬间。

只好继续来写脚本，思路是，每5s检测一次磁盘io，当发现问题去查询mysql的processlist。

提示：你可以用iostat -x 1 5 来判定磁盘的io，主要看%util

参考代码：

```
#!/bin/bash

while :
do
    n=`iostat -x 1 5 |tail -n3|head -n1 |awk '{print $NF}'|cut -d. -f1`
```

```
    if [ $n -gt 70 ]
    then
        echo "`date` util% is $n%" >>/tmp/mysql_processlist.log
        mysql -uroot -pxxxxxx -e "show full processlist" >> /tmp/mysql_processlist.log
    fi
    sleep 5
done
```

【脚本92】截取tomcat日志

写一个截取tomcat catalina.out日志的脚本。

tomcat实例t1-t4：

```
[root@server ~]# tree -L 1 /opt/TOM/
/opt/TOM/
├── crontabs
├── t1
├── t2
├── t3
└── t4

5 directories, 0 files
```

catalina.out日志路径：

```
[root@server ~]# find /opt/TOM/ -name catalina.out
/opt/TOM/t1/logs/catalina.out
/opt/TOM/t3/logs/catalina.out
/opt/TOM/t4/logs/catalina.out
/opt/TOM/t2/logs/catalina.out
```

要求：

1. 这个脚本可以取tomcat实例t1-t4的日志
2. 这个脚本可以自定义取日志的起始点，比如取今天早上10点之后到现在的数据
3. 这个脚本可以自定义取日志的起始点和终点，比如取今天早上9点到晚上8点的数据

catalina.out 日志片段：

```
Mar 29, 2016 1:52:24 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler [“http-bio-8080”]
Mar 29, 2016 1:52:24 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler [“ajp-bio-8009”]
Mar 29, 2016 1:52:24 PM org.apache.catalina.startup.Catalina start
INFO: Server startup in 2102 ms
```

