1）编写**hello world**脚本

```
#!/bin/bash

# 编写hello world脚本

echo "Hello World!"
```

2）通过位置变量创建 **Linux** 系统账户及密码

```
#!/bin/bash

# 通过位置变量创建 Linux 系统账户及密码

#$1 是执行脚本的第一个参数,$2 是执行脚本的第二个参数
useradd    "$1"
echo "$2"  |  passwd  --stdin  "$1"
```

3）备份日志

```
#!/bin/bash
# 每周 5 使用 tar 命令备份/var/log 下的所有日志文件
# vim  /root/logbak.sh
# 编写备份脚本,备份后的文件名包含日期标签,防止后面的备份将前面的备份数据覆盖
# 注意 date 命令需要使用反引号括起来,反引号在键盘<tab>键上面
tar  -czf  log-`date +%Y%m%d`.tar.gz  /var/log

# crontab -e  #编写计划任务,执行备份脚本
00  03  *  *  5  /root/logbak.sh
```

4）一键部署 **LNMP(RPM** 包版本**)**

```
#!/bin/bash
# 一键部署 LNMP(RPM 包版本)
# 使用 yum 安装部署 LNMP,需要提前配置好 yum 源,否则该脚本会失败
# 本脚本使用于 centos7.2 或 RHEL7.2
yum -y install httpd
yum -y install mariadb mariadb-devel mariadb-server
yum -y install php  php-mysql

systemctl start httpd mariadb
systemctl enable httpd mariadb
```

5）监控内存和磁盘容量，小于给定值时报警

```
#!/bin/bash

# 实时监控本机内存和硬盘剩余空间,剩余内存小于500M、根分区剩余空间小于1000M时,发送报警邮件给root管理员

# 提取根分区剩余空间
disk_size=$(df / | awk '/\//{print $4}')

# 提取内存剩余空间
mem_size=$(free | awk '/Mem/{print $4}')
while :
do
# 注意内存和磁盘提取的空间大小都是以 Kb 为单位
if  [  $disk_size -le 512000 -a $mem_size -le 1024000  ]
then
    mail  -s  "Warning"  root  <<EOF
  Insufficient resources,资源不足
EOF
fi
done
```

6）猜数字游戏

```
#!/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
```

7）检测本机当前用户是否为超级管理员,如果是管理员,则使用 **yum** 安装 **vsftpd**,如果不是,则提示您非管理员(使用字串对比版本)

```
#!/bin/bash

# 检测本机当前用户是否为超级管理员,如果是管理员,则使用 yum 安装 vsftpd,如果不
# 是,则提示您非管理员(使用字串对比版本)
if [ $USER == "root" ]
then
  yum -y install vsftpd
else
    echo "您不是管理员,没有权限安装软件"
fi
```

**8**）检测本机当前用户是否为超级管理员,如果是管理员,则使用 **yum** 安装 **vsftpd**,如果不是,则提示您非管理员**(**使用 **UID** 数字对比版本**)**

```
#!/bin/bash

# 检测本机当前用户是否为超级管理员,如果是管理员,则使用 yum 安装 vsftpd,如果不
# 是,则提示您非管理员(使用 UID 数字对比版本)
if [ $UID -eq 0 ];then
    yum -y install vsftpd
else
    echo "您不是管理员,没有权限安装软件"
fi
```

**9**）编写脚本**:**提示用户输入用户名和密码,脚本自动创建相应的账户及配置密码。如果用户不输入账户名,则提示必须输入账户名并退出脚本;如果用户不输入密码,则统一使用默认的 **123456** 作为默认密码。

```
#!/bin/bash

# 编写脚本:提示用户输入用户名和密码,脚本自动创建相应的账户及配置密码。如果用户
# 不输入账户名,则提示必须输入账户名并退出脚本;如果用户不输入密码,则统一使用默
# 认的 123456 作为默认密码。
read -p "请输入用户名: " user
#使用-z 可以判断一个变量是否为空,如果为空,提示用户必须输入账户名,并退出脚本,退出码为 2
#没有输入用户名脚本退出后,使用$?查看的返回码为 2
if [ -z $user ];then
     echo "您不需输入账户名"
   exit 2
fi
#使用 stty -echo 关闭 shell 的回显功能
#使用 stty  echo 打开 shell 的回显功能
stty -echo
read -p "请输入密码: " pass
stty echo
pass=${pass:-123456}
useradd "$user"
echo "$pass" | passwd --stdin "$user"
```

