一.shell 应用

1.1 批量添加用户并且添加密码

```
[root@shell shell]# cat 1.sh
#!/bin/bash
echo "-------add user--------"
for i in {1..9}
do
        useradd user$i
        passwd=old$i
        echo "$passwd"|passwd --stdin user$i
done
```

1.2 编写 tomcat 的启动脚本

```
[root@long01 init.d]# vi tomcat
#!/bin/bash
tomcat_home=/usr/local/tomcat7
case $1 in
        start)
                sh $tomcat_home/bin/startup.sh
                ;;
        stop)
                sh $tomcat_home/bin/shutdown.sh
                ;;
        restart)
                sh $tomcat_home/bin/shutdown.sh
                sh $tomcat_home/bin/startup.sh
                ;;
        *)
                echo 'please use : tomcat{start |stop|restart}'
                ;;
esac
exit 0
[root@long01 init.d]# ./tomcat start
```

1.3 查找一个文件是否存在

```
[root@shell 7-11]# cat 3.sh
#!/bin/bash
if test -f /etc/hosts
        then
                echo 1
else
                echo 0
fi
[root@shell 7-11]# cat 2.sh
#!/bin/bash
if [ -f /etc/hosts ];then
        echo 1
else
        echo 0
fi
```

1.4 判断两个变量的大小

```
[root@shell 7-11]# cat 6.sh
#!/bin/bash
read -p "please input two num:" a b
if [ $a -gt $b ];then
        echo "yes,$a greater $b"
elif [ $a -lt $b ];then
        echo "yes,$a lesster $b"
else
        echo "yes, $a equal $b"
fi
[root@shell 7-11]# source 6.sh
please input two num:1 2
yes,1 lesster 2
[root@shell 7-11]# source 6.sh
please input two num:3 2
yes,3 greater 2
[root@shell 7-11]# source 6.sh
please input two num:1 1
yes, 1 equal 1
```

1.5 判断 mysql 服务是否正常

```
[root@shell 7-11]# cat 7.sh
#!/bin/bash
if [ `netstat -ntlp|grep mysqld|wc -l` -gt 0 ];then
        echo "mysqld is running"
else
        echo "mysqld is stopped"
        /etc/init.d/mysqld start
Fi
[root@shell 7-11]# source 7.sh
mysqld is running
```

1.6 计算 1 到 100 的和

```
[root@shell 7-18]# sh 4.sh
totalsum is :5050
[root@shell 7-18]# cat 4.sh
#!/bin/bash
sum=0
for i in {1..100}
do
        sum=$[$sum+$i]
done
        echo "totalsum is :$sum"
```

1.7 判断变量值或字符串是否为整数

```
[root@shell 7-21]# i=5
[root@shell 7-21]# expr $i + 6 &>/dev/null
[root@shell 7-21]# echo $?
0
[root@shell 7-21]# i=oldboy
[root@shell 7-21]# expr $i + 6 &>/dev/null
[root@shell 7-21]# echo $?
2
```

1.8 批量更改文件的扩展名

```
[root@shell 7-27]# ls
11.sh      21.gif    24.gif    27.gif    2.sh       4.sh           test.gif
1.sh       22.gif    25.gif    28.gif    30.gif     5.sh           tetre.gif
20.gif     23.gif    26.gif    29.gif    3.sh       gogo.gif
[root@shell 7-27]# cat 11.sh
#!/bin/bash
cd /shell/7-27
for filename in `ls|grep "txt$"`
do
        mv $filename `echo $filename|cut -d . -f1`.gif
done

[oracle@oracle oracledata]$ ls
datacenter20201025.dmp                  datacenter20201025.log                  qzj_tar_20201025.dmp
qzj_tar_20201025.log    qzj_tar_datacenter20201025.dmp    qzj_tar_datacenter20201025.log
[oracle@oracle oracledata]$ pwd
/data1/backup/oracledata
[oracle@oracle script]$ cat 1.sh
#!/bin/bash
datatime=`date +%Y%m%d`
datetime_three_ago=`date -d "3 day ago" +%Y%m%d`
source /etc/profile
source ~/.bash_profile
cd /data1/backup/oracledata
for file in `ls|grep .$`
do
  newfile=`echo $file|sed 's/'${datatime}'/'${datetime_three_ago}'/g'`
  mv $file $newfile
done
```

1.9 乘法表

```
[root@shell 7-27]# cat 12.sh
#!/bin/bash
for num1 in `seq 9`
do
        for num2 in `seq 9`
        do
        if [ $num1 -ge $num2    ];then
                if (((num1*num2)>9));then
```

```
            echo -en "${COLOR}${num1}x${num2}=$((num1*num2))$RES "
        else
            echo -en "${COLOR}${num1}x${num2}=$((num1*num2))$RES    "
        fi
    fi
        done
echo ""
done
```

1.10 数据库备份

```
#!/bin/bash
myuser=root
mypass=00000000
socket=/var/lib/mysql/mysql.sock
mycmd="mysql -u$myuser -p$mypass -S $socket"
for dbname in boy girl
do
    $mycmd -e "create database $dbname"
done
```

```
#!/bin/bash
dbpath=/root/backup
myuser=root
mypass=00000000
socket=/var/lib/mysql/mysql.sock
mycmd="mysql -u$myuser -p$mypass -S $socket"
mydump="mysqldump -u$myuser -p$mypass -S $socket"
[ ! -d "$dbpath"    ] && mkdir $dbpath
for dbname in `$mycmd -e "show databases;"|sed '1,2d'|egrep -v "mysql|schema"`
do
    $mydump $dbname|gzip>$dbpath/${dbname}_$(date +%F).sql.gz
Done
```

```
#!/bin/bash
myuser=root
mypass=00000000
socket=/var/lib/mysql/mysql.sock
mycmd="mysql -u$myuser -p$mypass -S $socket"
for dbname in boy girl
do
    $mycmd -e "use $dbname;create table test2(id int,name varchar(16));insert into test2
```