参考代码：

```
#!/bin/bash

export LANG=en_US.UTF-8
export PATH=$PATH
IPADD=`/sbin/ifconfig | grep "inet addr" | head -1 | awk '{print $2}'| awk -F '.' '{print $NF}'`
LOGFILE="/opt/TOM/$1/logs/catalina.out"
YEAR=`date +%Y`
DATE=`date +%m%d_%H%M`
TOMCAT=$1
BEGIN_TIME=$YEAR$2
END_TIME=$YEAR$3

##judge is  a.m.or p.m.
TIME_HOUR1=`echo  ${BEGIN_TIME:9:2}`

cut_log() {
```

```
        N_DATE1=`echo $1 | sed 's/_/ /g'`
        D_DATE1=`echo $2 | sed 's/_/ /g'`
        E_DATE1=`echo $3 | sed 's/_/ /g'`
        [ $4 ] && N_DATE2=`echo $4 | sed 's/_/ /g'`
        [ $5 ] && D_DATE2=`echo $5 | sed 's/_/ /g'`
        [ $6 ] && E_DATE2=`echo $6 | sed 's/_/ /g'`
        BEGIN=`grep -nE "${N_DATE1}|${D_DATE1}|${E_DATE1}" ${LOGFILE} | head -1 | cut -d : -f1`
        [ "$N_DATE2" ] && END=`grep -nE "${N_DATE2}|${D_DATE2}|${E_DATE2}" ${LOGFILE} | tail -1 | cut -d : -f1`

        [ ! -z "${TIME_HOUR1}" ] && if [ ${TIME_HOUR1} -gt 12 ] ; then
                BEGIN1=`grep -nE "${N_DATE1}|${D_DATE1}|${E_DATE1}" ${LOGFILE} |grep " PM " |grep "${E_DATE1}" | head -1 | cut

                if [ ! -z "${BEGIN1}" ] ; then
                [ "${BEGIN1}" -gt "${BEGIN}" ] ; BEGIN=${BEGIN1}
                fi
        fi

        if [ "$BEGIN" ] && [ -z "$END" ] ; then
                if [ "$N_DATE2" ]; then
                        echo   "${END_TIME}时间点没有访问日志，请重新设置时间点."
                else
                        sed -n "${BEGIN},[       DISCUZ_CODE_0       ]quot;p ${LOGFILE} > /home/gcweb/${IPADD}_${TOMCAT}_${DATE}.log
        fi
    elif [ "$END" ];then
        [ "$BEGIN" ] || BEGIN=1
        sed -n "${BEGIN},${END}"p ${LOGFILE} > /home/gcweb/${IPADD}_${TOMCAT}_${DATE}.log
    else
        [ "$END_TIME" != "$YEAR" ] && echo "该时段 ${BEGIN_TIME}～${END_TIME} 没有日志."
        [ "$END_TIME" = "$YEAR" ] && echo "该时段 ${BEGIN_TIME}～now 没有日志."
    fi

    if [ -s /home/gcweb/${IPADD}_${TOMCAT}_${DATE}.log ]; then
        cd /home/gcweb  &&  tar -zcf ${IPADD}_${TOMCAT}_${DATE}.tar.gz ${IPADD}_${TOMCAT}_${DATE}.log
        rm -f /home/gcweb/${IPADD}_${TOMCAT}_${DATE}.log
        sz /home/gcweb/${IPADD}_${TOMCAT}_${DATE}.tar.gz
        echo "Success to get logs."
        rm -f /home/gcweb/${IPADD}_${TOMCAT}_${DATE}.tar.gz
    fi
}

get_time() {
    case "$1" in
        4)

         N_DATE=`date -d "$2" +"%Y-%m-%d" 2>/dev/null`
         D_DATE=`date -d "$2" +"%Y/%m/%d" 2>/dev/null`
         E_DATE=`date -d "$2" +"%h %e,_%Y" 2>/dev/null|sed 's/ /_/g'`
         echo $N_DATE $D_DATE $E_DATE
         ;;

        7)

         TIME=`echo $2 | awk -F'_' '{print $1,$2}'`
         N_DATE=`date -d "$TIME" +"%Y-%m-%d_%H" 2>/dev/null`
         D_DATE=`date -d "$TIME" +"%Y/%m/%d_%H" 2>/dev/null`
         E_DATE=`date -d "$TIME" +"%h %e,_%Y %l" 2>/dev/null|sed 's/ /_/g'`
         echo  "$N_DATE"  "$D_DATE" "$E_DATE"
        ;;

        9)

         TIME=`echo $2 | awk -F'_' '{print $1,$2}'`
         N_DATE=`date -d "$TIME" +"%Y-%m-%d_%H:%M" 2>/dev/null`
         D_DATE=`date -d "$TIME" +"%Y/%m/%d_%H:%M" 2>/dev/null`
         E_DATE=`date -d "$TIME" +"%h %e,_%Y %l:%M" 2>/dev/null|sed 's/ /_/g'`
         echo  "$N_DATE" "$D_DATE" "$E_DATE"
        ;;

        *)
```

```
            echo 1
            ;;
        esac
}

check_arguments () {

        if [ "$1" == 1 ] || [ -z  "$1" ] ;then
            echo "你输入时间参数的格式无法识别, usage: 0108、0108_10、0108_1020"
            exit 3
        fi

}

check_tomcat () {

        if [ ! -s "${LOGFILE}" ] ;then
          echo "tomcat_name: ${TOMCAT} is not exist"
          echo "you can choose:"
          /bin/ls  /home/gcweb/usr/local/
        fi

        if [ $1 -lt 2 ] || [ ! -s "${LOGFILE}" ];then
            echo "usage: $0 tomcat_name {begin_time|begin_time end_time}"
            exit 2
        fi
}

case "$#" in

   0)
      echo "usage: $0 tomcat_name {begin_time|begin_time end_time}"
      exit 1
      ;;

   1)
      check_tomcat $#
      ;;

   2)

      check_tomcat $#
      len=`echo $2 | awk '{print length($0)}'`
      A_DATE=$(get_time  $len $BEGIN_TIME)
      eval  $( echo $A_DATE |awk '{print "N_DATE="$1,"D_DATE="$2,"E_DATE="$3}')
      check_arguments "${N_DATE}"
      cut_log "${N_DATE}" "${D_DATE}" "${E_DATE}"
      ;;

   3)

      check_tomcat $#
      len1=`echo $2 | awk '{print length($0)}'`
      len2=`echo $3 | awk '{print length($0)}'`

      A_DATE=$(get_time ${len1}  $BEGIN_TIME)
      eval  $( echo $A_DATE |awk '{print "N_DATE1="$1,"D_DATE1="$2,"E_DATE1="$3}')
      check_arguments "${N_DATE1}"
      A_DATE=$(get_time ${len2}  $END_TIME)
      eval  $( echo $A_DATE |awk '{print "N_DATE="$1,"D_DATE="$2,"E_DATE="$3}')
      check_arguments "${N_DATE}"
      cut_log ${N_DATE1} ${D_DATE1} ${E_DATE1} "${N_DATE}" "${D_DATE}" "${E_DATE}"
      ;;

   *)

      echo "usage: $0 tomcat_name {begin_time|begin_time end_time};你使用的参数太多哦."
      ;;
```

```
esac
```

