

资料整理自网络，仅作免费交流分享，侵权删！

更多面试资料请关注微信公众号：程序员面试吧



Q:1 Shell脚本是什么、它是必需的吗？

答:一个Shell脚本是一个文本文件，包含一个或多个命令。作为系统管理员，我们经常需要使用多个命令来完成一项任务，我们可以添加这些所有命令在一个文本文件(Shell脚本)来完成这些日常工作任务。

Q:2 什么是默认登录shell，如何改变指定用户的登录shell

答:在Linux操作系统，“/bin/bash”是默认登录shell，是在创建用户时分配的。使用chsh命令可以改变默认的shell。示例如下所示：

```
# chsh <用户名> -s <新shell>
# chsh linuxtech1 -s /bin/sh
```

Q:3 可以在shell脚本中使用哪些类型的变量？

答：在shell脚本，我们可以使用两种类型的变量：

- 系统定义变量
- 用户定义变量

系统变量是由系统系统自己创建的。这些变量通常由大写字母组成，可以通过“set”命令查看。

用户变量由系统用户来生成和定义，变量的值可以通过命令“echo \$<变量名>”查看。

Q:4 如何将标准输出和错误输出同时重定向到同一位置？

答：这里有两个方法来实现：

方法一：

```
2>&1 (如# ls /usr/share/doc > out.txt 2>&1 )
```

方法二：

```
&> (如# ls /usr/share/doc &> out.txt )
```

Q:5 shell脚本中“if”语法如何嵌套？

答：基础语法如下：

```
if [ 条件 ]
then
命令1
命令2
...
else
if [ 条件 ]
then
命令1
命令2
...
else
命令1
命令2
...
fi
fi
```

Q:6 shell脚本中“\$?”标记的用途是什么？

答：在写一个shell脚本时，如果你想要检查前一命令是否执行成功，在if条件中使用“\$?”可以来检查前一命令的结束状态。简单的例子如下：

```
root@localhost:~# ls /usr/bin/shar
/usr/bin/shar
root@localhost:~# echo $?
0
```

如果结束状态是0，说明前一个命令执行成功。

```
root@localhost:~# ls /usr/bin/share
ls: cannot access /usr/bin/share: No such file or directory
root@localhost:~# echo $?
2
```

如果结束状态不是0，说明命令执行失败。

Q:7 在shell脚本中如何比较两个数字？

答：在if-then中使用测试命令（-gt 等）来比较两个数字，例子如下：

```
#!/bin/bash
x=10
y=20
if [ $x -gt $y ]
then
echo "x is greater than y"
else
echo "y is greater than x"
fi
```

Q:8 shell脚本中break命令的作用？

答：break命令一个简单的用途是退出执行中的循环。我们可以在while和until循环中使用break命令跳出循环。

Q:9 shell脚本中continue命令的作用？

答：continue命令不同于break命令，它只跳出当前循环的迭代，而不是整个循环。continue命令很多时候是很有用的，例如错误发生，但我们依然希望继续执行大循环的时候。

Q:10 告诉我shell脚本中Case语句的语法？

答：基础语法如下：

```
case 变量 in
    值1)
        命令1
        命令2
        ...
    最后命令
    !!
    值2)
        命令1
        命令2
        .....
    最后命令
;;
esac
```

Q:11 shell脚本中while循环语法？

答：如同for循环，while循环只要条件成立就重复它的命令块。不同于for循环，while循环会不断迭代，直到它的条件不为真。基础语法：

```
while [ 条件 ]
do
    命令...
done
```

Q:12 如何使脚本可执行？

答：使用chmod命令来使脚本可执行。例子如下：

```
# chmod a+x myscript.sh
```

Q:13 “#!/bin/bash”的作用？

答：#!/bin/bash是shell脚本的第一行，称为释伴（shebang）行。这里#符号叫做hash，而! 叫做bang。它的意思是命令通过 /bin/bash 来执行。

Q:14 shell脚本中for循环语法？

答：for循环的基础语法：

```
for 变量 in 循环列表
do
  命令1
  命令2
  ...
  最后命令
done
```

Q:15 如何调试shell脚本？

答：使用'-x'参数（sh -x myscript.sh）可以调试shell脚本。另一个种方法是使用'-nv'参数(sh -nv myscript.sh)。

Q:16 shell脚本如何比较字符串？

答：test命令可以用来比较字符串。测试命令会通过比较字符串中的每一个字符来比较。

Q:17 Bourne shell(bash) 中有哪些特殊的变量？

答：下面的表列出了Bourne shell为命令行设置的特殊变量。

内建变量	解释
\$0	命令行中的脚本名字
\$1	第一个命令行参数
\$2	第二个命令行参数
...
\$9	第九个命令行参数
\$#	命令行参数的数量
\$*	所有命令行参数，以空格隔开

Q:18 在shell脚本中，如何测试文件？

答：test命令可以用来测试文件。基础用法如下表格：

Test	用法
-d 文件名	如果文件存在并且是目录，返回true
-e 文件名	如果文件存在，返回true
-f 文件名	如果文件存在并且是普通文件，返回true
-r 文件名	如果文件存在并可读，返回true
-s 文件名	如果文件存在并且不为空，返回true
-w 文件名	如果文件存在并可写，返回true
-x 文件名	如果文件存在并可执行，返回true

Q:19 在shell脚本中，如何写入注释？

答：注释可以用来描述一个脚本可以做什么和它是如何工作的。每一行注释以#开头。例子如下：

```
#!/bin/bash
# This is a command
echo "I am logged in as $USER"
```

Q:20 如何让 shell 脚本得到来自终端的输入？

答：read命令可以读取来自终端（使用键盘）的数据。read命令得到用户的输入并置于你给出的变量中。例子如下：

```
# vi /tmp/test.sh
#!/bin/bash
echo 'Please enter your name'
read name
echo "My Name is $name"
# ./test.sh
Please enter your name
LinuxTechi
My Name is LinuxTechi
```

Q:21 如何取消变量或取消变量赋值？

答：“unset”命令用于取消变量或取消变量赋值。语法如下所示：

```
# unset <变量名>
```

Q:22 如何执行算术运算？

答：有两种方法来执行算术运算：

1.使用expr命令

```
# expr 5 + 2
```

2.用一个美元符号和方括号（\$[表达式]）例如：

```
test=$((16 + 4)) ; test=$((16 + 4))
```

Q:23 do-while语句的基本格式？

答：do-while语句类似于while语句，但检查条件语句之前先执行命令（LCTT 译注：意即至少执行一次。）。下面是用do-while语句的语法

```
do
{
    命令
} while (条件)
```

Q:24 在shell脚本如何定义函数呢？

答：函数是拥有名字的代码块。当我们定义代码块，我们就可以在我们的脚本调用函数名字，该块就会被执行。示例如下所示：

```
$ diskusage () { df -h ; }
译注：下面是我给的shell函数语法，原文没有
[ function ] 函数名 [()]
{
    命令；
    [return int;]
}
```

1-24题

原文：linuxtechi

译文：LCTT

链接：<http://linux.cn/article-5311-1.html>

Q:25 获取随机字符串或数字

获取随机8位字符串：


```
方法1:
# echo $RANDOM |md5sum |cut -c 1-8
471b94f2
方法2:
# openssl rand -base64 4
vg3BEg==
方法3:
# cat /proc/sys/kernel/random/uuid |cut -c 1-8
ed9e032c
```

获取随机8位数字:

```
方法1:
# echo $RANDOM |cksum |cut -c 1-8
23648321
方法2:
# openssl rand -base64 4 |cksum |cut -c 1-8
38571131
方法3:
# date +%N |cut -c 1-8
69024815
```

cksum: 打印CRC效验和统计字节

Q:26 定义一个颜色输出字符串函数

```
方法1:
function echo_color() {
    if [ $1 == "green" ]; then
        echo -e "\033[32;40m$2\033[0m"
    elif [ $1 == "red" ]; then
        echo -e "\033[31;40m$2\033[0m"
    fi
}
方法2:
function echo_color() {
    case $1 in
        green)
            echo -e "\033[32;40m$2\033[0m"
            ;;
        red)
            echo -e "\033[31;40m$2\033[0m"
            ;;
        *)
            echo "Example: echo_color red string"
    esac
}
使用方法: echo_color green "test"
```

function关键字定义一个函数，可加或不加。

Q:27 批量创建用户

```
#!/bin/bash
DATE=$(date +%F_%T)
USER_FILE=user.txt
echo_color(){
    if [ $1 == "green" ]; then
        echo -e "\033[32;40m$2\033[0m"
    elif [ $1 == "red" ]; then
        echo -e "\033[31;40m$2\033[0m"
    fi
}
# 如果用户文件存在并且大小大于0就备份
if [ -s $USER_FILE ]; then
    mv $USER_FILE ${USER_FILE}-${DATE}.bak
    echo_color green "$USER_FILE exist, rename ${USER_FILE}-${DATE}.bak"
fi
echo -e "User\tPassword" >> $USER_FILE
echo "-----" >> $USER_FILE
for USER in user{1..10}; do
    if ! id $USER &>/dev/null; then
        PASS=$(echo $RANDOM |md5sum |cut -c 1-8)
        useradd $USER
        echo $PASS |passwd --stdin $USER &>/dev/null
        echo -e "$USER\t$PASS" >> $USER_FILE
        echo "$USER User create successful."
    else
        echo_color red "$USER User already exists!"
    fi
done
```

Q:28 检查软件包是否安装

```
#!/bin/bash
if rpm -q sysstat &>/dev/null; then
    echo "sysstat is already installed."
else
    echo "sysstat is not installed!"
fi
```

Q:29 检查服务状态

```
#!/bin/bash
PORT_C=$(ss -anu |grep -c 123)
PS_C=$(ps -ef |grep ntpd |grep -vc grep)
if [ $PORT_C -eq 0 -o $PS_C -eq 0 ]; then
    echo "内容" | mail -s "主题" dst@example.com
fi
```

Q:30 检查主机存活状态

方法1：将错误IP放到数组里面判断是否ping失败三次

```
#!/bin/bash
IP_LIST="192.168.18.1 192.168.1.1 192.168.18.2"
for IP in $IP_LIST; do
    NUM=1
    while [ $NUM -le 3 ]; do
        if ping -c 1 $IP > /dev/null; then
            echo "$IP Ping is successful."
            break
        else
            # echo "$IP Ping is failure $NUM"
            FAIL_COUNT[$NUM]=$IP
            let NUM++
        fi
    done
    if [ ${#FAIL_COUNT[*]} -eq 3 ];then
        echo "${FAIL_COUNT[1]} Ping is failure!"
        unset FAIL_COUNT[*]
    fi
done
```

方法2：将错误次数放到FAIL_COUNT变量里面判断是否ping失败三次

```
#!/bin/bash
IP_LIST="192.168.18.1 192.168.1.1 192.168.18.2"
for IP in $IP_LIST; do
    FAIL_COUNT=0
    for ((i=1;i<=3;i++)); do
        if ping -c 1 $IP >/dev/null; then
            echo "$IP Ping is successful."
            break
        else
            # echo "$IP Ping is failure $i"
            let FAIL_COUNT++
        fi
    done
    if [ $FAIL_COUNT -eq 3 ]; then
        echo "$IP Ping is failure!"
    fi
done
```

方法3：利用for循环将ping通就跳出循环继续，如果不跳出就会走到打印ping失败

```
#!/bin/bash
ping_success_status() {
    if ping -c 1 $IP >/dev/null; then
        echo "$IP Ping is successful."
        continue
    fi
}
IP_LIST="192.168.18.1 192.168.1.1 192.168.18.2"
for IP in $IP_LIST; do
    ping_success_status
    ping_success_status
    ping_success_status
    echo "$IP Ping is failure!"
done
```