## 10）输入三个数并进行升序排序

```
#!/bin/bash

# 依次提示用户输入 3 个整数,脚本根据数字大小依次排序输出 3 个数字
read -p "请输入一个整数:" num1
read -p "请输入一个整数:" num2
read -p "请输入一个整数:" num3
# 不管谁大谁小,最后都打印 echo "$num1,$num2,$num3"
# num1 中永远存最小的值,num2 中永远存中间值,num3 永远存最大值
# 如果输入的不是这样的顺序,则改变数的存储顺序,如:可以将 num1 和 num2 的值对调
tmp=0
# 如果 num1 大于 num2,就把 num1 和和 num2 的值对调,确保 num1 变量中存的是最小值
if [ $num1 -gt $num2 ];then
  tmp=$num1
  num1=$num2
  num2=$tmp
fi
# 如果 num1 大于 num3,就把 num1 和 num3 对调,确保 num1 变量中存的是最小值
if [ $num1 -gt $num3 ];then
    tmp=$num1
    num1=$num3
    num3=$tmp
fi
# 如果 num2 大于 num3,就把 num2 和 num3 对标,确保 num2 变量中存的是小一点的值
if [ $num2 -gt $num3 ];then
    tmp=$num2
    num2=$num3
    num3=$tmp
fi
echo "排序后数据(从小到大)为:$num1,$num2,$num3"
```

## 11）石头、剪刀、布游戏

```
#!/bin/bash

# 编写脚本,实现人机<石头,剪刀,布>游戏
game=(石头 剪刀 布)
num=$[RANDOM%3]
computer=${game[$num]}
# 通过随机数获取计算机的出拳
# 出拳的可能性保存在一个数组中,game[0],game[1],game[2]分别是 3 中不同的可能

echo "请根据下列提示选择您的出拳手势"
echo "1.石头"
echo "2.剪刀"
echo "3.布"

read -p "请选择 1-3:" person
case  $person  in
1)
  if [ $num -eq 0 ]
```

```
  then
    echo "平局"
    elif [ $num -eq 1 ]
    then
      echo "你赢"
  else
    echo "计算机赢"
  fi;;
2)
  if [ $num -eq 0 ]
  then
    echo "计算机赢"
    elif [ $num -eq 1 ]
    then
      echo "平局"
  else
    echo "你赢"
  fi;;
3)
  if [ $num -eq 0 ]
  then
    echo "你赢"
    elif [ $num -eq 1 ]
    then
      echo "计算机赢"
  else
    echo "平局"
  fi;;
*)
  echo "必须输入 1-3 的数字"
esac
```

**12**）编写脚本测试 **192.168.4.0/24** 整个网段中哪些主机处于开机状态,哪些主机处于关机状态**(for** 版本**)**

```
#!/bin/bash

# 编写脚本测试 192.168.4.0/24 整个网段中哪些主机处于开机状态,哪些主机处于关机
# 状态(for 版本)
for i in {1..254}
do
  # 每隔0.3秒ping一次，一共ping2次，并以1毫秒为单位设置ping的超时时间
     ping -c 2 -i 0.3 -W 1 192.168.4.$i  &>/dev/null
    if  [ $? -eq 0 ];then
        echo "192.168.4.$i is up"
     else
        echo  "192.168.4.$i is down"
     fi
done
```

**13**）编写脚本测试 **192.168.4.0/24** 整个网段中哪些主机处于开机状态,哪些主机处于关机状态**(while** 版本**)**

```
#!/bin/bash

# 编写脚本测试 192.168.4.0/24 整个网段中哪些主机处于开机状态,哪些主机处于关机
# 状态(while 版本)
i=1
while [ $i -le 254 ]
do
     ping -c 2 -i 0.3 -W 1 192.168.4.$i  &>/dev/null
     if  [ $? -eq 0 ];then
         echo "192.168.4.$i is up"
    else
         echo  "192.168.4.$i is down"
     fi
     let i++
done
```

**14**）编写脚本测试 **192.168.4.0/24** 整个网段中哪些主机处于开机状态,哪些主机处于关机状态**(**多进程版**)**

```
#!/bin/bash

# 编写脚本测试 192.168.4.0/24 整个网段中哪些主机处于开机状态,哪些主机处于关机
# 状态(多进程版)
#定义一个函数,ping 某一台主机,并检测主机的存活状态
myping(){
ping -c 2 -i 0.3 -W 1 $1  &>/dev/null
if  [ $? -eq 0 ];then
  echo "$1 is up"
else
  echo "$1 is down"
fi
}
for i in {1..254}
do
     myping 192.168.4.$i &
done
# 使用&符号,将执行的函数放入后台执行
# 这样做的好处是不需要等待ping第一台主机的回应,就可以继续并发ping第二台主机,依次类推。
```