【脚本93】数组

写一个脚本让用户输入多个城市的名字（可以是中文），要求不少于5个，然后把这些城市存到一个数组里，最后用for循环把它们打印出来。

参考代码：

```
#!/bin/bash

read -p "请输入至少5个城市的名字，用空格分隔:" city
n=`echo $city|awk '{print NF}'`

if [ $n -lt 5 ]
then
    echo "输入的城市个数至少为5"
    exit
fi

name=($city)

for i in ${name[@]}
do
    echo $i
done
```

【脚本94】批量同步代码

需求背景是：

一个业务，有3台服务器（A，B，C）做负载均衡，由于规模太小目前并未使用专业的自动化运维工具。有新的需求时，开发同事改完代码会把变更上传到其中一台服务器A上。但是其他2台服务器也需要做相同变更。

写一个shell脚本，把A服务器上的变更代码同步到B和C上。

其中，你需要考虑到不需要同步的目录（假如有tmp、upload、logs、caches）

参考代码：

```
#!/bin/bash

echo "该脚本将会把A机器上的/data/wwwroot/www.aaa.com目录同步到B,C机器上";
read -p "是否要继续？(y|n) "

rs() {
    rsync -azP \
    --exclude logs \
    --exclude upload \
    --exclude caches \
    --exclude tmp \
www.aaa.com/ $1:/data/wwwroot/www.aaa.com/

}

if [ $REPLY == 'y' -o $REPLY == 'Y' ]
then
    echo "即将同步……"
    sleep 2
    cd /data/wwwroot/
    rs B机器ip
    rs C机器ip
    echo "同步完成。"
elif [ $REPLY == 'n' -o $REPLY == 'N' ]
```

```
then
    exit 1
else
    echo "请输入字母y或者n"
fi
```

【脚本95】统计并发量

需求背景：

- 需要统计网站的并发量，并绘图。

思路：

1. 借助zabbix成图
2. 通过统计访问日志每秒的日志条数来判定并发量
3. zabbix获取数据间隔30s

说明： 只需要写出shell脚本即可，不用关心zabbix配置。

假设日志路径为：

```
/data/logs/www.aaa.com_access.log
```

日志格式如下：

```
112.107.15.12 - [07/Nov/2017:09:59:01 +0800] www.aaa.com "/api/live.php" 200"-" "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident
```

参考代码：

```
#!/bin/bash
log=/data/logs/www.aaa.com_access.log
t=`date -d "-1 second" +%Y:%H:%M:%S`
#可以大概分析一下每分钟日志的量级，比如说不超过3000
n=tail -3000 $log |grep -c "$t"
echo $n
```