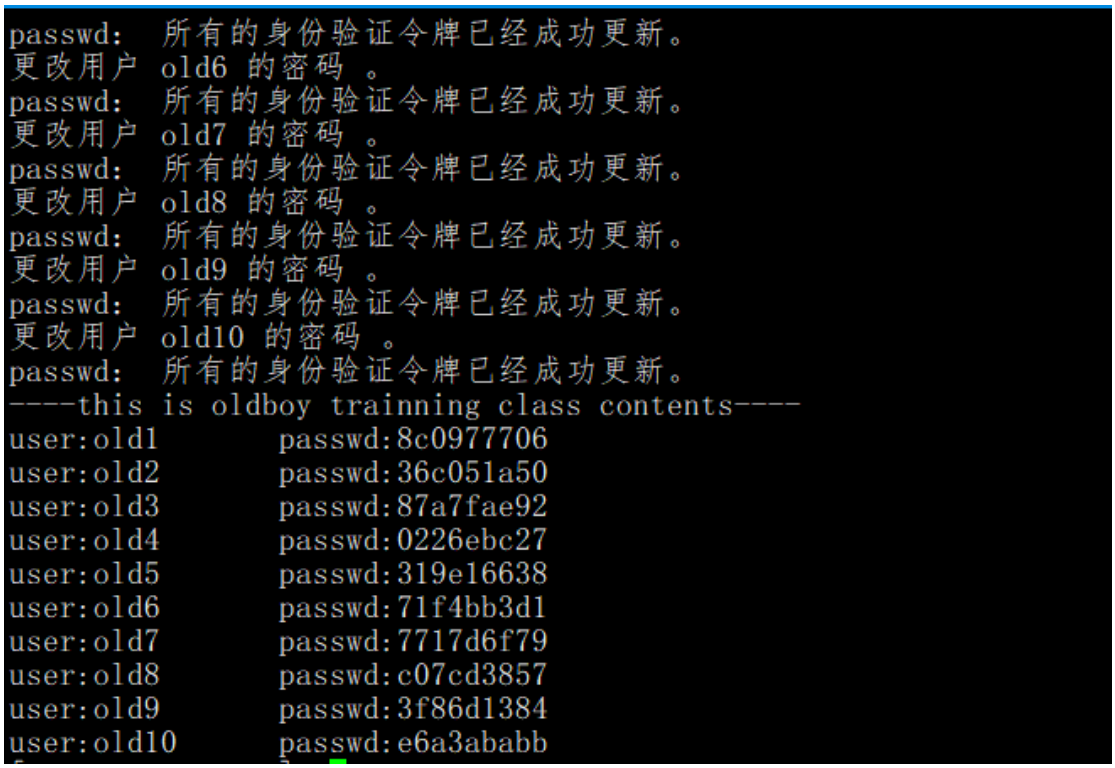
```
values(1,'testdate');"
done
```

```
#!/bin/bash
myuser=root
mypass=00000000
socket=/var/lib/mysql/mysql.sock
mycmd="mysql -u$myuser -p$mypass -S $socket"
for dbname in boy girl
do
    echo "========$dbname.test2========"
    $mycmd -e "use $dbname;select * from ${dbname}.test2;"
done
```

1.11 批量添加随机密码的用户

```
[root@shell 7-28]# cat 5.sh
#!/bin/bash
user="old"
passfile="/tmp/user.log"
for num in {1..10}
do
    useradd $user$num

        pass="`echo "test$RANDOM"|md5sum|cut -c3-11`"
    echo "$pass"|passwd --stdin $user$num
    echo -e "user:$user$num\tpasswd:$pass">>$passfile
done
echo ----this is oldboy trainning class contents----
cat $passfile
```



1.12 判断字符串的个数

```
[root@shell 7-30]# sh 2.sh
i
am
oldboy
to
oldboy
class
[root@shell 7-30]# cat 2.sh
#!/bin/bash
a=(i am oldboy teacher welcome to oldboy training class)
for n in ${a[*]}
do
    if [ `echo $n|wc -L` -le 6 ];then
        echo $n
    fi
done
```

1.13 扫描网段内存活的主机

```
[jk@lb01 server_scripts]$ cat check_ip.sh
#!/bin/bash
# Author: hhl
# Mail: 267@qq.com
# Function: This script is used to scan the IP survivability in this network segment
# Version: 1.1

date=`date +%Y-%m-%d\ %H:%M`
mail_user=(13242@139.com)

cmd="ping -c 1"
datecenter_server_ip="192.168.1"

for n in {213..214}
do
    for count in {1..3}
    do

    # ping 三次，只要其中有一次通，则跳出循环，再通过 ok 的值判断对应状态

        ok=0
        $cmd $datecenter_server_ip.$n &>/dev/null
        if [ $? -eq 0 ];then
                ok=1
            break
        fi
        done
    if [ $ok -eq 0 ];then
        echo -e "告警时间: ${date}\n 告警服务器 ip: ${datecenter_server_ip}.$n \n 服务器状
态:  宕机或网络不通" |mail -s "数据中心服务器存活状态告警" ${mail_user[*]}
    fi
done
```

#脚本说明:通过 ping 命令监控主机是否存活，如果 ping 失败则继续 ping，三次不通则认为宕机或者网络不通，这时可发邮件告警

#https://blog.csdn.net/qq_40907977/article/details/103277646

1.14 mysql 主从复制异常监控

```
[root@mysql02 ~]# cat 2.sh
#!/bin/bash
MYSQL_PD="mmds"
MYSQLCMD="mysql -uroot -p${MYSQL_PD}"
Slave_IO_Running=`${MYSQLCMD} -e "show slave status\G;" 2>/dev/null|grep 'Slave_IO_Running'|awk '{print $2}'`
Slave_SQL_Running=`${MYSQLCMD} -e "show slave status\G;" 2>/dev/null|grep 'Slave_SQL_Running'|awk '{print $2}'`
Seconds_Behind_Master=`${MYSQLCMD} -e "show slave status\G;" 2>/dev/null|grep 'Seconds_Behind_Master'|awk '{print $2}'`
if [[ "${Slave_IO_Running}" = "Yes" || "${Slave_SQL_Running}" = "Yes" || ${Seconds_Behind_Master} = 0 ]];then
        echo "mysql_salve_status succeed!"
else
        echo "mysql_salve_status failed!"
fi
[root@mysql02 ~]# sh 2.sh
mysql_salve_status succeed!
```