Q:31 监控CPU、内存和硬盘利用率

1) CPU

借助vmstat工具来分析CPU统计信息。

```
#!/bin/bash
DATE=$(date +%F" "%H:%M)
IP=$(ifconfig eth0 |awk -F '[' :]+ ' '/inet addr/{print $4}') # 只支持CentOS6
MAIL="example@mail.com"
if ! which vmstat &>/dev/null; then
    echo "vmstat command no found, please install procps package."
    exit 1
fi
US=$(vmstat |awk 'NR==3{print $13}')
SY=$(vmstat |awk 'NR==3{print $14}')
IDLE=$(vmstat |awk 'NR==3{print $15}')
WAIT=$(vmstat |awk 'NR==3{print $16}')
USE=$((US+$SY))
if [ $USE -ge 50 ]; then
    echo "
    Date: $DATE
    Host: $IP
    Problem: CPU utilization $USE
    " | mail -s "CPU Monitor" $MAIL
fi
```

2) 内存

```
#!/bin/bash
DATE=$(date +%F" "%H:%M)
IP=$(ifconfig eth0 |awk -F '[' :]+ ' '/inet addr/{print $4}')
MAIL="example@mail.com"
TOTAL=$(free -m |awk '/Mem/{print $2}')
USE=$(free -m |awk '/Mem/{print $3-$6-$7}')
FREE=$((TOTAL-USE))
# 内存小于1G发送报警邮件
if [ $FREE -lt 1024 ]; then
    echo "
```

```

Date: $DATE
Host: $IP
Problem: Total=$TOTAL,Use=$USE,Free=$FREE
" | mail -s "Memory Monitor" $MAIL
fi

```

3) 硬盘

```

#!/bin/bash
DATE=$(date +%F" "%H:%M)
IP=$(ifconfig eth0 |awk -F '[:]+' '/inet addr/{print $4}')
MAIL="example@mail.com"
TOTAL=$(fdisk -l |awk -F '[:]+' 'BEGIN{OFS="="/}^Disk \dev/{printf "%s=%sG,",$2,$3}')
PART_USE=$(df -h |awk 'BEGIN{OFS="="/}^\\dev/{print $1,int($5),$6}')
for i in $PART_USE; do
    PART=$(echo $i |cut -d"=" -f1)
    USE=$(echo $i |cut -d"=" -f2)
    MOUNT=$(echo $i |cut -d"=" -f3)
    if [ $USE -gt 80 ]; then
        echo "
        Date: $DATE
        Host: $IP
        Total: $TOTAL
        Problem: $PART=$USE($MOUNT)
        " | mail -s "Disk Monitor" $MAIL
    fi
done

```

Q:32 批量主机磁盘利用率监控

前提监控端和被监控端SSH免交互登录或者密钥登录。

写一个配置文件保存被监控主机SSH连接信息，文件内容格式： IP User Port

```

#!/bin/bash
HOST_INFO=host.info
for IP in $(awk '/^[^#]/{print $1}' $HOST_INFO); do
    USER=$(awk -v ip=$IP 'ip==$1{print $2}' $HOST_INFO)
    PORT=$(awk -v ip=$IP 'ip==$1{print $3}' $HOST_INFO)
    TMP_FILE=/tmp/disk.tmp
    ssh -p $PORT $USER@$IP 'df -h' > $TMP_FILE
    USE_RATE_LIST=$(awk 'BEGIN{OFS="="/}^\\dev/{print $1,int($5)}' $TMP_FILE)
    for USE_RATE in $USE_RATE_LIST; do
        PART_NAME=${USE_RATE%=*}
        USE_RATE=${USE_RATE#*=}
        if [ $USE_RATE -ge 80 ]; then
            echo "warning: $PART_NAME Partition usage $USE_RATE%!"
        fi
    done
done
done

```

Q:33 检查网站可用性

1) 检查URL可用性

方法1:

```
check_url() {
    HTTP_CODE=$(curl -o /dev/null --connect-timeout 3 -s -w "%{http_code}" $1)
    if [ $HTTP_CODE -ne 200 ]; then
        echo "Warning: $1 Access failure!"
    fi
}
```

方法2:

```
check_url() {
    if ! wget -T 10 --tries=1 --spider $1 >/dev/null 2>&1; then
        # -T超时时间, --tries尝试1次, --spider爬虫模式
        echo "Warning: $1 Access failure!"
    fi
}
```

使用方法: check_url www.baidu.com

2) 判断三次URL可用性

思路与上面检查主机存活状态一样。

方法1: 利用循环技巧, 如果成功就跳出当前循环, 否则执行到最后一行

```
#!/bin/bash
check_url() {
    HTTP_CODE=$(curl -o /dev/null --connect-timeout 3 -s -w "%{http_code}" $1)
    if [ $HTTP_CODE -eq 200 ]; then
        continue
    fi
}
URL_LIST="www.baidu.com www.agasgf.com"
for URL in $URL_LIST; do
    check_url $URL
    check_url $URL
    check_url $URL
    echo "Warning: $URL Access failure!"
done
```

方法2: 错误次数保存到变量

```
#!/bin/bash
URL_LIST="www.baidu.com www.agasgf.com"
for URL in $URL_LIST; do
    FAIL_COUNT=0
```

```

        for ((i=1;i<=3;i++)); do
            HTTP_CODE=$(curl -o /dev/null --connect-timeout 3 -s -w "%{http_code}"
$URL)
            if [ $HTTP_CODE -ne 200 ]; then
                let FAIL_COUNT++
            else
                break
            fi
        done
        if [ $FAIL_COUNT -eq 3 ]; then
            echo "Warning: $URL Access failure!"
        fi
    done

```

方法3：错误次数保存到数组

```

#!/bin/bash
URL_LIST="www.baidu.com www.agasgf.com"
for URL in $URL_LIST; do
    NUM=1
    while [ $NUM -le 3 ]; do
        HTTP_CODE=$(curl -o /dev/null --connect-timeout 3 -s -w "%{http_code}"
$URL)
        if [ $HTTP_CODE -ne 200 ]; then
            FAIL_COUNT[$NUM]=$IP #创建数组，以$NUM下标，$IP元素
            let NUM++
        else
            break
        fi
    done
    if [ ${#FAIL_COUNT[*]} -eq 3 ]; then
        echo "Warning: $URL Access failure!"
        unset FAIL_COUNT[*] #清空数组
    fi
done

```

Q:34 检查MySQL主从同步状态

```

#!/bin/bash
USER=bak
PASSWD=123456
IO_SQL_STATUS=$(mysql -u$USER -p$PASSWD -e 'show slave status\G' |awk -F:
'/slave_.*_Running/{gsub(": ","");print $0}') #gsub去除冒号后面的空格
for i in $IO_SQL_STATUS; do
    THREAD_STATUS_NAME=${i%:*}
    THREAD_STATUS=${i#*:}
    if [ "$THREAD_STATUS" != "Yes" ]; then
        echo "Error: MySQL Master-Slave $THREAD_STATUS_NAME status is
$THREAD_STATUS!"
    fi
done

```

Q:35 屏蔽网站访问频繁IP

11、iptables自动屏蔽访问网站频繁的IP

场景：恶意访问,安全防范

1) 屏蔽每分钟访问超过200的IP

方法1：根据访问日志（Nginx为例）

```
#!/bin/bash
DATE=$(date +%d/%b/%Y:%H:%M)
ABNORMAL_IP=$(tail -n5000 access.log |grep $DATE |awk '{a[$1]++}END{for(i in a)if(a[i]>100)print i}')
#先tail防止文件过大，读取慢，数字可调整每分钟最大的访问量。awk不能直接过滤日志，因为包含特殊字符。
for IP in $ABNORMAL_IP; do
    if [ $(iptables -vnL |grep -c "$IP") -eq 0 ]; then
        iptables -I INPUT -s $IP -j DROP
    fi
done
```

方法2：通过TCP建立的连接

```
#!/bin/bash
ABNORMAL_IP=$(netstat -an |awk '$4~/:80$/ && $6~/ESTABLISHED/{gsub(/:[0-9]+/, "", $5);{a[$5]++}}END{for(i in a)if(a[i]>100)print i}')
#gsub是将第五列（客户端IP）的冒号和端口去掉
for IP in $ABNORMAL_IP; do
    if [ $(iptables -vnL |grep -c "$IP") -eq 0 ]; then
        iptables -I INPUT -s $IP -j DROP
    fi
done
```

2) 屏蔽每分钟SSH尝试登录超过10次的IP

方法1：通过lastb获取登录状态:

```
#!/bin/bash
DATE=$(date +"%a %b %e %H:%M") #星期月天时分 %e单数字时显示7，而%d显示07
ABNORMAL_IP=$(lastb |grep "$DATE" |awk '{a[$3]++}END{for(i in a)if(a[i]>10)print i}')
for IP in $ABNORMAL_IP; do
    if [ $(iptables -vnL |grep -c "$IP") -eq 0 ]; then
        iptables -I INPUT -s $IP -j DROP
    fi
done
```

方法2：通过日志获取登录状态


```
#!/bin/bash
DATE=$(date +%b %d %H")
ABNORMAL_IP="$(tail -n10000 /var/log/auth.log |grep "$DATE" |awk
'/Failed/{a[$(NF-3)]++}END{for(i in a){if(a[i]>5){print i}}}")
for IP in $ABNORMAL_IP; do
    if [ $(iptables -vnL |grep -c "$IP") -eq 0 ]; then
        iptables -A INPUT -s $IP -j DROP
        echo "$(date +%F %T)" - iptables -A INPUT -s $IP -j DROP" >>~/ssh-login-
limit.log
    fi
done
```

Q:36 判断用户输入的是否为IP地址

方法1:

```
#!/bin/bash
function check_ip(){
    IP=$1
    VALID_CHECK=$(echo $IP|awk -F. '{1<=255&&2<=255&&3<=255&&4<=255}{print
"yes"}')
    if echo $IP|grep -E "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]
{1,3}$">/dev/null; then
        if [ $VALID_CHECK == "yes" ]; then
            echo "$IP available."
        else
            echo "$IP not available!"
        fi
    else
        echo "Format error!"
    fi
}
check_ip 192.168.1.1
check_ip 256.1.1.1
```

方法2:

```
#!/bin/bash
function check_ip(){
    IP=$1
    if [[ $IP =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]; then
        FIELD1=$(echo $IP|cut -d. -f1)
        FIELD2=$(echo $IP|cut -d. -f2)
        FIELD3=$(echo $IP|cut -d. -f3)
        FIELD4=$(echo $IP|cut -d. -f4)
        if [ $FIELD1 -le 255 -a $FIELD2 -le 255 -a $FIELD3 -le 255 -a $FIELD4 -le
255 ]; then
            echo "$IP available."
        else
            echo "$IP not available!"
        fi
    else
        echo "Format error!"
    fi
}
```

```
}  
check_ip 192.168.1.1  
check_ip 256.1.1.1
```

增加版:

加个死循环, 如果IP可用就退出, 不可用提示继续输入, 并使用awk判断。

```
#!/bin/bash  
function check_ip(){  
    local IP=$1  
    VALID_CHECK=$(echo $IP|awk -F. '{ $1<=255&&$2<=255&&$3<=255&&$4<=255{print  
"yes"}}')  
    if echo $IP|grep -E "^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$"  
>/dev/null; then  
        if [ $VALID_CHECK == "yes" ]; then  
            return 0  
        else  
            echo "$IP not available!"  
            return 1  
        fi  
    else  
        echo "Format error! Please input again."  
        return 1  
    fi  
}  
while true; do  
    read -p "Please enter IP: " IP  
    check_ip $IP  
    [ $? -eq 0 ] && break || continue  
done
```