**15**）编写脚本,显示进度条

```
#!/bin/bash

# 编写脚本,显示进度条
jindu(){
while :
do
     echo -n '#'
     sleep 0.2
done
```

```
}
jindu &
cp -a $1 $2
killall $0
echo "拷贝完成"
```

## 16）进度条,动态时针版本；定义一个显示进度的函数,屏幕快速显示| / - \

```
#!/bin/bash

# 进度条,动态时针版本
# 定义一个显示进度的函数,屏幕快速显示|  / - \
rotate_line(){
INTERVAL=0.5  #设置间隔时间
COUNT="0"     #设置4个形状的编号,默认编号为 0(不代表任何图像)
while :
do
  COUNT=`expr $COUNT + 1` #执行循环,COUNT 每次循环加 1,(分别代表4种不同的形状)
  case $COUNT in          #判断 COUNT 的值,值不一样显示的形状就不一样
  "1")                    #值为 1 显示-
          echo -e '-'"\b\c"
          sleep $INTERVAL
          ;;
    "2")                  #值为 2 显示\\,第一个\是转义
          echo -e '\\'"\b\c"
          sleep $INTERVAL
          ;;
    "3")                  #值为 3 显示|
          echo -e "|\b\c"
          sleep $INTERVAL
          ;;
   "4")                   #值为 4 显示/
          echo -e "/\b\c"
          sleep $INTERVAL
          ;;
    *)                    #值为其他时,将 COUNT 重置为 0
          COUNT="0";;
    esac
done
}
rotate_line
```

## 17）9*9 乘法表

```
#!/bin/bash

# 9*9 乘法表(编写 shell 脚本,打印 9*9 乘法表)
for i in `seq 9`
do
    for j in `seq $i`
     do
         echo -n "$j*$i=$[i*j]  "
```

```
     done
    echo
done
```

**18**）使用死循环实时显示 **eth0** 网卡发送的数据包流量

```
#!/bin/bash

# 使用死循环实时显示 eth0 网卡发送的数据包流量
while :
do
   echo  '本地网卡 eth0 流量信息如下: '
    ifconfig eth0 | grep "RX pack" | awk '{print $5}'
    ifconfig eth0 | grep "TX pack" | awk '{print $5}'
     sleep 1
done
```

**19**）使用 **user.txt** 文件中的人员名单,在计算机中自动创建对应的账户并配置初始密码本脚本执行,需要提前准备一个 **user.txt** 文件,该文件中包含有若干用户名信息

```
#!/bin/bash

# 使用 user.txt 文件中的人员名单,在计算机中自动创建对应的账户并配置初始密码
# 本脚本执行,需要提前准备一个 user.txt 文件,该文件中包含有若干用户名信息
for i in `cat user.txt`
do
     useradd  $i
     echo "123456" | passwd --stdin $i
done
```

**20**）编写批量修改扩展名脚本

```
#!/bin/bash

# 编写批量修改扩展名脚本,如批量将 txt 文件修改为 doc 文件
# 执行脚本时,需要给脚本添加位置参数
# 脚本名  txt  doc(可以将 txt 的扩展名修改为 doc)
# 脚本名  doc  jpg(可以将 doc 的扩展名修改为 jpg)
for i in `ls *.$1`
do
     mv $i ${i%.*}.$2
done
```

**21**）使用 **expect** 工具自动交互密码远程其他主机安装 **httpd** 软件

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

```
# 使用 expect 工具自动交互密码远程其他主机安装 httpd 软件

# 删除~/.ssh/known_hosts 后,ssh 远程任何主机都会询问是否确认要连接该主机
rm  -rf  ~/.ssh/known_hosts
expect <<EOF
spawn ssh 192.168.4.254
expect "yes/no" {send "yes\r"}
# 根据自己的实际情况将密码修改为真实的密码字串
expect "password" {send  "密码\r"}
expect "#" {send  "yum -y install httpd\r"}
expect "#" {send  "exit\r"}
EOF
```