1.15 往 oracle 插入数据

```
[oracle@oracle script]$ cat mem_total.sh
```

```
#!/bin/bash
source /etc/profile
source ~/.bash_profile
total=`free -m|awk    'NR==2 {print $2}'`
used=`free -m|awk    'NR==2 {print $3}'`
free=`free -m|awk    'NR==2 {print $4}'`
shared=`free -m|awk    'NR==2 {print $5}'`
buffers=`free -m|awk    'NR==2 {print $6}'`
cached=`free -m|awk    'NR==2 {print $7}'`
up_date=`date +%Y/%m/%d`
id="`echo "test$RANDOM"|md5sum|cut -c3-11`"
sqlplus -s qzj/qzj@datacenter << EOF
insert                  into                  mem_total(total,used,free,shared,buffers,cached,up_date,id)
values(${total},${used},${free},${shared},${buffers},${cached},sysdate,'${id}');
commit;
exit;
EOF
```

1.16 oracle 表空间使用率告警

```
[oracle@oracle script]$ cat oracle_space.sh
#!/bin/bash
MAIL_USER=(1324@139.com)
source /etc/profile
source ~/.bash_profile
space=`sqlplus -s devwang/mmds@datacenter << EOF
set feedback off
set heading off
SELECT
round((total - free) / total, 2) * 100 "use_present"
   FROM (SELECT tablespace_name, SUM(bytes) free
                FROM dba_free_space
               GROUP BY tablespace_name) a,
            (SELECT tablespace_name, SUM(bytes) total
                FROM dba_data_files
               GROUP BY tablespace_name) b
  WHERE a.tablespace_name = b.tablespace_name and b.tablespace_name='SJZX_DATA';
exit;
EOF`
if [ $space -gt 10 ];then
       echo "当前表空间使用率为${space}%">/tmp/space.txt
       mail -s "$HOSTNAME  表空间告警"    ${MAIL_USER[*]}< /tmp/space.txt
fi
```

1.17 oracle 数据库备份

```
#创建备份目录，赋予用户权限
[root@oracle ~]# mkdir /data1/backup/oracledata -p
[root@oracle ~]# chown -R oracle:oinstall /data1/
[oracle@oracle ~]$ sqlplus sys/oracle as sysdba
SQL> create or replace directory backup_path as '/data1/backup/oracledata';
SQL> grant read,write on directory backup_path to devwang;
[oracle@oracle script]$ cat oracle_backup.sh
#!/bin/bash
datatime=`date +%Y%m%d`
datetime_three_ago=`date -d "3 day ago" +%Y%m%d`
backup_dir=/data1/backup/oracledata
source /etc/profile
source ~/.bash_profile
expdp devwang/mmds directory=backup_path dumpfile=datacenter${datatime}.dmp logfile=datacenter${datatime}.log FULL=y;
expdp devwang/mmds directory=backup_path dumpfile=qzj_tar_datacenter${datatime}.dmp logfile=qzj_tar_datacenter${datatime}.log schemas=qzj_tar;
expdp qzj_tar/qzj directory=backup_path dumpfile=qzj_tar_${datatime}.dmp logfile=qzj_tar_${datatime}.log tables=MEM_TOTAL,T%;
cd ${backup_dir}
rm -rf *${datetime_three_ago}*
```

1.18 输出 500 内是 7 的倍数的数

```
[root@python ~]# cat 1.sh
#!/bin/bash
for i in {0..500}
do
    if [ $(($i % 7 )) -eq 0 ];then
        echo $i
    fi
done
```

1.19 统计文本中每行中包含 1-5 数字的个数和总数

写一个 bash 脚本以统计一个文本文件 nowcoder.txt 中每一行出现的 1,2,3,4,5 数字个数并且要计算一下整个文档中一共出现了几个 1,2,3,4,5 数字数字总数。

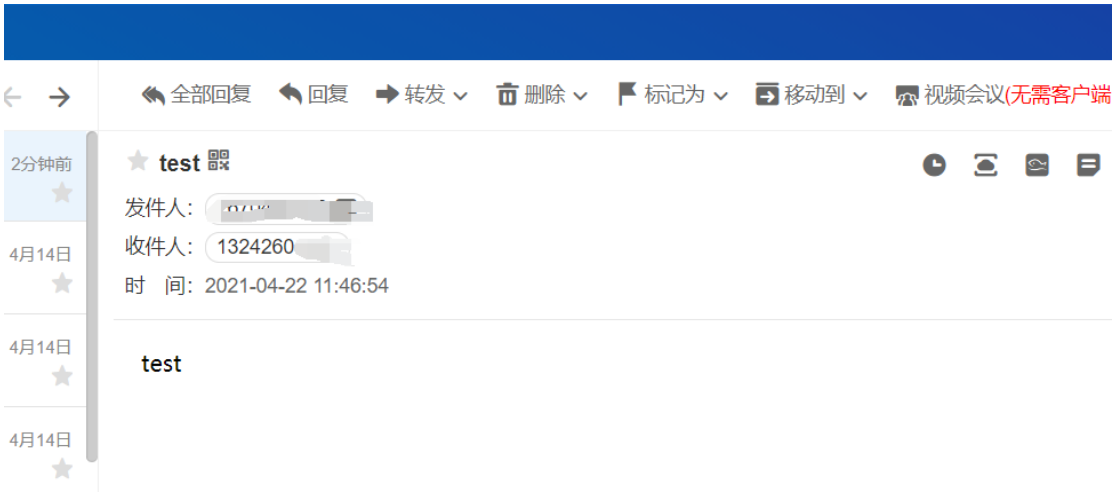
示例:
假设 nowcoder.txt 内容如下：
a12b8
10ccc
2521abc
9asf
你的脚本应当输出：
line1 number: 2
line2 number: 1
line3 number: 4
line4 number: 0
sum is 7
[root@python ~]# sh 4.sh
line1 number: 2
line2 number: 1
line3 number: 4
line4 number: 0
sum is 7
[root@python ~]# vi 4.sh
[root@python ~]# cat 4.sh
#!/bin/bash
a=0
sum=0
for i in `cat /root/nowcoder.txt`
do
    n=`echo "$i" |grep -o -E '[1-5]'|wc -l`
    a=$((a + 1))
    echo "line${a} number: ${n}"
    sum=$(($sum + $n))
done
echo "sum is ${sum}"
#-o 只显示出条件里的，其他不显示