Q:37判断用户输入的是否为数字

方法1:

```
#!/bin/bash  
if [[ $1 =~ ^[0-9]+$ ]]; then  
    echo "Is Number."  
else  
    echo "No Number."  
fi
```

方法2:

```
#!/bin/bash  
if [ $1 -gt 0 ] 2>/dev/null; then  
    echo "Is Number."  
else  
    echo "No Number."  
fi
```

方法3:

```
#!/bin/bash
echo $1 |awk '{print $0~/^[0-9]+$/?"Is Number.":"No Number."}' #三目运算符
12.14 找出包含关键字的文件
DIR=$1
KEY=$2
for FILE in $(find $DIR -type f); do
    if grep $KEY $FILE &>/dev/null; then
        echo "--> $FILE"
    fi
done
```

Q:38 给定目录找出包含关键字的文件

```
#!/bin/bash
DIR=$1
KEY=$2
for FILE in $(find $DIR -type f); do
    if grep $KEY $FILE &>/dev/null; then
        echo "--> $FILE"
    fi
done
```

Q:39 监控目录，将新创建的文件名追加到日志中

场景：记录目录下文件操作。

需先安装inotify-tools软件包。

```
#!/bin/bash
MON_DIR=/opt
inotifywait -mq --format %f -e create $MON_DIR | \
while read files; do
    echo $files >> test.log
done
```

Q:40 给用户提供多个网卡选择

场景：服务器多个网卡时，获取指定网卡，例如网卡流量

```
#!/bin/bash
function local_nic() {
    local NUM ARRAY_LENGTH
    NUM=0
    for NIC_NAME in $(ls /sys/class/net|grep -vE "lo|docker0"); do
        NIC_IP=$(ifconfig $NIC_NAME |awk -F'[: ]+' '/inet addr/{print $4}')
    done
}
```

```

        if [ -n "$NIC_IP" ]; then
            NIC_IP_ARRAY[$NUM]="$NIC_NAME:$NIC_IP"      #将网卡名和对应IP放到数组
            let NUM++
        fi
    done
    ARRAY_LENGTH=${#NIC_IP_ARRAY[*]}
    if [ $ARRAY_LENGTH -eq 1 ]; then      #如果数组里面只有一条记录说明就一个网卡
        NIC=${NIC_IP_ARRAY[0]:*}
        return 0
    elif [ $ARRAY_LENGTH -eq 0 ]; then    #如果没有记录说明没有网卡
        echo "No available network card!"
        exit 1
    else
        #如果有多条记录则提醒输入选择
        for NIC in ${NIC_IP_ARRAY[*]}; do
            echo $NIC
        done
        while true; do
            read -p "Please enter local use to network card name: "
            INPUT_NIC_NAME
            COUNT=0
            for NIC in ${NIC_IP_ARRAY[*]}; do
                NIC_NAME=${NIC:*}
                if [ $NIC_NAME == "$INPUT_NIC_NAME" ]; then
                    NIC=${NIC_IP_ARRAY[$COUNT]:*}
                    return 0
                else
                    COUNT+=1
                fi
            done
            echo "Not match! Please input again."
        done
    fi
}
local_nic

```

Q:41 查看网卡实时流量

适用于CentOS6操作系统。

```

#!/bin/bash
# Description: Only CentOS6
traffic_unit_conv() {
    local traffic=$1
    if [ $traffic -gt 1024000 ]; then
        printf "%.1f%s" "$(($traffic/1024/1024))" "MB/s"
    elif [ $traffic -lt 1024000 ]; then
        printf "%.1f%s" "$(($traffic/1024))" "KB/s"
    fi
}
NIC=$1
echo -e " In ----- Out"
while true; do
    OLD_IN=$(awk -F'[: ]+' ' $0~'"$NIC"'{print $3}' /proc/net/dev)
    OLD_OUT=$(awk -F'[: ]+' ' $0~'"$NIC"'{print $11}' /proc/net/dev)
    sleep 1
    NEW_IN=$(awk -F'[: ]+' ' $0~'"$NIC"'{print $3}' /proc/net/dev)

```

```

NEW_OUT=$(awk -F'[: ]+' ' $0~'$NIC''{print $11}' /proc/net/dev)
IN=$((NEW_IN-OLD_IN))
OUT=$((NEW_OUT-OLD_OUT))
echo "$(traffic_unit_conv $IN) $(traffic_unit_conv $OUT)"
sleep 1
done

```

使用: ./traffic.sh eth0

Q:42、MySQL数据库备份

```

#!/bin/bash
DATE=$(date +%F_%H-%M-%S)
HOST=192.168.1.120
DB=test
USER=bak
PASS=123456
MAIL="zhangsan@example.com lisi@example.com"
BACKUP_DIR=/data/db_backup
SQL_FILE=${DB}_full_${DATE}.sql
BAK_FILE=${DB}_full_${DATE}.zip
cd $BACKUP_DIR
if mysqldump -h$HOST -u$USER -p$PASS --single-transaction --routines --triggers -
B $DB > $SQL_FILE; then
    zip $BAK_FILE $SQL_FILE && rm -f $SQL_FILE
    if [ ! -s $BAK_FILE ]; then
        echo "$DATE 内容" | mail -s "主题" $MAIL
    fi
else
    echo "$DATE 内容" | mail -s "主题" $MAIL
fi
find $BACKUP_DIR -name '*.zip' -ctime +14 -exec rm {} \;

```

Q:43、Nginx服务管理脚本

场景: 使用源码包安装Nginx不含带服务管理脚本, 也就是不能使用"service nginx start"或"/etc/init.d/nginx start", 所以写了以下的服务管理脚本。

```

#!/bin/bash
# Description: Only support RedHat system
. /etc/init.d/functions
WORD_DIR=/usr/local/nginx
DAEMON=$WORD_DIR/sbin/nginx
CONF=$WORD_DIR/conf/nginx.conf
NAME=nginx
PID=$(awk -F'[: ]+' '/^[^#]/{if($0~/pid;/)print $2}' $CONF)
if [ -z "$PID" ]; then
    PID=$WORD_DIR/logs/nginx.pid
else
    PID=$WORD_DIR/$PID
fi

```

```

stop() {
    $DAEMON -s stop
    sleep 1
    [ ! -f $PID ] && action "* Stopping $NAME" /bin/true || action "* Stopping
$NAME" /bin/false
}
start() {
    $DAEMON
    sleep 1
    [ -f $PID ] && action "* Starting $NAME" /bin/true || action "* Starting
$NAME" /bin/false
}
reload() {
    $DAEMON -s reload
}
test_config() {
    $DAEMON -t
}
case "$1" in
    start)
        if [ ! -f $PID ]; then
            start
        else
            echo "$NAME is running..."
            exit 0
        fi
        ;;
    stop)
        if [ -f $PID ]; then
            stop
        else
            echo "$NAME not running!"
            exit 0
        fi
        ;;
    restart)
        if [ ! -f $PID ]; then
            echo "$NAME not running!"
            start
        else
            stop
            start
        fi
        ;;
    reload)
        reload
        ;;
    testconfig)
        test_config
        ;;
    status)
        [ -f $PID ] && echo "$NAME is running..." || echo "$NAME not running!"
        ;;
    *)
        echo "Usage: $0 {start|stop|restart|reload|testconfig|status}"
        exit 3
        ;;
esac

```

Q:44 用户根据菜单选择要连接的Linux主机

Linux主机SSH连接信息:

```
# cat host.txt
web 192.168.1.10 root 22
DB 192.168.1.11 root 22
```

内容格式: 主机名 IP User Port

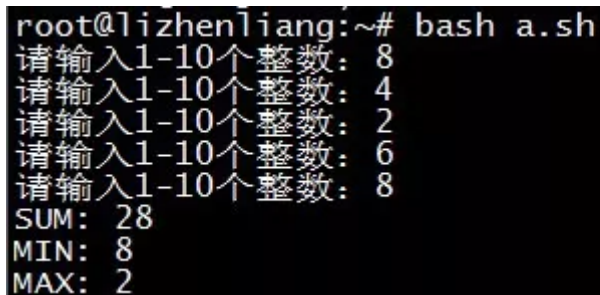
```
#!/bin/bash
PS3="Please input number: "
HOST_FILE=host.txt
while true; do
    select NAME in $(awk '{print $1}' $HOST_FILE) quit; do
        [ ${NAME:=empty} == "quit" ] && exit 0
        IP=$(awk -v NAME=${NAME} '$1==NAME{print $2}' $HOST_FILE)
        USER=$(awk -v NAME=${NAME} '$1==NAME{print $3}' $HOST_FILE)
        PORT=$(awk -v NAME=${NAME} '$1==NAME{print $4}' $HOST_FILE)
        if [ $IP ]; then
            echo "Name: $NAME, IP: $IP"
            ssh -o StrictHostKeyChecking=no -p $PORT -i id_rsa $USER@$IP # 密钥免
交互登录
            break
        else
            echo "Input error, Please enter again!"
            break
        fi
    done
done
```

Q:45 从FTP服务器下载文件

```
#!/bin/bash
if [ $# -ne 1 ]; then
    echo "Usage: $0 filename"
fi
dir=$(dirname $1)
file=$(basename $1)
ftp -n -v << EOF # -n 自动登录
open 192.168.1.10 # ftp服务器
user admin password
binary # 设置ftp传输模式为二进制, 避免MD5值不同或.tar.gz压缩包格式错误
cd $dir
get "$file"
EOF
```

Q:46 连续输入5个100以内的数字，统计和、最小和最大

```
#!/bin/bash
COUNT=1
SUM=0
MIN=0
MAX=100
while [ $COUNT -le 5 ]; do
    read -p "请输入1-10个整数: " INT
    if [[ ! $INT =~ ^[0-9]+$ ]]; then
        echo "输入必须是整数!"
        exit 1
    elif [[ $INT -gt 100 ]]; then
        echo "输入必须是100以内!"
        exit 1
    fi
    SUM=$((SUM+$INT))
    [ $MIN -lt $INT ] && MIN=$INT
    [ $MAX -gt $INT ] && MAX=$INT
    let COUNT++
done
echo "SUM: $SUM"
echo "MIN: $MIN"
echo "MAX: $MAX"
```



```
root@lizhenliang:~# bash a.sh
请输入1-10个整数: 8
请输入1-10个整数: 4
请输入1-10个整数: 2
请输入1-10个整数: 6
请输入1-10个整数: 8
SUM: 28
MIN: 2
MAX: 8
```

Q:47 将结果分别赋值给变量

应用场景：希望将执行结果或者位置参数赋值给变量，以便后续使用。

方法1:

```
for i in $(echo "4 5 6"); do
    eval a$i=$i
done
echo $a4 $a5 $a6
```

方法2：将位置参数192.168.1.1{1,2}拆分为到每个变量


```
num=0
for i in $(eval echo $*);do    #eval将{1,2}分解为1 2
    let num+=1
    eval node${num}="$i"
done
echo $node1 $node2 $node3
# bash a.sh 192.168.1.1{1,2}
192.168.1.11 192.168.1.12
```

方法3:

```
arr=(4 5 6)
INDEX1=$(echo ${arr[0]})
INDEX2=$(echo ${arr[1]})
INDEX3=$(echo ${arr[2]})
```

Q:48 批量修改文件名

示例:

```
# touch article_{1..3}.html
# ls
article_1.html  article_2.html  article_3.html
```

目的: 把article改为bbs

方法1:

```
for file in $(ls *.html); do
    mv $file bbs_${file#*_}
    # mv $file $(echo $file |sed -r 's/.*(._.)/bbs\1/')
    # mv $file $(echo $file |echo bbs_$(cut -d_ -f2))
done
```

方法2:

```
for file in $(find . -maxdepth 1 -name "*.html"); do
    mv $file bbs_${file#*_}
done
```