【脚本96】关闭服务

在centos6系统里，我们可以使用ntsysv关闭不需要开机启动的服务，当然也可以使用chkconfig工具来实现。

写一个shell脚本，用chkconfig工具把不常用的服务关闭。脚本需要写成交互式的，需要我们给它提供关闭的服务名字。

参考代码：

```
#!/bin/bash

LANG=en
c="1"

while [ ! $c == "q" ]
do
    echo -e "\033[35mPlease chose a service to close from this list: \033[0m"
    chkconfig --list |awk '/3:on/ {print $1}'
    read -p "Which service to close: " s
    chkconfig $s off
    service $s stop
    read -p  "If you want's to quit this program, tab "q", or tab "Ctrl c": " c
done
```

【脚本97】重启tomcat服务

在生产环境中，经常遇到tomcat无法彻底关闭，也就是说用tomcat自带shutdown.sh脚本无法将java进程完全关掉。所以，需要借助shell脚本，将进程杀死，然后再启动。

写一个shell脚本，实现上述功能。彻底杀死一个进程的命令是 kill -9 pid.

参考代码：

```
#!/bin/bash

###功能： 重启 tomcat 进程
###要求：对于tomcat中的某些应用，使用shutdown.sh是无法完全停掉所有服务的 实际操作中都需要kill掉tomcat再重启
##
### root can not run this script.
##

if [ $USER = root ]
then
        echo "root cann't run this script!please run with other user!"
        exit 1
fi

##
### check the Parameter
##

if [[ $# -ne 1 ]]
then
        echo "Usage:$0 tomcatname"
        exit 1
fi

##
### only one process can run one time
##

TMP_FILE_U=/tmp/.tmp.ps.keyword.$USER.956327.txt
#echo $TMP_FILE_U
KEYWORD1="$0"
KEYWORD2="$1"

# 使用赋值会多fork出一个进程,所以要先重定向到一个文本,再统计.

ps ux |grep "$KEYWORD1"|grep "\<$KEYWORD2\>"|grep -v "grep" > $TMP_FILE_U
Pro_count=`cat $TMP_FILE_U |wc -l`
if [ $Pro_count -gt 1 ]
then
        echo "An other process already running ,exit now!"
        exit 1
fi

#####################################################
#                                   #
#         begin of the script         #
#                                   #
#####################################################

##
### set the Parameter
##

TOM=`echo $1|sed 's#/##g'`
TOMCAT_DIRECTORY=~/usr/local/$TOM
STARTUP_SCRIPT=$TOMCAT_DIRECTORY/bin/startup.sh
TOMCAT_LOG=$TOMCAT_DIRECTORY/logs/catalina.out
CONF_FILE=$TOMCAT_DIRECTORY/conf/server.xml
TEMPFILE=/tmp/.tmpfile.x.89342.c4r3.tmp
```

```
##
### check if the tomcat directory exist
##

if [ ! -d "$TOMCAT_DIRECTORY" ]
then
        echo  "the tomcat \"$TOM\" not exist.check again!"
        exit 1
fi

##
### log roteta and delete log one week ago
##

rotate_log(){
TIME_FORMART=$(date +%Y%m%d%H%M%S)
LOG_DIR=$(dirname $TOMCAT_LOG)
mv $TOMCAT_LOG ${TOMCAT_LOG}_${TIME_FORMART}
find $LOG_DIR -type f -ctime +7 -exec rm -rf {} \;
}

##
### function start the tomcat
##

start_tomcat()
{
#echo start-tomcat-func
if [ -x  "$STARTUP_SCRIPT" ]
then

        rotate_log
        $STARTUP_SCRIPT
        sleep 1
        tail -f $TOMCAT_LOG
else
        if [ -e $STARTUP_SCRIPT ]
        then
                chmod +x $STARTUP_SCRIPT
#        echo "permition added!"
                if [ -x  "$STARTUP_SCRIPT" ]
                then

                        rotate_log
                        $STARTUP_SCRIPT
                        sleep 1
                        tail -f $TOMCAT_LOG
                else
                        echo  "The script not have excute permision,Couldn't add permision to Script!"
                        exit 1
                fi
        else
                echo  "error,the script \"startup.sh\" not exist!"
                exit 1
        fi
fi
}

##
### function stop the tomcat
##

stop_tomcat()
{
rm -rf $TEMPFILE
ps ux |grep /$TOM/ |grep -v "grep /$TOM/"|grep java > $TEMPFILE
Pro_Count=`cat $TEMPFILE|wc -l`
PIDS=`cat $TEMPFILE|awk '{print $2}'`
rm -rf $TEMPFILE
#echo $Pro_Count
```

```
if [ $Pro_Count -eq 0 ]
then
        echo "The tomcat not running now!"
else
        if [ $Pro_Count -ne 1 ]
        then
                echo "The have $Pro_Count process running,killed!"
                kill -9 `echo $PIDS`
                WC=`ps aux | grep "/$TOM/" | grep -v "grep /$TOM/" | grep java |wc -l`
                [ $WC -ne 0 ] && (echo "kill process failed!";exit 1)
        else
                echo "Process killed!"
                kill -9 `echo $PIDS`
                WC=`ps aux | grep "/$TOM/" | grep -v "grep /$TOM/" | grep java |wc -l`
                [ $WC -ne 0 ] && (echo "kill process failed!";exit 1)
        fi
fi
}

############################
####            ####
#### The main script ####
####            ####
############################

echo -e "are you sure restart $TOM?(y or n)"
read ANS
if [ "$ANS"a != ya ]
then
    echo -e "bye! \n"
    exit 1
fi

stop_tomcat
echo "start tomcat ..."
sleep 2
start_tomcat
# end
```