1.20 磁盘监控

#邮件配置
yum install -y mailx
vi /etc/mail.rc
set from=26704864@qq.com
set smtp=smtp.qq.com
set smtp-auth-user=26704864@qq.com
set smtp-auth-password=gxjvjnsnbjmu #授权码

set smtp-auth=login

echo test | mail -s "test" 1324@139.com #测试邮件发送是否正常



#添加监控用户

useradd jk

echo "jk" |--stdin passwd jk

su – jk

#新建监控脚本存放目录

mkdir /home/jk/server_scripts/

cat disk.sh

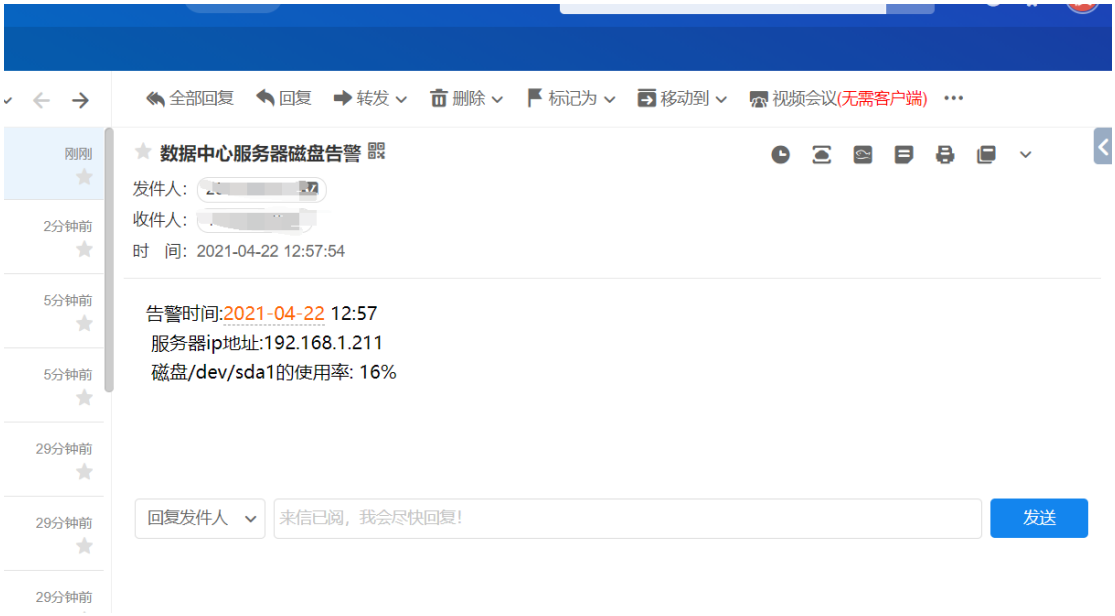
```
#!/bin/bash
#Date: 2021-04-22
#Author: hhl
#Mail: 267@qq.com
#Function: This purpose of this scripts is to monitor disk space
#Version: 1.1

date=`date +%Y-%m-%d\ %H:%M`
mail_user=(132426@139.com)
datacenter_server_ip=`ifconfig |grep inet|awk 'NR==3 {print $2}'`
warn_dev_name=`df -hT|grep '^/dev/*'|awk '{print $1}'`
dev_rate=`df -hT|grep '^/dev/*'|awk '{print $6}'|awk -F'%' '{print $1}'`
for i in $dev_rate
do
if [ $i -ge 80 ];then
        for g in $warn_dev_name
        do
            count=`df -hT|grep "${g}"|awk '{print $6}'|grep "${i}"|wc -l`
            if [ $count -ge 1 ];then
                echo -e "告警时间:${date}\n 服务器 ip 地址:${datacenter_server_ip}\n 磁盘${g}的使用率: ${i}%" | mail -s "数据中心服务器磁盘告警" ${mail_user[*]}
```

```
            fi
        done
fi
done

chmod +x disk.sh
sh disk.sh
```

#查看邮件



#加入定时任务

```
crontab -l
* */1 * * * sh /home/jk/scripts/disk.sh
```

1.21 内存监控

```
[jk@lb01 server_scripts]$ cat free.sh
#!/bin/bash
#Date: 2021-04-22
#Author: hhl
#Mail: 267@qq.com
#Function: the script monitors the server memory
#Version: 1.1

date=`date +%Y-%m-%d\ %H:%M`
mail_user=(132426@139.com)
datacenter_server_ip=`ifconfig |grep inet|awk 'NR==3 {print $2}'`
total=`free -m|sed -n '2p'|awk '{print $2}'`
```

```
used=`free -m|sed -n '2p'|awk '{print $3}'`
rate=`echo "scale=2;$used/$total" |bc|awk -F'.' '{print $2}'`
#scale=2 设小数位，2 代表保留两位
top10_processes_mem=`ps aux | grep -v USER | sort -rn -k4 | head`
if [ $rate -ge 27 ];then
        echo -e "告警时间: ${date}\n 告警服务器 ip: ${datacenter_server_ip}\n 内存使用率: ${rate}%\n 占用内存前十的进程:\n ${top10_processes_mem}" |mail -s "数据中心服务器内存告警" ${mail_user[*]}
fi
## echo -e 开启转义
```

1.22 cpu 监控

```
[jk@lb01 server_scripts]$ cat cpu.sh
#!/bin/bash
#Author: hhl
#Mail: 267@qq.com
#Function: the purpose of this script is to monitor the server CPU
#Version: 1.1

date=`date +%Y-%m-%d\ %H:%M`
mail_user=(13242@139.com)
datacenter_server_ip=`ifconfig |grep inet|awk 'NR==3 {print $2}'`
cpu_count=`cat /proc/cpuinfo | grep "processor" | wc -l`
cpu_load_average=`uptime |awk '{print $NF}'|cut -d'.' -f 1`
cpu_use=`top -b -n  1|grep "Cpu"|awk '{print $2}'|cut -d '.' -f 1`
cpu_use_top10=`ps aux|grep -v USER|sort -rn -k3|head`
if [ $cpu_use -gt 80 ];then
      echo -e "告警时间:${date}\n 服务器 ip 地址:${datacenter_server_ip}\n cpu 使用率: ${cpu_use}%\n 占用 cpu 最高的十个进程:\n ${cpu_use_top10}" |mail -s "数据中心服务器 cpu 告警" ${mail_user}
fi
if [ $cpu_load_average -gt $cpu_count ];then
        echo -e "告警时间:${date}\n 服务器 ip 地址:${datacenter_server_ip}\ncpu 逻辑个数: ${cpu_count}\ncpu 负载: ${cpu_load_average}\n 占用 cpu 最高的十个进程:\n ${cpu_use_top10}" |mail -s "数据中心服务器 cpu 告警" ${mail_user[*]}
fi
```