方法3:

```
# rename article bbs *.html
```

Q:49 统计当前目录中以.html结尾的文件总大

方法1:

```
# find . -name "*.html" -exec du -k {} \; | awk '{sum+=$1}END{print sum}'
```

方法2:

```
for size in $(ls -l *.html | awk '{print $5}'); do
    sum=$((sum+$size))
done
echo $sum
```

Q:50 扫描主机端口状态

```
#!/bin/bash
HOST=$1
PORT="22 25 80 8080"
for PORT in $PORT; do
    if echo &>/dev/null > /dev/tcp/$HOST/$PORT; then
        echo "$PORT open"
    else
        echo "$PORT close"
    fi
done
```

Q:51 Expect实现SSH免交互执行命令

Expect是一个自动交互式应用程序的工具，如telnet，ftp，passwd等。

需先安装expect软件包。

方法1：EOF标准输出作为expect标准输入

```
#!/bin/bash
USER=root
PASS=123.com
IP=192.168.1.120
expect << EOF
set timeout 30
spawn ssh $USER@$IP
expect {
    "(yes/no)" {send "yes\r"; exp_continue}
    "password:" {send "$PASS\r"}
}
expect "$USER@*" {send "$1\r"}
expect "$USER@*" {send "exit\r"}
expect eof
EOF
```

方法2:

```
#!/bin/bash
USER=root
PASS=123.com
IP=192.168.1.120
expect -c "
    spawn ssh $USER@$IP
    expect {
        \"(yes/no)\" {send \"yes\r\"; exp_continue}
        \"password:\" {send \"${PASS}\r\"; exp_continue}
        \"${USER}@*\" {send \"df -h\r exit\r\"; exp_continue}
    }"
```

方法3: 将expect脚本独立出来

登录脚本:

```
# cat login.exp
#!/usr/bin/expect
set ip [lindex $argv 0]
set user [lindex $argv 1]
set passwd [lindex $argv 2]
set cmd [lindex $argv 3]
if { $argc != 4 } {
    puts "Usage: expect login.exp ip user passwd"
    exit 1
}
set timeout 30
spawn ssh $user@$ip
expect {
    "(yes/no)" {send "yes\r"; exp_continue}
    "password:" {send "${passwd}\r"}
}
expect "$user@*" {send "$cmd\r"}
expect "$user@*" {send "exit\r"}
expect eof
```

执行命令脚本: 写个循环可以批量操作多台服务器

```
#!/bin/bash
HOST_INFO=user_info.txt
for ip in $(awk '{print $1}' $HOST_INFO)
do
    user=$(awk -v I="$ip" 'I==$1{print $2}' $HOST_INFO)
    pass=$(awk -v I="$ip" 'I==$1{print $3}' $HOST_INFO)
    expect login.exp $ip $user $pass $1
done
```

Linux主机SSH连接信息:

```
# cat user_info.txt
192.168.1.120 root 123456
```

Q:52 批量修改服务器用户密码

Linux主机SSH连接信息：旧密码

```
# cat old_pass.txt
192.168.18.217 root 123456 22
192.168.18.218 root 123456 22
```

内容格式：IP User Password Port

SSH远程修改密码脚本：新密码随机生成

```
#!/bin/bash
OLD_INFO=old_pass.txt
NEW_INFO=new_pass.txt
for IP in $(awk '/^[^#]/{print $1}' $OLD_INFO); do
    USER=$(awk -v I=$IP 'I==$1{print $2}' $OLD_INFO)
    PASS=$(awk -v I=$IP 'I==$1{print $3}' $OLD_INFO)
    PORT=$(awk -v I=$IP 'I==$1{print $4}' $OLD_INFO)
    NEW_PASS=$(mkpasswd -l 8) # 随机密码
    echo "$IP $USER $NEW_PASS $PORT" >> $NEW_INFO
    expect -c "
    spawn ssh -p$PORT $USER@$IP
    set timeout 2
    expect {
        \"(yes/no)\" {send \"yes\r\";exp_continue}
        \"password:\" {send \"$PASS\r\";exp_continue}
        \"$USER@*\" {send \"echo '$NEW_PASS' |passwd --stdin $USER\r
    exit\r\";exp_continue}
    }"
done
```

生成新密码文件：

```
# cat new_pass.txt
192.168.18.217 root n8wX3mU% 22
192.168.18.218 root c87;ZnnL 22
```

Q:53 打印乘法口诀

方法1：

```
# awk 'BEGIN{for(n=0;n++<9;){for(i=0;i++<n;)printf i"x"n"="i*n" ";print ""}}'
```

方法2：

```

for ((i=1;i<=9;i++)); do
    for ((j=1;j<=i;j++)); do
        result=$((i*j))
        echo -n "$j*i=$result "
    done
done
echo
done

```

Q:54 getopt工具完善脚本命令行参数

getopts是一个解析脚本选项参数的工具。

命令格式: getopt optstring name [arg]

初次使用你要注意几点:

- 脚本位置参数会与optstring中的单个字母逐个匹配, 如果匹配到就赋值给name, 否则赋值name为问号;
- optstring中单个字母是一个选项, 如果字母后面加冒号, 表示该选项后面带参数, 参数值并会赋值给OPTARG变量;
- optstring中第一个是冒号, 表示屏蔽系统错误 (test.sh: illegal option -- h) ;
- 允许把选项放一起, 例如-ab

下面写一个打印文件指定行的简单例子, 引导你思路:

```

#!/bin/bash
while getopt :f:n: option; do
    case $option in
        f)
            FILE=$OPTARG
            [ ! -f $FILE ] && echo "$FILE File not exist!" && exit
            ;;
        n)
            sed -n "${OPTARG}p" $FILE
            ;;
        ?)
            echo "Usage: $0 -f <file_path> -n <line_number>"
            echo "-f, --file          specified file"
            echo "-n, --line-number    print specified line"
            exit 1
            ;;
    esac
done

```

```

root@lizhenliang:~# bash a.sh -h
Usage: a.sh -f <file_path> -n <line_number>
-f, --file          specified file
-n, --line-number    print specified line
root@lizhenliang:~# bash a.sh -f /etc/passwd -n 1
root:x:0:0:root:/root:/bin/bash

```

Q:54 list_sys_status.sh

显示系统使用的以下信息:

主机名、IP地址、子网掩码、网关、DNS服务器IP地址信息

```
#!/bin/bash
IP=`ifconfig eth0 | head -2 | tail -1 | awk '{print $2}' | awk -F":" '{print $2}'`
ZW=`ifconfig eth0 | head -2 | tail -1 | awk '{print $3}' | awk -F":" '{print $2}'`
GW=`route -n | tail -1 | awk '{print $2}'`
HN=`hostname`
DNS=`head -1 /etc/resolv.conf | awk '{print $2}'`
echo '此机IP地址是' $IP
echo '此机子网掩码是' $ZW
echo '此机网关是' $GW
echo '此机主机名是' $HN
echo '此机DNS是' $DNS
```

Q:55 mysqlbak.sh备份数据库目录脚本

```
#!/bin/bash
DAY=`date +%Y%m%d`
SIZE=`du -sh /var/lib/mysql`
echo "Date: $DAY" >> /tmp/dbinfo.txt
echo "Data Size: $SIZE" >> /tmp/dbinfo.txt
cd /opt/dbbak && /dev/null || mkdir /opt/dbbak
tar zcf /opt/dbbak/mysqlbak-${DAY}.tar.gz /var/lib/mysql /tmp/dbinfo.txt &> /dev/null
rm -f /tmp/dbinfo.txt

crontab-e
55 23 */3 * * /opt/dbbak/dbbak.sh
```

Q:56 每周日半夜23点半, 对数据库服务器上的webdb库做完整备份

每备份文件保存到系统的/mysqlbak目录里

用系统日期做备份文件名 webdb-YYYY-mm-dd.sql

每次完整备份后都生成新的binlog日志

把当前所有的binlog日志备份到/mysqlbinlog目录下

```
#mkdir /mysqlbak
#mkdir /mysqlbinlog
#service mysqld start
cd /shell
#vi webdb.sh
#!/bin/bash
day=`date +%F`
mysqldump -hlocalhost -uroot -p123 webdb > /mysqlbak/webdb-${day}.sql
mysql -hlocalhost -uroot -p -e "flush logs"
tar zcf /mysqlbinlog.tar.gz /var/lib/mysql/mysqld-bin.0*
#chmod +x webdb.sh
#crontab -e
30 23 * * 7 /shell/webdb.sh
```

Q:57 very.ser.sh(检查任意一个服务的运行状态)

只检查服务vsftpd httpd sshd crond、mysql中任意一个服务的状态

如果不是这5个中的服务，就提示用户能够检查的服务名并退出脚本

如果服务是运行着的就输出 "服务名 is running"

如果服务没有运行就启动服务

方法1: 使用read写脚本

```
#!/bin/bash
read -p "请输入你的服务名:" service
if [ $service != 'crond' -a $service != 'httpd' -a $service != 'sshd' -a $service != 'mysqld' -a $service != 'vsftpd' ];then
echo "只能够检查'vsftpd,httpd,crond,mysqld,sshd'"
exit 5
fi
service $service status &> /dev/null

if [ $? -eq 0 ];then
echo "服务在线"
else
service $service start
fi
```

方法2: 使用位置变量来写脚本

```
if [ -z $1 ];then
echo "You mast specify a servername!"
echo "Usage: `basename$0` servername"
exit 2
fi
if [ $1 == "crond" ] || [ $1 == "mysql" ] || [ $1 == "sshd" ] || [ $1 == "httpd" ] || [ $1 == "vsftpd" ];then
service $1 status &> /dev/null
if [ $? -eq 0 ];then
echo "$1 is running"
else
service $1 start
fi
```

```
else
echo "Usage:`basename $0` server name"
echo "But only check for vsftpd httpd sshd crond mysqld" && exit2
fi
```

Q:58 pc_noline.sh

输出192.168.1.0/24网段内在线主机的ip地址

统计不在线主机的台数，并把不在线主机的ip地址和不在线时的时间保存到/tmp/ip.txt文件里

```
#!/bin/bash
ip=192.168.1.
j=0
for i in `seq 10 12`
do
ping -c 3 $ip$i &> /dev/null
if [ $? -eq 0 ];then
echo 在线的主机有: $ip$i
else
let j++
echo $ip$i >> /tmp/ip.txt
date >> /tmp/ip.txt
fi
done
echo 不在线的主机台数有 $j
```

Q:59 一个简单的网站论坛测试脚本

用交互式的输入方法实现自动登录论坛数据库，修改用户密码

```
[root@test1 scripts]# vim input.sh

#!/bin/bash

End=ucenter_members
Mysql=/home/lnmp/mysql/bin/mysql

read -p "Enter a website directory : " webdir
webPath=/home/webSer/$webdir/config
echo $webPath

read -p "Enter dbuser name : " dbuser
echo $dbuser

read -sp "Enter dbuser password : " dbpass

read -p "Enter db name : " dbname
echo $dbname

read -p "Enter db tablepre : " dbtablepre
echo $dbtablepre
```



```

Globalphp=`grep "tablepre*" $WebPath/config_global.php |cut -d '"' -f8`
Ucenterphp=`grep "UC_DBTABLEPRE*" $WebPath/config_ucenter.php |cut -d '.' -f2 |
awk -F '"' '{print $1}'`

if [ $dbtablepre == $Globalphp ] && [ $dbtablepre == $Ucenterphp ];then

    Start=$dbtablepre
    Pre=`echo $Start$End`

    read -p "Enter you name : " userset
    echo $userset

    Result=`$Mysql -u$dbuser -p$dbpass $dbname -e "select username from $Pre
where username='$userset'\G"|cut -d ' ' -f2|tail -1`
    echo $Result
    if [ $userset == $Result ];then
        read -p "Enter your password : " userpass
        passnew=`echo -n $userpass|openssl md5|cut -d ' ' -f2`

        $Mysql -u$dbuser -p$dbpass $dbname -e "update $Pre set
password='$passnew' where username='$userset';"
        $Mysql -u$dbuser -p$dbpass $dbname -e "flush privileges;"
    else
        echo "$userset is not right user!"
        exit 1
    fi
else
    exit 2
fi