【脚本98】取消后缀

至少用两种方法，批量把当前目录下面所有文件名后缀为.bak的后缀去掉，比如1.txt.bak去掉后为1.txt

假设取消的后缀为.bak

方法一：

```
#!/bin/bash

for i in `ls *.bak`
do
    mv $i `echo $i|sed 's/\.bak//g'`
done
```

方法二：

```
#!/bin/bash

for i in `ls *.bak`
do
    newname=`echo $i|awk -F '.bak' '{print $1}'`
    mv $i $newname
done
```

【脚本99】域名到期提醒

写一个shell脚本，查询指定域名的过期时间，并在到期前一周，每天发一封提醒邮件。

思路： 大家可以在linux下使用命令“whois 域名”，如”whois xxx.com”，来获取该域名的一些信息。

提示： whois命令，需要安装jwhois包

参考代码：

```
#!/bin/bash

t1=`date +%s`
is_install_whois()
{
    which whois >/dev/null 2>/dev/null
    if [ $? -ne 0 ]
    then
        yum install -y jwhois
    fi
}

notify()
{
    e_d=`whois $1|grep 'Expiry Date'|awk '{print $4}'|cut -d 'T' -f 1`
    e_t=`date -d "$e_d" +%s`
    n=`echo "86400*7"|bc`
    e_t1=$[$e_t-$n]
    if [ $t1 -ge $e_t1 ] && [ $t1 -lt $e_t ]
    then
        /usr/local/sbin/mail.py aming_test@163.com "Domain $1 will be expire." "Domain $1 expire date is $e_d."
    fi
}

is_install_whois
notify xxx.com
```