1.23 磁盘分区

```
cat make_partition.sh
#!/bin/bash
```

```
#Author: hhl
#Mail: 267@qq.com
#Function: this scripts is use to make partition
#Version: 1.1

fdisk /dev/sdb <<EOF
n
p
2

+2G
p
w
EOF
fdisk -l |grep '/dev/sdb'
```

1.24 数组实战

描述
写一个 bash 脚本以实现一个需求，求输入的一个的数组的平均值

第 1 行为输入的数组长度 N
第 2~N 行为数组的元素，如以下为:
数组长度为 4，数组元素为 1 2 9 8
示例:
4
1
2
9
8

那么平均值为:5.000(保留小数点后面 3 位)
你的脚本获取以上输入应当输出：
5.000

代码如下:

```
#!/bin/bash
read -t 30    -p "请输入数组的长度和数组的元素":
array=(${REPLY})
sum=0
for i in    ${array[*]}
do
        sum=$(($sum+$i))
```

```
done
array_length=`echo ${array[0]} + 0|bc`
sum=`echo $sum - $array_length|bc`
echo "scale=3;${sum}/${array_length}"|bc

[root@docker02 scripts]# sh avg.sh
请输入数组的长度和数组的元素:3 8 2 4
4.666
You have new mail in /var/spool/mail/root
[root@docker02 scripts]# sh avg.sh
请输入数组的长度和数组的元素:3 9 2 8
6.333
```

1.25 nginx 日志分析

描述
假设 nginx 的日志我们存储在 nowcoder.txt 里，格式如下：

```
192.168.1.20 - - [21/Apr/2020:14:27:49 +0800] "GET /1/index.php HTTP/1.1" 404 490 "-" "Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:45.0) Gecko/20100101 Firefox/45.0"
192.168.1.21 - - [21/Apr/2020:15:27:49 +0800] "GET /2/index.php HTTP/1.1" 404 490 "-" "Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:45.0) Gecko/20100101 Firefox/45.0"
192.168.1.22 - - [21/Apr/2020:21:27:49 +0800] "GET /3/index.php HTTP/1.1" 404 490 "-" "Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:45.0) Gecko/20100101 Firefox/45.0"
192.168.1.23 - - [21/Apr/2020:22:27:49 +0800] "GET /1/index.php HTTP/1.1" 404 490 "-" "Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:45.0) Gecko/20100101 Firefox/45.0"
```

1.25.1 ip 统计

现在需要你统计出 2020 年 4 月 23 号的访问 ip 次数，并且按照次数降序排序。你的脚本应该输出：

```
5 192.168.1.22
4 192.168.1.21
3 192.168.1.20
2 192.168.1.25
1 192.168.1.24
```

代码如下：

```
grep -e '23/Apr/2020' nowcoder.txt |awk '{print $1}'|sort |uniq -c|sort -r|awk '{print $1 " " $2}'
```

1.25.2 统计某个时间段的 IP

现在你需要统计 2020 年 04 月 23 日 20-23 点的去重 IP 访问量，你的脚本应该输出

代码如下:
grep -E "23/Apr/2020:[20..22]|23/Apr/2020:23:00" nowcoder.txt|awk '{print $1}'|sort|uniq -c|wc -l

1.25.3 统计访问 3 次以上的 IP

代码如下:
cat nowcoder.txt |awk '{print $1}'|sort|uniq -c|sort -r|awk '$1>3 {print $1 " "$2} '

1.25.4 查询某个 IP 的详细访问情况

现在需要你查询 192.168.1.22 的详细访问情况，按访问频率降序排序。
代码如下:
grep -e '192.168.1.22' nowcoder.txt |awk '{print $1 " "$7}'|sort|uniq -c|sort -r|awk '{print $1 " "$3}'

4 /1/index.php
2 /3/index.php

1.25.5 统计爬虫抓取 404 的次数

现在需要你统计百度爬虫抓取 404 的次数,代码如下:
grep -E "http://www.baidu.com/search/spider.html" nowcoder.txt|grep '404'|wc -l

1.25.6 统计每分钟的请求数

现在需要你统计每分钟的请求数，并且按照请求数降序排序 代码如下：
awk -F':' '{print $2 ":" $3}' nowcoder.txt |sort|uniq -c|sort -r|awk '{print $1 " " $2}'

1.25.7 查看各个状态的连接数

描述

假设 netstat 命令运行的结果我们存储在 nowcoder.txt 里，格式如下：

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	0	0.0.0.0:6160	0.0.0.0:*	LISTEN
tcp	0	0	127.0.0.53:53	0.0.0.0:*	LISTEN
tcp	0	0	0.0.0.0:22	0.0.0.0:*	LISTEN
tcp	0	0	172.16.56.200:41856	172.16.34.144:3306	ESTABLISHED
tcp	0	0	172.16.56.200:49822	172.16.0.24:3306	ESTABLISHED
tcp	0	0	172.16.56.200:49674	172.16.0.24:3306	ESTABLISHED
tcp	0	0	172.16.56.200:42316	172.16.34.144:3306	ESTABLISHED
tcp	0	0	172.16.56.200:44076	172.16.240.74:6379	ESTABLISHED
tcp	0	0	172.16.56.200:49656	172.16.0.24:3306	ESTABLISHED
tcp	0	0	172.16.56.200:58248	100.100.142.4:80	TIME_WAIT
tcp	0	0	172.16.56.200:50108	172.16.0.24:3306	ESTABLISHED
tcp	0	0	172.16.56.200:41944	172.16.34.144:3306	ESTABLISHED
tcp	0	0	172.16.56.200:35548	100.100.32.118:80	TIME_WAIT
tcp	0	0	172.16.56.200:39024	100.100.45.106:443	TIME_WAIT
tcp	0	0	172.16.56.200:41788	172.16.34.144:3306	ESTABLISHED
tcp	0	0	172.16.56.200:58260	100.100.142.4:80	TIME_WAIT
tcp	0	0	172.16.56.200:41812	172.16.34.144:3306	ESTABLISHED
tcp	0	0	172.16.56.200:41854	172.16.34.144:3306	ESTABLISHED
tcp	0	0	172.16.56.200:58252	100.100.142.4:80	TIME_WAIT
tcp	0	0	172.16.56.200:49586	172.16.0.24:3306	ESTABLISHED
tcp	0	0	172.16.56.200:41754	172.16.34.144:3306	ESTABLISHED
tcp	0	0	172.16.56.200:50466	120.55.222.235:80	TIME_WAIT
tcp	0	0	172.16.56.200:38514	100.100.142.5:80	TIME_WAIT
tcp	0	0	172.16.56.200:49832	172.16.0.24:3306	ESTABLISHED
tcp	0	0	172.16.56.200:52162	100.100.30.25:80	ESTABLISHED
tcp	0	0	172.16.56.200:50372	172.16.0.24:3306	ESTABLISHED
tcp	0	0	172.16.56.200:50306	172.16.0.24:3306	ESTABLISHED
tcp	0	0	172.16.56.200:49600	172.16.0.24:3306	ESTABLISHED
tcp	0	0	172.16.56.200:41908	172.16.34.144:3306	ESTABLISHED
tcp	0	0	172.16.56.200:60292	100.100.142.1:80	TIME_WAIT
tcp	0	0	172.16.56.200:37650	100.100.54.133:80	TIME_WAIT
tcp	0	0	172.16.56.200:41938	172.16.34.144:3306	ESTABLISHED
tcp	0	0	172.16.56.200:49736	172.16.0.24:3306	ESTABLISHED
tcp	0	0	172.16.56.200:41890	172.16.34.144:3306	ESTABLISHED
udp	0	0	127.0.0.1:323	0.0.0.0:*	
udp	0	0	0.0.0.0:45881	0.0.0.0:*	
udp	0	0	127.0.0.53:53	0.0.0.0:*	
udp	0	0	172.16.56.200:68	0.0.0.0:*	

```
udp6       0      0 ::1:323                 :::*
raw6       0      0 :::58                   :::*                    7
```