```

Q:60 slave_status.sh (检查mysql主从结构中从数据库服务器的状态)

- 1) 本机的数据库服务是否正在运行
- 2) 能否与主数据库服务器正常通信
- 3) 能否使用授权用户连接数据库服务器
- 4) 本机的slave_IO进程是否处于YES状态

本机的slave_SQL进程是否处于YES状态

```

[root@test1 scripts]# vim test.sh

#!/bin/bash
netstat -tulnp | grep :3306 > /dev/null
if [ $? -eq 0 ];then
echo "服务正在运行"
else
service mysqld start
fi
ping -c 3 192.168.1.100 &> /dev/null
if [ $? -eq 0 ];then
echo "网络连接正常"
else
echo "网络连接失败"

```

```

fi
mysql -h192.168.1.100 -uroot -p123456 &> /dev/null
if [ $? -eq 0 ];then
echo "数据库连接成功"
else
echo "数据库连接失败"
fi
IO= mysql -uroot -p123 -e "show slave status\G" | grep Slave_IO_Running | awk
'{print $2}' > /dev/null
SQL= mysql -uroot -p123 -e "show slave status\G" | grep Slave_SQL_Running | awk
'{print $2}' /dev/null
if [ IO==Yes ] && [ SQL==Yes ];then
echo "IO and SQL 连接成功"
else
echo "IO线程和SQL线程连接失败"
fi

```

Q:61 轮询检测Apache状态并启用钉钉报警

```

#!/bin/bash

shell_user="root"
shell_domain="apache"

shell_list="/root/ip_list"
shell_row=`cat $shell_list |wc -l`

function trans_text(){
text=$1

curl 'https://oapi.dingtalk.com/robot/send?access_token=b4fcf5862088a1bc7f2bf66a'
-H'Content-Type: application/json' -d'{          #指定钉钉机器人hook地址
    "msgtype": "text",
    "text": {
        "content": "'"$text"'"
    },
}'
}

function apache_check_80(){
ip=$1
URL="http://$ip/index.html"
HTTP_CODE=`curl -o /dev/null -s -w "%{http_code}" "${URL}"`

if [ $HTTP_CODE != 200 ]
then
trans_text "

=====
\n $ip Apache 服务器状态异常，网页返回码:
'"$HTTP_CODE"' 请及时处理 ! \n

```

```

===== \n"
    fi
}

while true
do

shell_list="/root/ip_list"
shell_row=`cat $shell_list |wc -l`
    for temp in `seq 1 $shell_row`
    do
        Ip_Addr=`cat $shell_list |head -n $temp |tail -n 1`
        apache_check_80 $Ip_Addr
    done

    sleep 10
done

```

Q:62 一台监控主机，一台被监控主机。被监控主机分区使用率大于80%，就发告警邮件。放到crontab里面，每10分钟执行一次。

```

#!/bin/bash

FSMAX="80"
remote_user='root'
remote_ip=(IP地址列表)
ip_num='0'

while [ "$ip_num" -le "$(expr ${#remote_ip[@]} - 1)" ]
do
    read_num='1'
    ssh "$remote_user"@${remote_ip[$ip_num]} df -h > /tmp/diskcheck_tmp
    grep '^/dev/*' /tmp/diskcheck_tmp | awk '{print $5}' | sed 's/\%//g' >
/tmp/diskcheck_num_tmp

    while [ "$read_num" -le $(wc -l < /tmp/diskcheck_num_tmp) ]
    do
        size=$(sed -n "$read_num" 'p' /tmp/diskcheck_num_tmp)
        if [ "size" -gt "$FSMAX" ]
        then
            $(grep '^/dev/*' /tmp/diskcheck_tmp | sed -n $read_num 'p'
> /tmp/disk_check_mail)
            $(echo ${remote_ip[$ip_num]}) >> /tmp/disk_check_mail)
            $(mail -s "diskcheck_alert" admin <
/tmp/disk_check_mail)
        fi

        read_num=$(expr $read_num + 1)
    done

    ip_num=$(expr $ip_num + 1)
done

```

Q:63 监控主机的磁盘空间,当使用空间超过90%就通过发mail来发警告

```
#!/bin/bash
#monitor available disk space
#提取本服务器的IP地址信息
IP=`ifconfig eth0 | grep "inet addr" | cut -f 2 -d ":" | cut -f 1 -d " "`
SPACE=`df -hP | awk '{print int($5)}'`
if [ $SPACE -ge 90 ]
then
    echo "$IP 服务器 磁盘空间 使用率已经超过90%，请及时处理。"|mail -s "$IP 服务器硬盘告警，
    公众号：网络技术干货圈" fty89@163.com
fi
```

Q:64 自动ftp上传

```
#!/bin/bash

ftp -n << END_FTP
open 192.168.1.22
user test testing //用户名test 密码: testing
binary
prompt off //关闭提示
mput files //上传files文件
close
bye
END_FTP
```

Q:65.mysqlbak.sh备份数据库目录脚本

```
#!/bin/bash

DAY=`date +%Y%m%d`
SIZE=`du -sh /var/lib/mysql`
echo "Date: $DAY" >> /tmp/dbinfo.txt
echo "Data Size: $SIZE" >> /tmp/dbinfo.txt
cd /opt/dbbak && /dev/null || mkdir /opt/dbbak
tar zcf /opt/dbbak/mysqlbak-${DAY}.tar.gz /var/lib/mysql /tmp/dbinfo.txt &>
/dev/null
rm -f /tmp/dbinfo.txt

crontab-e
55 23 */3 * * /opt/dbbak/dbbak.sh
```

Q:66.打印彩虹

```
declare -a ary

for i in `seq 40 49`
do

    ary[$i]=" "
```

```

        echo -en "\e[$i;5m ${ary[@]}\e[;0m"

done

declare -a ary
for s in `seq 1 10000`
do
    for i in `seq 40 49`
    do
        ary[$i]=" "
        echo -en "\e[$i;5m ${ary[@]}\e[;0m"
    done
done
done

```

Q:67.打印菱形

```

#!/bin/bash

for (( i = 1; i < 12; i++))
do
    if [[ $i -le 6 ]]
    then
        for ((j = ((12-i)); j > i; j--))
        do
            echo -n " "
        done

        for ((m = 1; m <= ((2*i-1)); m++))
        do
            echo -n "*"
        done
        echo ""

#*****
    elif [[ $i -gt 6 ]]
    then
        n=$((12-i))
        for ((j = ((12-n)); j > n; j--))
        do
            echo -n " "
        done

        for ((m = 1; m <= ((2*n-1)); m++))
        do
            echo -n "*"
        done
        echo ""
    fi
done

```

Q:68.expect实现远程登陆自动交互

```
#!/usr/bin/expect -f

set ipaddress [lindex $argv 0]

set passwd [lindex $argv 1]

set timeout 30

spawn ssh-copy-id root@$ipaddress

expect {

    "yes/no" { send "yes\r";exp_continue }

    "password:" { send "$passwd\r" }

}

#expect "*from*"

#send "mkdir -p ./tmp/testfile\r"

#send "exit\r"

#expect "#" #i# 命令运行完，你要期待一个结果，结果就是返回shell提示符了(是# 或者$)
```

Q:69.http心跳检测

```
URL="http://192.168.22.191/index.html"

THHP_CODE=`curl -o /dev/null -s -w "%{http_code}" "${URL}"`

if [ $HTTP_CODE != 200 ]
then
    echo -e "apache code:"$HTTP_CODE""
fi
```

Q:70.PV过量自动实现防火墙封IP

```
#!/bin/bash

log=/tmp/tmp.log

[ -f $log ] || touch $log

function add_iptales()
{
    while read line
    do
        ip=`echo $line |awk '{print $2}'`
        count=`echo $line |awk '{print $1}'`
```

```

        if [ $count -gt 100 ] && [ `iptables -L -n |grep "$ip"
|wc -l` -lt 1 ]
        then
            iptables -I INPUT -s $ip -j DROP
            echo -e "$list
isdropped">>/tmp/droplist.log
        fi

        done<$log
    }

function main()
{
    while true
    do
        netstat -an|grep "EST" |awk -F '[:]+' '{print $6}'|sort |uniq -c
>$log
        add_iptales
        sleep 180
    done
}

main

```

Q:71.shell实现自动安装

```

#!/bin/bash

function MyInstall
{
    if ! rpm -qa |grep -q "^$1"
    then

        yum install $1
        if [ $? -eq 0 ]
        then
            echo -e "$i install is ok\n"
        else
            echo -e "$1 install no\n"
        fi
    else
        echo -e "yi an zhuang ! \n"
    fi
}

for ins in mysql php httpd
do
    MyInstall $ins
done

```

Q:72.shell实现插入排序

```
#!/bin/bash

declare -a array

for i in `seq 1 10`
do
    array[$i]=$RANDOM
done

echo -e "Array_1:  ${array[@]}"

for (( x=1;x<=9;x++ ))
do
    for(( y=1;y<=9;y++ ))
    do
        if [ ${array[$y]} -gt ${array[$y+1]} ]
        then
            temp=${array[$y]}
            array[$y]=${array[$y+1]}
            array[$y+1]=$temp
        fi
    done
done

echo -e "Array_2:  ${array[@]}"
```

Q:73.bash实现动态进度条

```
#!/bin/bash
i=0
bar=''
index=0
arr=( "|" "/" "-" "\\" )

while [ $i -le 100 ]
do
    let index=index%4
    printf "[%s][%d%][\e[43;46;1m%c\e[0m\r" "$bar" "$i" "${arr[$index]}"
    let i++
    let index++
    usleep 30000
    bar+='#'
    clear
done

printf "\n"
```


Q:74. 根据文件内容创建账号

```
#!/bin/bash

for Uname in `cat /root/useradd.txt |gawk '{print $1}'`
do

    id $Uname &> /dev/null
    if [ $? -eq 0 ]
    then
        echo -e "这个账号已存在!来源: 微信公众号【网络技术干货圈】"
        continue
    fi
    for Upasswd in `cat /root/useradd.txt |gawk '{print $2}'`
    do
        useradd $Uname &> /dev/null
        echo "$Upasswd" |passwd --stdin $Uname &> /dev/null
        if [ $? -eq 0 ]
        then
            echo -e "账号创建成功!"
        else
            echo -e "创建失败!"
        fi
    done
done
```

Q:75. 红色进度条

```
#!/bin/bash

declare -a ary

for i in `seq 0 20`
do

    ary[$i]=" "
    echo -en "\e[41;5m ${ary[@]}\e[;0m"
    sleep 1
done
```

Q:76. 监控服务器网卡流量

```
#!/bin/bash
#network
#Mike.Xu
while : ; do
speedtime='date +%m"- "%d" "%k": "%M'
speedday='date +%m"- "%d'
```

```

speedrx_before='ifconfig eth0|sed -n "8"p|awk '{print $2}'|cut -c7-'
speedtx_before='ifconfig eth0|sed -n "8"p|awk '{print $6}'|cut -c7-'
sleep 2
speedrx_after='ifconfig eth0|sed -n "8"p|awk '{print $2}'|cut -c7-'
speedtx_after='ifconfig eth0|sed -n "8"p|awk '{print $6}'|cut -c7-'
speedrx_result=$((speedrx_after-speedrx_before)/256]
speedtx_result=$((speedtx_after-speedtx_before)/256]
echo"$speedday$speedtime Now_In_Speed: "$speedrx_result"kbps Now_Out_Speed:
"$speedtx_result"kbps"
sleep 2
done