【脚本100】自动增加公钥

写一个shell脚本，当我们执行时，提示要输入对方的ip和root密码，然后可以自动把本机的公钥增加到对方机器上，从而实现密钥认证。

参考代码：

```
#!/bin/bash

read -p "Input IP: " ip
ping $ip -w 2 -c 2 >> /dev/null

## 查看ip是否可用
while [ $? -ne 0 ]
do
    read -p "your ip may not useable, Please Input your IP: " ip
    ping $ip -w 2 -c 2 >> /dev/null
done
read -p "Input root\'s password of this host: " password

## 检查命令子函数

check_ok() {
if [ $? != 0 ]
then
    echo "Error!."
    exit 1
fi
}

## yum需要用到的包
```

```
myyum() {
if ! rpm -qa |grep -q "$1"
then
    yum install -y $1
    check_ok
else
    echo $1  already installed
fi
}

for p in openssh-clients openssh expect
do
    myyum $p
done

## 在主机A上创建密钥对

if [ ! -f ~/.ssh/id_rsa ] || [ ! -f ~/.ssh/id_rsa.pub ]
then
    if [ -d ~/.ssh ]
    then
        mv ~/.ssh/  ~/.ssh_old
    fi
    echo -e "\n" | ssh-keygen -t rsa -P ''
    check_ok
fi

## 传私钥给主机B

if [ ! -d /usr/local/sbin/rsync_keys ]
then
    mkdir /usr/local/sbin/rsync_keys
fi
cd /usr/local/sbin/rsync_keys
if [ -f rsync.expect ]
then
    d=`date +%F-%T`
    mv rsync.expect $d.expect
fi

#创建远程同步的expect文件

cat >  rsync.expect <<EOF
#!/usr/bin/expect
set host [lindex \$argv 0]
#主机B的密码
set passwd [lindex \$argv 1]
spawn rsync -av /root/.ssh/id_rsa.pub root@\$host:/tmp/tmp.txt
expect {
"yes/no" { send "yes\r"; exp_continue}
"password:" { send "\$passwd\r" }
}
expect eof
spawn ssh root@\$host
expect {
"password:" { send "\$passwd\r" }
}
expect "]*"
send "\[ -f /root/.ssh/authorized_keys \] && cat /tmp/tmp.txt >>/root/.ssh/authorized_keys \r"
expect "]*"
send "\[ -f /root/.ssh/authorized_keys \] || mkdir -p /root/.ssh/ \r"
send "\[ -f /root/.ssh/authorized_keys \] || mv /tmp/tmp.txt /root/.ssh/authorized_keys\r"
expect "]*"
send "chmod 700 /root/.ssh; chmod 600 /root/.ssh/authorized_keys\r"
expect "]*"
send "exit\r"
EOF

check_ok
```

```
/usr/bin/expect /usr/local/sbin/rsync_keys/rsync.expect $ip $password
echo "OK,this script is successful. ssh $ip to test it"
```

【脚本101】自动封/解封ip

需求背景：

discuz论坛，每天有很多注册机注册的用户，然后发垃圾广告帖子。虽然使用了一些插件但没有效果。分析访问日志，发现有几个ip访问量特别大，所以想到可以写个shell脚本，通过分析访问日志，把访问量大的ip直接封掉。

但是这个脚本很有可能误伤，所以还需要考虑到自动解封这些ip。

思路：

1. 可以每分钟分析1次访问日志，设定一个阈值，把访问量大的ip用iptables封掉80端口
2. 每20分钟检测一次已经被封ip的请求数据包数量，设定阈值，把没有请求的或者请求量很小的解封