现在需要你查看系统 tcp 连接中各个状态的连接数，并且按照连接数降序输出。代码如下:
[root@docker02 scripts]# grep -e 'tcp' nowcoder.txt |awk '{print $6}'|sort|uniq -c|sort -nr|awk '{print $2 " " $1}'

```
ESTABLISHED 22
TIME_WAIT 9
LISTEN 3
```

1.25.8 查看和 3306 端口建立的连接

现在需要你查看和本机 3306 端口建立连接并且状态是 established 的所有 IP，按照连接数降序排序。脚本如下:
grep -e '3306' nowcoder.txt |grep 'ESTABLISHED'|awk '{print $5}'|cut -d':' -f1|sort|uniq -c|sort -nr|awk '{print $1" "$2}'

1.25.9 输出每个 IP 的连接数

现在需要你输出每个 IP 的连接数，按照连接数降序排序。代码如下:
grep 'tcp' nowcoder.txt |awk '{print $5}'|cut -d':' -f1|sort|uniq -c|sort -nr|awk '{print $2 " "$1}'

```
172.16.34.144 10
172.16.0.24 10
100.100.142.4 3
0.0.0.0 3
172.16.240.74 1
120.55.222.235 1
100.100.54.133 1
100.100.45.106 1
100.100.32.118 1
100.100.30.25 1
100.100.142.5 1
100.100.142.1 1
```

1.25.10 输出和 3306 端口建立连接总的各个状态的数目

在需要你输出和本机 3306 端口建立连接的各个状态的数目。代码如下:

```
#!/bin/bash

total_ip=`grep -e '3306' nowcoder.txt|awk '{print $5}'|cut -d':' -f1|sort|uniq -c|wc -l`
total_link=`grep -e '3306' nowcoder.txt|wc -l`
total_state=`grep -e '3306' nowcoder.txt|awk '{print $6}'|sort|uniq -c|awk '{print $2 " " $1}'`

echo "TOTAL_IP $total_ip"
echo $total_state
```

echo "TOTAL_LINK $total_link"

```
TOTAL_IP 2
ESTABLISHED 20 TIME_WAIT 1
TOTAL_LINK 21
```

1.25.11 业务分析-提取值

描述
假设我们的日志 nowcoder.txt 里，内容如下
12-May-2017 10:02:22.789 信息 [main] org.apache.catalina.startup.VersionLoggerListener.log
Server version:Apache Tomcat/8.5.15
12-May-2017 10:02:22.813 信息 [main] org.apache.catalina.startup.VersionLoggerListener.log
Server built:May 5 2017 11:03:04 UTC
12-May-2017 10:02:22.813 信息 [main] org.apache.catalina.startup.VersionLoggerListener.log
Server number:8.5.15.0
12-May-2017 10:02:22.814 信息 [main] org.apache.catalina.startup.VersionLoggerListener.log
OS Name:Windows, OS Version:10
12-May-2017 10:02:22.814 信息 [main] org.apache.catalina.startup.VersionLoggerListener.log
Architecture:x86_64

现在需要你提取出对应的值 ，输出内容如下
serverVersion:Apache Tomcat/8.5.15
serverName:8.5.15.0
osName:Windows
osVersion:10

代码如下:

```
#!/bin/bash
serverVersion=`cat nowcoder.txt |grep 'Server version'|awk -F':' '{print $NF}'`
serverName=`cat nowcoder.txt |grep 'Server number'|awk -F':' '{print $NF}'`
osName=`cat nowcoder.txt |grep 'OS Name'|awk '{print $7}'|cut -d':' -f2|sed 's/,//g'`
osVersion=`cat nowcoder.txt |grep 'OS Version'|awk -F':' '{print $NF}'`

echo "serverVersion:$serverVersion"
echo "serverName:$serverName"
echo "osName:$osName"
echo "osVersion:$osVersion"
```

```
serverVersion:Apache Tomcat/8.5.15
serverName:8.5.15.0
osName:Windows
osVersion:10
```