```

Q:77. 检测CPU剩余百分比

```

#!/bin/bash

#Inspect CPU

#Sun Jul 31 17:25:41 CST 2016

PATH=/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/home/wl/bin
export PATH

TERM=linux
export TERM

CpuResult=$(top -bn 1 | grep "Cpu" | awk '{print $5}' | sed 's/\..*$/g')

if [[ $CpuResult < 20 ]];then
    echo "CPU WARNING : $CpuResult" > /service/script/.cpu_in.txt
    top -bn 1 >> /service/script/.cpu_in.txt
    mail -s "Inspcet CPU" wl < /service/script/.cpu_in.txt
fi

```

Q:78.检测磁盘剩余空间

```

#!/bin/bash

#Insepct Harddisk , If the remaining space is more than 80%, the message is sent
to the wl

#Tue Aug 2 09:45:56 CST 2016

PATH=/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/home/wl/bin
export PATH

for RemainingSpace in $(df -h | awk '{print $5}' | grep -v 'Use' | sed -e
's/[%]//g')
do
    if [[ $RemainingSpace > 80 ]];then
        echo -e "$RemainingSpace"
        echo -e "$(df -h | grep $RemainingSpace)" > /service/script/.Harddiskwarning
        mail -s "disk warning" wl < /service/script/.Harddiskwarning
    fi
done

```

Q:79. bash-实现检测apache状态并钉钉报警

```
#!/bin/bash

function trans_text(){
    text=$1
    curl 'https://oapi.dingtalk.com/robot/send?
access_token=b4fcf5862088a1bc7f2bf66aea051869e62ff5879fa0e0fddb0db9b1494781c2' -
H'Content-Type: application/json' -d'
{
    "msgtype": "text",
    "text": {
        "content": ""$text""
    },
}'
}

function desk_check(){

    dftype=$1
    shell_row=`df |wc -l`

    for i in `seq 2 $shell_row`
    do

        temp=(`df -h |head -n $i |tail -n 1 |awk '{print $5 "\t" $6}'`)
        disk=`echo ${temp[0]} |cut -d "%" -f 1`
        name="${temp[1]}"
        hostname=`hostname`
        IP=`ifconfig |grep -v "127.0.0.1" |grep "inet addr:" |sed 's/^.*inet
addr://g'|sed 's/ Bcas..*$/g`
        #echo -e "$disk      $name"
        Dat=`date "+%F %T"`

        if [ $disk -ge $dftype ]
        then
            echo "
                                ===== \n
                                >磁盘分区异常< \n
                                主机名: $hostname \n
                                IP地址: $IP \n
                                分区名: $name \n
                                使用率: $disk %\n
                                发生时间: $Dat \n
                                ===== \n"
        fi
    done
}

function apache_check(){
    url=$1
    URL="http://$url/"
    HTTP_CODE=`curl -o /dev/null -s -w "%{http_code}" "${URL}"`
}
```

```

if [ $HTTP_CODE != 200 ]
then
    echo "
                                ===== \n
                                >Apache服务异常<
                                主机名: $hostname \n
                                IP地址: $IP \n
                                返回代码: $HTTP_CODE \n
                                发生时间: $Dat \n
                                ===== \n"

fi
}

while true
do
    desk_check 10
    apache_check 127.0.0.1

    sleep 10
done

```

Q:80.内存检测

```

#!/bin/bash

#Inspect Memory : If the memory is less than 500 , then send mail to wl

#Tue Aug  2 09:13:43 CST 2016

PATH=/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/home/wl/bin

export PATH

MEM=$(free -m | grep "Mem" | awk '{print $4}')

if [[ MEM < 500 ]];then
    echo -e "Memory Warning : Memory free $MEM" > /service/script/.MemoryWarning
    mail -s "Memory warning" wl < /service/script/.MemoryWarning
fi

```

Q:81.剩余inode检测

```

#!/bin/bash

#Inspcet Inode : If the free INODE is less than 200, the message is sent to the
wl

#Tue Aug  2 10:21:29 CST 2016

PATH=/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/home/wl/bin

export PATH

for FreeInode in $(df -i | grep -v "Filesystem" | awk '{print $4}')

```

```
do
    if [[ $FreeInode < 200 ]];then
        echo -e "$(df -i | grep "$FreeInode")" > /service/script/.FreeInode
        mail -s "FreeInode warning" wl < /service/script/.FreeInode
    fi
done
```

Q:82.判断哪些用户登陆了系统

```
#!/bin/bash

declare -i count=0

while true;do

    if who |grep -q -E "^wang"
    then
        echo -e "用户wang 登陆了系统\n 这是第$count 次!威信公众浩: wljsghq"
        break
    else
        let count++
    fi

    sleep 3
done
~
```

示例：找出UID为偶数的所有用户，显示其用户名和ID号：

```
#!/bin/bash
while read line; do
    userid=$(echo $line | cut -d: -f3)
    if [ ${userid%2} -eq 0 ]; then
echo $line | cut -d: -f1,3
    fi
done < /etc/passwd
```

Q:83.批量创建账号

```
#!/bin/bash

sum=1

while [ $sum -le 30 ]
do
    if [ $sum -le 9 ]
    then
        user="user_0$sum"
    else
        user="user_$sum"
    fi

    useradd $user
    echo "123456" |passwd --stdin $user
```

```
chage -d 0 $user
let sum=sum+1
```

done

Q:84.批量扫面存活

```
#!/bin/bash
#By:lyshark

#nmap 192.168.22.0/24>ip

MAC=`cat ip |awk '$1 == "MAC" && $NF == "(VMware)" {print $3}'`

for i in `seq 1 20`
do

temp=`echo ${MAC[@]} |awk '{print $i}'`

IP=`cat /ip |grep -B5 $temp |grep "Nmap scan"|awk '{print $5}'`

    echo $IP |awk '{print $1}'
done
```

Q:85.正则匹配IP

```
^[0-9]{0,2}|^1[0-9]{0,2}|^2[0-5]{0,2}

egrep "(^([0-9]{1,2}|1[0-9]{0,2}|2[0-5]{0,2})\.([0-9]{1,2}|1[0-9]{0,2}|2[0-5]{0,2})\.([0-9]{1,2}|1[0-9]{0,2}|2[0-5]{0,2})\.([0-9]{1,2}|1[0-9]{0,2}|2[0-5]{0,2}))$"

([0-9]{1,2}|1[0-9]{0,2}|2[0-5]{0,2})
([0-9]{1,2}|1[0-9]{0,2}|2[0-5]{0,2})
([0-9]{1,2}|1[0-9]{0,2}|2[0-5]{0,2})
([0-9]{1,2}|1[0-9]{0,2}|2[0-5]{0,2})

egrep "((25[0-5]|2[0-4][0-9]|((1[0-9]{2})|([1-9]?[0-9])))\.){3}(25[0-5]|2[0-4][0-9]|((1[0-9]{2})|([1-9]?[0-9])))"
```

```
ls |egrep "((25[0-5]|2[0-4][0-9]|((1[0-9]{2})|([1-9]?[0-9])))\.){3}(25[0-5]|2[0-4][0-9]|((1[0-9]{2})|([1-9]?[0-9])))"
```

Q:86.正则匹配邮箱

```
egrep "^([0-9a-zA-Z][0-9a-zA-Z_]{1,16}[0-9a-zA-Z]\@[0-9a-zA-Z-]*([0-9a-zA-Z])?)\.(com|com.cn|net|org|cn)$" rui
```

```
ls |egrep "^((([1-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-4]))$"
```

Q:87.实现布片效果

```
#!/bin/bash

function ary_go
{
    $1 $2

    for (( i=0;i<=$1;i++ ))
    do
        for (( s=0;s<=$2;s++ ))
        do
            if [ ${i%2} == 0 ]
            then

                if [ ${s%2} == 0 ]
                then
                    echo -en "  "
                else
                    echo -en "\e[;44m  \e[;m"
                fi
            else

                if [ ${s%2} == 0 ]
                then
                    echo -en "\e[;42m  \e[;m"
                else
                    echo -en "  "
                fi
            fi

        done
        echo

    done

}

ary_go 25 50
```

Q:88.剔除白名单以外的用户

```
#!/bin/bash
w | awk 'NR>=3 {printf $1 "\t" $2 "\t" $3 "\n"}' > /tmp/who.txt
for i in $(awk '{printf $1}' /tmp/bai.txt)
do
    k=$(egrep -v "$i" /tmp/who.txt | awk '{printf $2} "\n"' | awk '{printf $2 "\n"}')
    for j in $k
    do
        pkill -9 -t "$j"
    done
done
```

61-88题

来源: <https://www.cnblogs.com/LyShark/p/9117041.html>

作者: LyShark

Q:89.一键安装 MongoDB 数据库脚本

```
#!/bin/bash
#
#*****
#Author:      wangxiaochun
#QQ:          29308620
#Date:        2021-02-19
#FileName:     install_mongodb.sh
#URL:          http://www.wangxiaochun.com
#Description:   The test script
#Copyright (C): 2021 All rights reserved
#*****
file=mongodb-linux-x86_64-ubuntu1804-4.4.4.tgz
url=https://fastdl.mongodb.org/linux/$file
db_dir=/data/db
install_dir=/usr/local
port=27017

color () {
    RES_COL=60
    MOVE_TO_COL="echo -en \033[${RES_COL}G"
    SETCOLOR_SUCCESS="echo -en \033[1;32m"
    SETCOLOR_FAILURE="echo -en \033[1;31m"
    SETCOLOR_WARNING="echo -en \033[1;33m"
    SETCOLOR_NORMAL="echo -en \E[0m"
    echo -n "$2" && $MOVE_TO_COL
    echo -n "["
    if [ $1 = "success" -o $1 = "0" ] ;then
        ${SETCOLOR_SUCCESS}
        echo -n $" OK "
    elif [ $1 = "failure" -o $1 = "1" ] ;then
        ${SETCOLOR_FAILURE}
        echo -n $"FAILED"
    else
        ${SETCOLOR_WARNING}
        echo -n $"WARNING"
```



```

    fi
    ${SETCOLOR_NORMAL}
    echo -n "]"
    echo

}

os_type () {
    awk -F'[ "]"' '/^NAME/{print $2}' /etc/os-release
}

check () {
    [ -e $db_dir -o -e $install_dir/mongodb ] && { color 1 "MongoDB 数据库已安
装";exit; }
    if [ `os_type` = "CentOS" ];then
        rpm -q curl &> /dev/null || yum install -y -q curl
    elif [ `os_type` = "Ubuntu" ];then
        dpkg -l curl &> /dev/null || apt -y install curl
    else
        color 1 不支持当前操作系统
        exit
    fi
}

file_prepare () {
    if [ ! -e $file ];then
        curl -O $url || { color 1 "MongoDB 数据库文件下载失败"; exit; }
    fi
}

install_mongodb () {
    tar xf $file -C $install_dir
    mkdir -p $db_dir
    ln -s $install_dir/mongodb-linux-x86_64-* $install_dir/mongodb
    echo PATH=$install_dir/mongodb/bin/:'$PATH' > /etc/profile.d/mongodb.sh
    . /etc/profile.d/mongodb.sh
    mongod --dbpath $db_dir --bind_ip_all --port $port --logpath
$db_dir/mongod.log --fork
    [ $? -eq 0 ] && color 0 "MongoDB 数据库安装成功!" || color 1 "MongoDB 数据库安装
失败!"
}

check
file_prepare
install_mongodb