参考代码：

```
#! /bin/bash

## To block the ip of bad requesting.
## Writen by aming 2017-11-18.

log="/data/logs/www.xxx.com.log"
tmpdir="/tmp/badip"
#白名单ip，不应该被封
goodip="27.133.28.101"

[ -d $tmpdir ] || mkdir -p $tmpdir

t=`date -d "-1 min"  +%Y:%H:%M`

#截取一分钟以前的日志
grep "$t:" $log > $tmpdir/last_min.log

#把一分钟内日志条数大于120的标记为不正常的请求
awk '{print $1}' $tmpdir/last_min.log |sort -n |uniq -c |sort -n |tail |awk '$1>120 {print $2}'|grep -v "$good_ip"> $tmpdir/bad.ip

d3=`date +%M`

#每隔20分钟解封一次ip
if [ $d3 -eq "20" ] || [ $d3 -eq "40" ] || [ $d3 -eq "00" ]
then
        /sbin/iptables -nvL INPUT|grep 'DROP' |awk '$1<10 {print $8}'>$tmpdir/good.ip
        if [ -s $tmpdir/good.ip ]
        then
        for ip in `cat $tmpdir/good.ip`
        do
                /sbin/iptables -D INPUT -p tcp --dport 80 -s $ip -j DROP
                d4=`date +%Y%m%d-%H:%M`
                echo "$d4 $ip unblock" >>$tmpdir/unblock.ip
        done
        fi

        #解封后，再把iptables的计数器清零
        /sbin/iptables -Z INPUT
fi

if [ -s $tmpdir/bad.ip ]
then
    for ip in `cat $tmpdir/bad.ip`
    do
        /sbin/iptables -A INPUT -p tcp --dport 80 -s $ip -j DROP
        d4=`date +%Y%m%d-%H:%M`
        echo "$d4 $ip block" >>$tmpdir/block.ip
```

```
    done
fi
```

【脚本102】单机部署SpringBoot项目

有一台测试服务器，经常需要部署SpringBoot项目，手动部署太麻烦，于是写了个部署脚本

脚本代码：

```
#!/bin/bash

# git仓库路径
GIT_REPOSITORY_HOME=/app/developer/git-repository
# jar包发布路径
PROD_HOME=/prod/java-back
# 应用列表
APPS=(app1 app2 app3)

if [ ! -n "$1" ]
then
    echo -e "请输入要发布的项目！"
    exit
fi

# cd dir
for((i=0;i<${#APPS[@]};i++))
do
    echo $1 ${APPS[i]}
    if [ $1 = ${APPS[i]} ]
    then
        echo -e ===========Enter $1=============
        cd ${GIT_REPOSITORY_HOME}/$1
        break
    fi
done

if [ `pwd` != ${GIT_REPOSITORY_HOME}/$1 ]
then
     echo -e "输入的项目名没有找到！"
     exit
fi

echo "==========git切换分之到master==============="
git checkout master

echo "==================git fetch======================"
git fetch

echo "==================git pull======================"
git pull

echo "===========选择线上环境编译并跳过单元测试==================="
mvn clean package -Dmaven.test.skip=true -Pprod

jar_path=${GIT_REPOSITORY_HOME}/$1/target/*-0.0.1-SNAPSHOT.jar
echo ${jar_path}
if [ -f ${jar_path} ]
then
     # backup dest
     echo -e "===========jar backup============="
     mv ${PROD_HOME}/$1/*-0.0.1-SNAPSHOT.jar ${PROD_HOME}/$1/$1-0.0.1-SNAPSHOT.jar.back

     # copy
     echo "======拷贝编译出来的jar包拷贝到$PROD_HOME======="
     if [ -d ${PROD_HOME}/$1 ]
     then
```

```
        /bin/cp ${GIT_REPOSITORY_HOME}/$1/target/*-0.0.1-SNAPSHOT.jar  ${PROD_HOME}/$1
    else
        mkdir ${PROD_HOME}/$1
        /bin/cp ${GIT_REPOSITORY_HOME}/$1/target/*-0.0.1-SNAPSHOT.jar  ${PROD_HOME}/$1
    fi

    echo "============停止项目============="
    jar_name=`jps |awk -F" " '{ print $2 }'| egrep ^$1.*jar$`
    pid=`jps |grep ${jar_name} | awk -F" " '{ print $1 }'`
    echo ${pid}
    kill -15 ${pid}

    echo "================sleep 10s========================="
    for i in 1 2 3 4 5 6 7 8 9 10
    do
        echo ${i}"s"
        sleep 1s
    done

    echo "============启动项目============="
    nohup java -jar ${PROD_HOME}/$1/*-0.0.1-SNAPSHOT.jar > /dev/null 2>&1 &
    echo -e "============Deploy Success============="
else
    echo -e "============Deploy Error============="
fi
```


shell 练习题 脚本 CentOS

6 收藏 分享