1.25.12 ps 分析-统计 VSZ,RSS 各自总和

描述

假设命令运行的结果我们存储在 nowcoder.txt 里，格式如下：

USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND

root 1 0.0 0.1 37344 4604 ? Ss 2020 2:13 /sbin/init

root 231 0.0 1.5 166576 62740 ? Ss 2020 15:15 /lib/systemd/systemd-journald

root 237 0.0 0.0 0 0 ? S< 2020 2:06 [kworker/0:1H]

root 259 0.0 0.0 45004 3416 ? Ss 2020 0:25 /lib/systemd/systemd-udevd

root 476 0.0 0.0 0 0 ? S< 2020 0:00 [edac-poller]

root 588 0.0 0.0 276244 2072 ? Ssl 2020 9:49 /usr/lib/accountsservice/accounts-daemon

message+ 592 0.0 0.0 42904 3032 ? Ss 2020 0:01 /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation

root 636 0.0 0.0 65532 3200 ? Ss 2020 1:51 /usr/sbin/sshd -D

daemon 637 0.0 0.0 26044 2076 ? Ss 2020 0:00 /usr/sbin/atd -f

root 639 0.0 0.0 29476 2696 ? Ss 2020 3:29 /usr/sbin/cron -f

root 643 0.0 0.0 20748 1992 ? Ss 2020 0:26 /lib/systemd/systemd-logind

syslog 645 0.0 0.0 260636 3024 ? Ssl 2020 3:17 /usr/sbin/rsyslogd -n

root 686 0.0 0.0 773124 2836 ? Ssl 2020 26:45 /usr/sbin/nscd

root 690 0.0 0.0 19472 252 ? Ss 2020 14:39 /usr/sbin/irqbalance --pid=/var/run/irqbalance.pid

ntp 692 0.0 0.0 98204 776 ? Ss 2020 25:18 /usr/sbin/ntpd -p /var/run/ntpd.pid -g -u 108:114

uuidd 767 0.0 0.0 28624 192 ? Ss 2020 0:00 /usr/sbin/uuidd --socket-activation

root 793 0.0 0.0 128812 3148 ? Ss 2020 0:00 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;

www-data 794 0.0 0.2 133376 9120 ? S 2020 630:57 nginx: worker process

www-data 795 0.0 0.2 133208 8968 ? S 2020 633:02 nginx: worker process

www-data 796 0.0 0.2 133216 9120 ? S 2020 634:24 nginx: worker process

www-data 797 0.0 0.2 133228 9148 ? S 2020 632:56 nginx: worker process

web 955 0.0 0.0 36856 2112 ? Ss 2020 0:00 /lib/systemd/systemd --user

web 956 0.0 0.0 67456 1684 ? S 2020 0:00 (sd-pam)

root 1354 0.0 0.0 8172 440 tty1 Ss+ 2020 0:00 /sbin/agetty --noclear

tty1 linux
root 1355 0.0 0.0 7988 344 ttyS0 Ss+ 2020 0:00 /sbin/agetty --keep-baud 115200 38400 9600 ttyS0 vt220
root 2513 0.0 0.0 0 0 ? S 13:07 0:00 [kworker/u4:1]
root 2587 0.0 0.0 0 0 ? S 13:13 0:00 [kworker/u4:2]
root 2642 0.0 0.0 0 0 ? S 13:17 0:00 [kworker/1:0]
root 2679 0.0 0.0 0 0 ? S 13:19 0:00 [kworker/u4:0]
root 2735 0.0 0.1 102256 7252 ? Ss 13:24 0:00 sshd: web [priv]
web 2752 0.0 0.0 102256 3452 ? R 13:24 0:00 sshd: web@pts/0
web 2753 0.5 0.1 14716 4708 pts/0 Ss 13:24 0:00 -bash
web 2767 0.0 0.0 29596 1456 pts/0 R+ 13:24 0:00 ps aux
root 10634 0.0 0.0 0 0 ? S Nov16 0:00 [kworker/0:0]
root 16585 0.0 0.0 0 0 ? S< 2020 0:00 [bioset]
root 19526 0.0 0.0 0 0 ? S Nov16 0:00 [kworker/1:1]
root 28460 0.0 0.0 0 0 ? S Nov15 0:03 [kworker/0:2]
root 30685 0.0 0.0 36644 2760 ? Ss 2020 0:00 /lib/systemd/systemd --user
root 30692 0.0 0.0 67224 1664 ? S 2020 0:00 (sd-pam)
root 32689 0.0 0.0 47740 2100 ? Ss 2020 0:00 /usr/local/ilogtail/ilogtail
root 32691 0.2 0.5 256144 23708 ? Sl 2020 1151:31 /usr/local/ilogtail/ilogtail

现在需要你统计 VSZ，RSS 各自的总和（以 M 兆为统计），输出格式如下
MEM TOTAL
VSZ_SUM:3250.8M,RSS_SUM:179.777M

代码如下:

```
#!/bin/bash

VSZ_SUM=0
RSS_SUM=0
VSZ=`cat nowcoder.txt |awk '{print $5}'|sed -n '2,$'p`
RSS=`cat nowcoder.txt |awk '{print $6}'|sed -n '2,$'p`


for i in $VSZ
do
    VSZ_SUM=$(($VSZ_SUM + $i))
done

for n in $RSS
do
        RSS_SUM=$(($RSS_SUM + $n))
done
```

```
VSZ_SUM=`echo "scale=1;$VSZ_SUM / 1024"|bc`
RSS_SUM=`echo "scale=3;$RSS_SUM / 1024"|bc`

echo "MEM TOTAL"
echo "VSZ_SUM:${VSZ_SUM}M,RSS_SUM:${RSS_SUM}M"
```

1.26 处理文本

描述
假设我们有一个 nowcoder.txt，假设里面的内容如下
111:13443
222:13211
111:13643
333:12341
222:12123
现在需要你写一个脚本按照以下的格式输出
[111]
13443
13643
[222]
13211
12123
[333]
12341

代码如下:

```
#!/bin/bash
number01=`cat nowcoder.txt|awk -F':' '{print $1}'|sort|uniq -c|awk '{print $2}'`

for i in $number01
do
    number02=`grep "${i}" nowcoder.txt|awk -F':' '{print $2}'`
    echo "[${i}]"
    echo -e    "${number02}"

done
```

1.27 打印只有一个数字的行

描述
假设我们有一个 nowcoder.txt，现在需要你写脚本，打印只有一个数字的行。
假设 nowcoder.txt 内容如下

haha
1
2ab
cd
77
那么你的脚本应该输出
1
2ab

代码如下:

```
[root@docker02 scripts]# cat 3.sh
#!/bin/bash

all_number=`grep '[0-9]' nowcoder.txt`
for i in $all_number
do
    for n in $i
    do
        number01=`echo $i |grep -o '[0-9]'|wc -l`
        #-o  会选出条件的内容，不显示其他的内容
        if [ $number01 -eq 1 ];then
            echo $i

        fi
    done



done
```