```

Q:90 使用mobaXtrem显示CentOS 上的图形工具

```

[root@centos ~]#yum install xorg-x11-xauth xorg-x11-fonts-* xorg-x11-font-utils
xorg-x11-fonts-Type1 firefox
[root@centos ~]#exit

```

Q:91 检测两台服务器指定目录下的文件一致性

```
#!/bin/bash
#####
检测两台服务器指定目录下的文件一致性
#####
#通过对比两台服务器上文件的md5值，达到检测一致性的目的
dir=/data/web
b_ip=192.168.88.10
#将指定目录下的文件全部遍历出来并作为md5sum命令的参数，进而得到所有文件的md5值，并写入到指定文件中
find $dir -type f|xargs md5sum > /tmp/md5_a.txt
ssh $b_ip "find $dir -type f|xargs md5sum > /tmp/md5_b.txt"
scp $b_ip:/tmp/md5_b.txt /tmp
#将文件名作为遍历对象进行一一比对
for f in `awk '{print 2} /tmp/md5_a.txt'`do
#以a机器为标准，当b机器不存在遍历对象中的文件时直接输出不存在的结果
if grep -qw "$f" /tmp/md5_b.txt
then
md5_a=`grep -w "$f" /tmp/md5_a.txt|awk '{print 1}'`
md5_b=`grep -w "$f" /tmp/md5_b.txt|awk '{print 1}'`
#当文件存在时，如果md5值不一致则输出文件改变的结果
if [ $md5_a != $md5_b ]then
echo "$f changed."
fi
else
echo "$f deleted."
fi
done
```

Q:92定时清空文件内容，定时记录文件大小

```
#!/bin/bash
#####
每小时执行一次脚本（任务计划），当时间为0点或12点时，将目标目录下的所有文件内容清空，但不删除文件，其他时间则只统计各个文件的大小，一个文件一行，输出到以时#间和日期命名的文件中，需要考虑目标目录下二级、三级等子目录的文件
#####
logfile=/tmp/`date +%H-%F`.log
n=`date +%H`
if [ $n -eq 00 ] || [ $n -eq 12 ]
then
#通过for循环，以find命令作为遍历条件，将目标目录下的所有文件进行遍历并做相应操作
for i in `find /data/log/ -type f`
do
true > $i
done
else
for i in `find /data/log/ -type f`
do
du -sh $i >> $logfile
done
fi
```

Q:93检测网卡流量，并按规定格式记录在日志中

```
#!/bin/bash
#####
#检测网卡流量，并按规定格式记录在日志中#规定一分钟记录一次
#日志格式如下所示：
#2019-08-12 20:40
#ens33 input: 1234bps
#ens33 output: 1235bps
#####3
while :
do
#设置语言为英文，保障输出结果是英文，否则会出现bug
LANG=en
logfile=/tmp/`date +%d`.log
#将下面执行的命令结果输出重定向到logfile日志中
exec >> $logfile
date +"%F %H:%M"
#sar命令统计的流量单位为kb/s，日志格式为bps，因此要*1000*8
sar -n DEV 1 59|grep Average|grep ens33|awk '{print
$2,"\t","input:","\t",$5*1000*8,"bps","\n",$2,"\t","output:","\t",$6*1000*8,"bps
"}'
echo "#####"
#因为执行sar命令需要59秒，因此不需要sleep
done
```

Q:94 计算文档每行出现的数字个数，并计算整个文档的数字总数

```
#!/bin/bash
#####
#计算文档每行出现的数字个数，并计算整个文档的数字总数
#####
#使用awk只输出文档行数（截取第一段）
n=`wc -l a.txt|awk '{print $1}'`
sum=0
#文档中每一行可能存在空格，因此不能直接用文档内容进行遍历
for i in `seq 1 $n`do
#输出的行用变量表示时，需要用双引号
line=`sed -n "$i"p a.txt`#wc -L选项，统计最长行的长度
n_n=`echo $line|sed s'/[^\0-9]//g'|wc -L`
echo $n_nsum=$((sum+$n_n))
done
echo "sum:$sum"
```

杀死所有脚本

```
#!/bin/bash
#####
#有一些脚本加入到了cron之中，存在脚本尚未运行完毕又有新任务需要执行的情况，
#导致系统负载升高，因此可通过编写脚本，筛选出影响负载的进程一次性全部杀死。
#####
ps aux|grep 指定进程名|grep -v grep|awk '{print $2}'|xargs kill -9
```

Q:95 从 FTP 服务器下载文件

```
#!/bin/bash
if [ $# -ne 1 ]; then
    echo "Usage: $0 filename"
fi
dir=$(dirname $1)
file=$(basename $1)
ftp -n -v << EOF    # -n 自动登录
open 192.168.1.10    # ftp服务器
user admin password
binary              # 设置ftp传输模式为二进制，避免MD5值不同或.tar.gz压缩包格式错误
cd $dir
get "$file"
EOF
```

Q:96、连续输入5个100以内的数字，统计和、最小和最大

```
#!/bin/bash
COUNT=1
SUM=0
MIN=0
MAX=100
while [ $COUNT -le 5 ]; do
    read -p "请输入1-10个整数: " INT
    if [[ ! $INT =~ ^[0-9]+$ ]]; then
        echo "输入必须是整数! "
        exit 1
    elif [[ $INT -gt 100 ]]; then
        echo "输入必须是100以内! "
        exit 1
    fi
    SUM=$((SUM+$INT))
    [ $MIN -lt $INT ] && MIN=$INT
    [ $MAX -gt $INT ] && MAX=$INT
    let COUNT++
done
echo "SUM: $SUM"
echo "MIN: $MIN"
echo "MAX: $MAX"
```

用户猜数字

```
#!/bin/bash # 脚本生成一个 100 以内的随机数,提示用户猜数字,根据用户的输入,提示用户猜对了,
# 猜小了或猜大了,直至用户猜对脚本结束。
# RANDOM 为系统自带的系统变量,值为 0-32767的随机数
# 使用取余算法将随机数变为 1-100 的随机数num=$((RANDOM%100+1))echo "$num"
# 使用 read 提示用户猜数字
# 使用 if 判断用户猜数字的大小关系:-eq(等于),-ne(不等于),-gt(大于),-ge(大于等于),
# -lt(小于),-le(小于等于)

while :
do
    read -p "计算机生成了一个 1-100 的随机数,你猜: " cai
    if [ $cai -eq $num ]
```

```

then
    echo "恭喜,猜对了"
    exit
elif [ $cai -gt $num ]
then
    echo "Oops,猜大了"
else
    echo "Oops,猜小了"
fi
done

```

Q:97、监测 Nginx 访问日志 502 情况，并做相应动作

假设服务器环境为 lnmp，近期访问经常出现 502 现象，且 502 错误在重启 php-fpm 服务后消失，因此需要编写监控脚本，一旦出现 502，则自动重启 php-fpm 服务。

```

#场景：
#1. 访问日志文件的路径： /data/log/access.log
#2. 脚本死循环，每10秒检测一次，10秒的日志条数为300条，出现502的比例不低于10%（30条）则需要重启php-fpm服务
#3. 重启命令为： /etc/init.d/php-fpm restart
#!/bin/bash
#####
#监测Nginx访问日志502情况，并做相应动作
#####
log=/data/log/access.log
N=30 #设定阈值
while :do
    #查看访问日志的最新300条，并统计502的次数
    err=`tail -n 300 $log |grep -c '502' ``
    if [ $err -ge $N ]
    then
        /etc/init.d/php-fpm restart 2> /dev/null
        #设定60s延迟防止脚本bug导致无限重启php-fpm服务
        sleep 60
    fi
    sleep 10
done

```

Q:98、将结果分别赋值给变量

应用场景：希望将执行结果或者位置参数赋值给变量，以便后续使用。

方法1：

```

for i in $(echo "4 5 6"); do
    eval a$i=$i
done
echo $a4 $a5 $a6

```

方法2：将位置参数192.168.1.1{1,2}拆分为到每个变量

```

num=0
for i in $(eval echo $*);do    #eval将{1,2}分解为1 2
    let num+=1
    eval node${num}="$i"
done
echo $node1 $node2 $node3
# bash a.sh 192.168.1.1{1,2}
192.168.1.11 192.168.1.12

```

```

方法3: arr=(4 5 6)
INDEX1=$(echo ${arr[0]})
INDEX2=$(echo ${arr[1]})
INDEX3=$(echo ${arr[2]})

```

Q:99、批量修改文件名

示例:

```

# touch article_{1..3}.html
# lsarticle_1.html article_2.html article_3.html
目的: 把article改为bbs

```

方法1:

```

for file in $(ls *.html); do
    mv $file bbs_${file#*_}
    # mv $file $(echo $file |sed -r 's/.*(._.)/bbs\1/')
    # mv $file $(echo $file |echo bbs_$(cut -d_ -f2))
done

```

方法2:

```

for file in $(find . -maxdepth 1 -name "*.html"); do
    mv $file bbs_${file#*_}done

```

方法3:

```

# rename article bbs *.html
把一个文档前五行中包含字母的行删掉, 同时删除6到10行包含的所有字母

```

1) 准备测试文件, 文件名为2.txt

```

第1行1234567不包含字母
第2行56789BBBBBB
第3行67890CCCCCCCC
第4行78asfDDDDDDDD
第5行123456EEEEEEEE
第6行1234567ASDF
第7行56789ASDF
第8行67890ASDF
第9行78asfADSF
第10行123456AAAA
第11行67890ASDF
第12行78asfADSF
第13行123456AAAA

```

2) 脚本如下:

```
#!/bin/bash
#####
把一个文档前五行中包含字母的行删掉，同时删除6到10行包含的所有字母
#####
sed -n '1,5'p 2.txt |sed '/[a-zA-Z]/'d
sed -n '6,10'p 2.txt |sed s'/[a-zA-Z]//'g
sed -n '11,$'p 2.txt
#最终结果只是在屏幕上打印结果，如果想直接更改文件，可将输出结果写入临时文件中，再替换2.txt或者使用-i选项
```

Q:100、统计当前目录中以.html结尾的文件总大小

方法1:

```
# find . -name "*.html" -exec du -k {} \; |awk '{sum+=$1}END{print sum}'
```

方法2:

```
```bash
for size in $(ls -l *.html |awk '{print $5}'); do
 sum=$((sum+$size))
done
echo $sum
```

## Q:101、扫描主机端口状态

```
#!/bin/bash
HOST=$1
PORT="22 25 80 8080"
for PORT in $PORT; do
 if echo &>/dev/null > /dev/tcp/$HOST/$PORT; then
 echo "$PORT open"
 else
 echo "$PORT close"
 fi
done
用 shell 打印示例语句中字母数小于6的单词

#示例语句:
#Bash also interprets a number of multi-character options.
#!/bin/bash
#####
#shell打印示例语句中字母数小于6的单词
#####
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
```

## Q:102、输入数字运行相应命令

```
#!/bin/bash
#####
#输入数字运行相应命令
#####
echo "*cmd menu* 1-date 2-ls 3-who 4-pwd 0-exit "
while :
do
#捕获用户键入值
read -p "please input number :" n
n1=`echo $n|sed s'/[0-9]//'g`
#空输入检测
if [-z "$n"]
then
continue
fi
#非数字输入检测
if [-n "$n1"]
then
exit 0
fi
break
done
case $n in
1)
date
;;
2)
ls
;;
3)
who
;;
4)
pwd
;;
0)
break
;;
*)
#输入数字非1-4的提示
echo "please input number is [1-4]"
esac
```

## Q:103、Expect 实现 SSH 免交互执行命令

Expect是一个自动交互式应用程序的工具，如telnet，ftp，passwd等。

需先安装expect软件包。

方法1：EOF标准输出作为expect标准输入



```
#!/bin/bash
USER=root
PASS=123.com
IP=192.168.1.120
expect << EOFset timeout 30spawn ssh $USER@$IP expect { "(yes/no)" {send
"yes\r"; exp_continue} "password:" {send "$PASS\r"}}
}
expect "$USER@*" {send "$1\r"}
expect "$USER@*" {send "exit\r"}
expect eof
EOF
```

方法2:

```
#!/bin/bash
USER=root
PASS=123.com
IP=192.168.1.120
expect -c "
 spawn ssh $USER@$IP
 expect {
 \"(yes/no)\" {send \"yes\r\"; exp_continue}
 \"password:\" {send \"$PASS\r\"; exp_continue}
 \"$USER@*\" {send \"df -h\r exit\r\"; exp_continue}
 }"
```

方法3: 将expect脚本独立出来

```
登录脚本:
cat login.exp
#!/usr/bin/expect
set ip [lindex $argv 0]
set user [lindex $argv 1]
set passwd [lindex $argv 2]
set cmd [lindex $argv 3]
if { $argc != 4 } {
 puts "Usage: expect login.exp ip user passwd"
 exit 1
}
set timeout 30
spawn ssh $user@$ip
expect {
 "(yes/no)" {send "yes\r"; exp_continue}
 "password:" {send "$passwd\r"}
}
expect "$user@*" {send "$cmd\r"}
expect "$user@*" {send "exit\r"}
expect eof
```

执行命令脚本: 写个循环可以批量操作多台服务器

```
#!/bin/bash
HOST_INFO=user_info.txt
for ip in $(awk '{print $1}' $HOST_INFO)
do
```

```

user=$(awk -v I="$ip" 'I==$1{print $2}' $HOST_INFO)
pass=$(awk -v I="$ip" 'I==$1{print $3}' $HOST_INFO)
expect login.exp $ip $user $pass $1
done
Linux主机SSH连接信息:
cat user_info.txt
192.168.1.120 root 123456
创建10个用户，并分别设置密码，密码要求10位且包含大小写字母以及数字，最后需要把每个用户的密码存在指定文件中
```bash
#!/bin/bash
#####
#创建10个用户，并分别设置密码，密码要求10位且包含大小写字母以及数字
#最后需要把每个用户的密码存在指定文件中#前提条件：安装mkpasswd命令
#####
#生成10个用户的序列（00-09）
for u in `seq -w 0 09`do
#创建用户
useradd user_$u
#生成密码
p=`mkpasswd -s 0 -l 10`
#从标准输入中读取密码进行修改（不安全）
echo $p|passwd --stdin user_$u
#常规修改密码
echo -e "$p\n$p"|passwd user_$u
#将创建的用户及对应的密码记录到日志文件中
echo "user_$u $p" >> /tmp/userpassworddone

```

Q:104、监控 httpd 的进程数，根据监控情况做相应处理

```

#!/bin/bash
#####
#####
#需求:
#1.每隔10s监控httpd的进程数，若进程数大于等于500，则自动重启Apache服务，并检测服务是否重启成功
#2.若未成功则需要再次启动，若重启5次依旧没有成功，则向管理员发送告警邮件，并退出检测
#3.如果启动成功，则等待1分钟后再次检测httpd进程数，若进程数正常，则恢复正常检测（10s一次），否则放弃重启并向管理员发送告警邮件，并退出检测
#####
#####
#计数器函数
check_service()
{
j=0
for i in `seq 1 5`
do
#重启Apache的命令
/usr/local/apache2/bin/apachectl restart 2> /var/log/httpderr.log
#判断服务是否重启成功
if [ $? -eq 0 ] then
break
else
j=$((j+1)) fi
#判断服务是否已尝试重启5次
if [ $j -eq 5 ] then
mail.py exit

```

```

fi
done }while :do
n=`pgrep -l httpd|wc -l`
#判断httpd服务进程数是否超过500
if [ $n -gt 500 ] then
/usr/local/apache2/bin/apachectl restart
if [ $? -ne 0 ]
then
check_service
else
sleep 60
n2=`pgrep -l httpd|wc -l`
#判断重启后是否依旧超过500
if [ $n2 -gt 500 ]
then
mail.py exit
fi
fi
fi
#每隔10s检测一次
sleep 10done

```

Q:105、批量修改服务器用户密码

Linux主机SSH连接信息：旧密码

```

# cat old_pass.txt
192.168.18.217 root 123456 22
192.168.18.218 root 123456 22
内容格式：IP User Password Port

```

SSH远程修改密码脚本：新密码随机生成

<https://www.linuxprobe.com/books>

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

```
OLD_INFO=old_pass.txt
```

```
NEW_INFO=new_pass.txt
```

```

for IP in $(awk '/^[^#]/{print $1}' $OLD_INFO); do
    USER=$(awk -v I=$IP 'I==$1{print $2}' $OLD_INFO)
    PASS=$(awk -v I=$IP 'I==$1{print $3}' $OLD_INFO)
    PORT=$(awk -v I=$IP 'I==$1{print $4}' $OLD_INFO)
    NEW_PASS=$(mkpasswd -l 8) # 随机密码
    echo "$IP $USER $NEW_PASS $PORT" >> $NEW_INFO
    expect -c "
    spawn ssh -p$PORT $USER@$IP
    set timeout 2
    expect {
        \"(yes/no)\" {send \"yes\r\";exp_continue}
        \"password:\" {send \"$PASS\r\";exp_continue}
        \"$USER@*\" {send \"echo '\$NEW_PASS' |passwd --stdin $USER\r
exit\r\";exp_continue}
    }"
done

```

生成新密码文件：

```

# cat new_pass.txt
192.168.18.217 root n8wX3mU% 22
192.168.18.218 root c87;ZnnL 22

```

Q:106、iptables 自动屏蔽访问网站频繁的IP

场景：恶意访问,安全防范

1) 屏蔽每分钟访问超过200的IP

方法1：根据访问日志（Nginx为例）

```
#!/bin/bash
DATE=$(date +%d/%b/%Y:%H:%M)
ABNORMAL_IP=$(tail -n5000 access.log |grep $DATE |awk '{a[$1]++}END{for(i in a)if(a[i]>100)print i}')
#先tail防止文件过大，读取慢，数字可调整每分钟最大的访问量。awk不能直接过滤日志，因为包含特殊字符。
for IP in $ABNORMAL_IP; do
    if [ $(iptables -vnL |grep -c "$IP") -eq 0 ]; then
        iptables -I INPUT -s $IP -j DROP    fidone
    fi
done
```

方法2：通过TCP建立的连接

```
#!/bin/bash
ABNORMAL_IP=$(netstat -an |awk '$4~/:80$/ && $6~/ESTABLISHED/{gsub(/:[0-9]+/, "", $5);{a[$5]++}}END{for(i in a)if(a[i]>100)print i}')
#gsub是将第五列（客户端IP）的冒号和端口去掉
for IP in $ABNORMAL_IP; do
    if [ $(iptables -vnL |grep -c "$IP") -eq 0 ]; then
        iptables -I INPUT -s $IP -j DROP
    fi
done
```

2) 屏蔽每分钟SSH尝试登录超过10次的IP

方法1：通过lastb获取登录状态:

```
#!/bin/bash
DATE=$(date +"%a %b %e %H:%M") #星期月天时分 %e单数字时显示7，而%d显示07
ABNORMAL_IP=$(lastb |grep "$DATE" |awk '{a[$3]++}END{for(i in a)if(a[i]>10)print i}')
for IP in $ABNORMAL_IP; do
    if [ $(iptables -vnL |grep -c "$IP") -eq 0 ]; then
        iptables -I INPUT -s $IP -j DROP    fidone
    fi
done
```

方法2：通过日志获取登录状态

```
#!/bin/bash
DATE=$(date +"%b %d %H")
ABNORMAL_IP="$(tail -n10000 /var/log/auth.log |grep "$DATE" |awk '/Failed/{a[(NF-3)]++}END{for(i in a)if(a[i]>5)print i}')"
for IP in $ABNORMAL_IP; do
    if [ $(iptables -vnL |grep -c "$IP") -eq 0 ]; then
        iptables -A INPUT -s $IP -j DROP
        echo "$(date +%F %T)" - iptables -A INPUT -s $IP -j DROP" >>~/ssh-login-limit.log
    fi
done
```

Q:107、根据web访问日志，封禁请求量异常的IP，如IP在半小时后恢复正常，则解除封禁

```
#!/bin/bash
#####
####
#根据web访问日志，封禁请求量异常的IP，如IP在半小时后恢复正常，则解除封禁
#####
####
logfile=/data/log/access.log
#显示一分钟前的小时和分钟
d1=`date -d "-1 minute" +%H%M`
d2=`date +%M`
ipt=/sbin/iptables
ips=/tmp/ips.txt
block()
{
#将一分钟前的日志全部过滤出来并提取IP以及统计访问次数
grep '$d1:' $logfile|awk '{print $1}'|sort -n|uniq -c|sort -n > $ips
#利用for循环将次数超过100的IP依次遍历出来并予以封禁
for i in `awk '$1>100 {print $2}' $ips`
do
$Ipt -I INPUT -p tcp --dport 80 -s $i -j REJECT
echo "`date +%F-%T` $i" >> /tmp/badip.log
done
}
unblock()
{
#将封禁后所产生的pkts数量小于10的IP依次遍历予以解封
for a in `$Ipt -nvL INPUT --line-numbers |grep '0.0.0.0/0'|awk '$2<10 {print $1}'|sort -nr`
do
$Ipt -D INPUT $a
done
$Ipt -Z
}
#当时间在00分以及30分时执行解封函数
if [ $d2 -eq "00" ] || [ $d2 -eq "30" ]
then
#要先解再封，因为刚刚封禁时产生的pkts数量很少
unblock
block
else
block
fi
fi
```

Q:108、判断用户输入的是否为IP地址

方法1:

```
#!/bin/bash
function check_ip(){
    IP=$1
    VALID_CHECK=$(echo $IP|awk -F. '{print ($1<=255&&$2<=255&&$3<=255&&$4<=255){print "yes"}}')
}
```

```

    if echo $IP|grep -E "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}">/dev/null; then
        if [ $VALID_CHECK == "yes" ]; then
            echo "$IP available."
        else
            echo "$IP not available!"
        fi
    else
        echo "Format error!"
    fi
}
check_ip 192.168.1.1
check_ip 256.1.1.1

```

方法2:

```

#!/bin/bash
function check_ip(){
    IP=$1
    if [[ $IP =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]; then
        FIELD1=$(echo $IP|cut -d. -f1)
        FIELD2=$(echo $IP|cut -d. -f2)
        FIELD3=$(echo $IP|cut -d. -f3)
        FIELD4=$(echo $IP|cut -d. -f4)
        if [ $FIELD1 -le 255 -a $FIELD2 -le 255 -a $FIELD3 -le 255 -a $FIELD4 -le 255 ]; then
            echo "$IP available."
        else
            echo "$IP not available!"
        fi
    else
        echo "Format error!"
    fi
}
check_ip 192.168.1.1
check_ip 256.1.1.1

```

增加版:

加个死循环, 如果IP可用就退出, 不可用提示继续输入, 并使用awk判断。

```

#!/bin/bash
function check_ip(){
    local IP=$1
    VALID_CHECK=$(echo $IP|awk -F. '{ $1<=255&&$2<=255&&$3<=255&&$4<=255{print "yes"} }')
    if echo $IP|grep -E "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}">/dev/null; then
        if [ $VALID_CHECK == "yes" ]; then
            return 0
        else
            echo "$IP not available!"
            return 1
        fi
    else
        echo "Format error! Please input again."
        return 1
    fi
}

```

```
        fi
    }
    while true; do
        read -p "Please enter IP: " IP
        check_ip $IP
        [ $? -eq 0 ] && break || continue
    done
```

91-108题来自： 高效运维