1.28 保留近七天的数据

文件夹的命名是以日期形式命名的。

```
[root@docker03 recordings]# ls
20220901    20220903    20220905    20220907    20220909    20220911    20220913    20220915    20220917    20220919    20220921
2022-09-01  2022-09-03  2022-09-05  2022-09-07  2022-09-09  2022-09-11  2022-09-13  2022-09-15  2022-09-17  2022-09-19  2022-09-21
20220902    20220904    20220906    20220908    20220910    20220912    20220914    20220916    20220918    20220920    20220922
2022-09-02  2022-09-04  2022-09-06  2022-09-08  2022-09-10  2022-09-12  2022-09-14  2022-09-16  2022-09-18  2022-09-20  2022-09-22
```

脚本如下:

```
cat save_file.sh
#!/bin/bash
#Author: hhl
#Mail: 267@qq.com
#Function: Save file data for seven days
```

```
#Version: 1.1

source /etc/profile

historyDir=/home/recordings/
today=$(date +%Y-%m-%d)
#echo "---------today is $today----------"
tt=`date -d last-week +%Y-%m-%d`
#echo "next is to delete release before $tt"
tt1=`date -d $tt +%s`    #小于此数值的目录删掉
#echo $tt1
for file in ${historyDir}*
do
        if test -d $file
                then
                name=`basename $file`
#               echo $name
                curr=`date -d $name +%s`
                if [ $curr -le $tt1 ]
                        then
#                               echo " delete $name-------"
                                rm -rf ${historyDir}${name}
                fi
        fi
done
```

1.29 将文件内容追加到当天的文件里

需求: 有个签名服务器，签名日志只会留存 6 个最新 txt ,写一个脚本定时追加写入一个文件【没半小时获取一次就行】,然后这个文件是每天都能重新新生成一个，名字最好和日期相同

代码思路: first.sh 需要手动执行一次，之后不用运行，作用是将目录下的文件名称写入到 filename.txt 里，并将日志追加到日期文件里，file.sh 可配置成定时任务，将目录下的文件名称写入 filenames.txt 里，然后和 filename.txt 对比，得出需要追加的文件名称，然后追加写入日期文件

```
#########################################################
####cat first.sh
#!/bin/bash
today=`date +%Y-%m-%d`
log_dir=/log
#日志所在目录
save_dir=/data
#追加文件所在目录
```

```
save_filename_dir=/tmp
#文件名称所在目录


for filename in `ls ${log_dir}/*`
do
        cat ${filename} >> ${save_dir}/${today}.txt
                echo ${filename} >> ${save_filename_dir}/filename.txt
done


#########################################################################
############cat file.sh
#!/bin/bash
today=`date +%Y-%m-%d`
log_dir=/log
#日志所在目录
save_dir=/data
#追加文件所在目录
save_filename_dir=/tmp
#文件名称所在目录


for filenames in `ls ${log_dir}/*`
do
        echo ${filenames} >> ${save_filename_dir}/filenames.txt
done

add_file=`grep -F -v -f /tmp/filename.txt /tmp/filenames.txt`
#通过两个文件对比  得出最新的文件名
add_file_count=`grep -F -v -f /tmp/filename.txt /tmp/filenames.txt|wc -l`
if [ ${add_file_count} -ge 1 ];then
        cat ${add_file} >> ${save_dir}/${today}.txt
        echo ${add_file} >> /tmp/filename.txt
fi
rm -rf /tmp/filenames.txt
```

1.30 检查 nginx 进程个数

需求:nginx 进程数小于 4，则输出错误代码

```
ps -ef|grep nginx|wc -l |awk '{if($1<4){print "错误代码"}}'
```

错误代码

1.31 按照文件内容分类存储

需求：大概有 100 个小文件，需要将文件第五列的内容分类存储，第五列可能是不同的字符串。文件内容如下：

```
[root@docker03 scripts]# cat /data/pdb/complex.1.pdb |head -n20
ATOM      1  N   ASN A 190      50.421  39.031  57.732  2     1 1.63          -.15
ATOM      2  CA  ASN A 190      49.407  39.114  58.819  2     1 2.03           .10
ATOM      3  C   ASN A 190      49.552  38.112  59.989  2     1 1.67           .60
ATOM      4  O   ASN A 190      49.348  38.509  61.137  2     1 1.38          -.55
ATOM      5  CB  ASN A 190      47.994  39.055  58.233  2     0 1.99           .00
ATOM      6  CG  ASN A 190      47.616  40.321  57.498  2     1 1.67           .55
ATOM      7  ND2 ASN A 190      46.938  40.168  56.368  2     1 1.63           .00
ATOM      8  OD1 ASN A 190      47.931  41.425  57.941  2     1 1.38          -.55
ATOM      9  N   PRO A 191      49.883  36.819  59.726  3     0 1.63          -.25
ATOM     10  CA  PRO A 191      50.167  35.977  60.900  3     0 2.03           .10
ATOM     11  C   PRO A 191      51.540  36.240  61.515  3     1 1.67           .60
ATOM     12  O   PRO A 191      52.564  35.840  60.961  3     1 1.38          -.55
ATOM     13  CB  PRO A 191      50.085  34.548  60.348  3     1 1.99           .00
ATOM     14  CG  PRO A 191      50.376  34.684  58.896  3     1 1.99           .00
ATOM     15  CD  PRO A 191      49.805  36.014  58.489  3     1 1.99           .10
ATOM     16  N   MET A 192      51.554  36.911  62.659  3     0 1.63          -.15
ATOM     17  CA  MET A 192      52.777  37.074  63.422  3     1 2.03           .10
ATOM     18  C   MET A 192      53.212  35.720  63.975  3     0 1.67           .60
ATOM     19  O   MET A 192      52.385  34.936  64.449 12     0 1.38          -.55
ATOM     20  CB  MET A 192      52.550  38.063  64.564  7     1 1.99           .00
```

代码如下:

```
cat h_file.sh
#!/bin/bash
source /etc/profile

data_dir=/data/pdb
#pdb 源文件存放目录

deal_dir=/data/deal
#存放处理之后的 pdb 文件目录

cd ${data_dir}


cat *.pdb|awk '{print $5}'|sort|uniq > /tmp/t.txt
#获取所有 pdb 文件第五列去重之后的字符串存入文件/tmp/t.txt

for i in `cat /tmp/t.txt`
do
        cat ${data_dir}/*.pdb|awk '{if($5==b) {print $0}}' b="${i}" > ${deal_dir}/${i}.pdb
        #这里使用了 awk 外部引用变量的用法
done
```