## 22）一键部署 LNMP(源码安装版本)

```
#!/bin/bash

# 一键部署 LNMP(源码安装版本)
menu()
{
clear
echo "  ###############----Menu----##############"
echo "# 1. Install Nginx"
echo "# 2. Install MySQL"
echo "# 3. Install PHP"
echo "# 4. Exit Program"
echo "  ########################################"
}

choice()
{
  read -p "Please choice a menu[1-9]:" select
}

install_nginx()
{
  id nginx &>/dev/null
  if [ $? -ne 0 ];then
    useradd -s /sbin/nologin nginx
  fi
  if [ -f nginx-1.8.0.tar.gz ];then
    tar -xf nginx-1.8.0.tar.gz
    cd nginx-1.8.0
    yum -y install  gcc pcre-devel openssl-devel zlib-devel make
    ./configure --prefix=/usr/local/nginx --with-http_ssl_module
    make
    make install
    ln -s /usr/local/nginx/sbin/nginx /usr/sbin/
    cd ..
  else
    echo "没有 Nginx 源码包"
  fi
}

install_mysql()
```

```
{
  yum -y install gcc gcc-c++ cmake ncurses-devel perl
  id mysql &>/dev/null
  if [ $? -ne 0 ];then
    useradd -s /sbin/nologin mysql
  fi
  if [ -f mysql-5.6.25.tar.gz ];then
    tar -xf mysql-5.6.25.tar.gz
    cd mysql-5.6.25
    cmake .
    make
    make install
    /usr/local/mysql/scripts/mysql_install_db --user=mysql --datadir=/usr/local/mysql/data/
--basedir=/usr/local/mysql/
    chown -R root.mysql /usr/local/mysql
    chown -R mysql /usr/local/mysql/data
    /bin/cp -f /usr/local/mysql/support-files/mysql.server /etc/init.d/mysqld
    chmod +x /etc/init.d/mysqld
    /bin/cp -f /usr/local/mysql/support-files/my-default.cnf /etc/my.cnf
    echo "/usr/local/mysql/lib/" >> /etc/ld.so.conf
    ldconfig
    echo 'PATH=\$PATH:/usr/local/mysql/bin/' >> /etc/profile
    export PATH
  else
    echo  "没有 mysql 源码包"
    exit
  fi
}

install_php()
{
#安装 php 时没有指定启动哪些模块功能,如果的用户可以根据实际情况自行添加额外功能如--with-gd 等
yum  -y  install  gcc  libxml2-devel
if [ -f mhash-0.9.9.9.tar.gz ];then
  tar -xf mhash-0.9.9.9.tar.gz
  cd mhash-0.9.9.9
  ./configure
  make
  make install
  cd ..
if [ ! -f /usr/lib/libmhash.so ];then
  ln -s /usr/local/lib/libmhash.so /usr/lib/
fi
ldconfig
else
  echo "没有 mhash 源码包文件"
  exit
fi
if [ -f libmcrypt-2.5.8.tar.gz ];then
  tar -xf libmcrypt-2.5.8.tar.gz
  cd libmcrypt-2.5.8
  ./configure
  make
  make install
  cd ..
  if [ ! -f /usr/lib/libmcrypt.so ];then
    ln -s /usr/local/lib/libmcrypt.so /usr/lib/
```

```
  fi
  ldconfig
else
  echo "没有 libmcrypt 源码包文件"
  exit
fi
if [ -f php-5.4.24.tar.gz ];then
  tar -xf php-5.4.24.tar.gz
  cd php-5.4.24
  ./configure  --prefix=/usr/local/php5  --with-mysql=/usr/local/mysql  --enable-fpm    --
  enable-mbstring  --with-mcrypt  --with-mhash  --with-config-file-path=/usr/local/php5/etc  --w
  mysqli=/usr/local/mysql/bin/mysql_config
  make && make install
  /bin/cp -f php.ini-production /usr/local/php5/etc/php.ini
  /bin/cp -f /usr/local/php5/etc/php-fpm.conf.default /usr/local/php5/etc/php-fpm.conf
  cd ..
else
  echo "没有 php 源码包文件"
  exit
fi
}

while :
do
  menu
  choice
  case $select in
  1)
    install_nginx
    ;;
  2)
    install_mysql
    ;;
  3)
    install_php
    ;;
  4)
    exit
    ;;
  *)
    echo Sorry!
  esac
done
```

## 23）编写脚本快速克隆 KVM 虚拟机

```
#!/bin/bash

# 编写脚本快速克隆 KVM 虚拟机

# 本脚本针对 RHEL7.2 或 Centos7.2
# 本脚本需要提前准备一个 qcow2 格式的虚拟机模板，
# 名称为/var/lib/libvirt/images  /.rh7_template 的虚拟机模板
```

```
# 该脚本使用 qemu-img 命令快速创建快照虚拟机
# 脚本使用 sed 修改模板虚拟机的配置文件,将虚拟机名称、UUID、磁盘文件名、MAC 地址
# exit code:
#    65 -> user input nothing
#    66 -> user input is not a number
#    67 -> user input out of range
#    68 -> vm disk image exists

IMG_DIR=/var/lib/libvirt/images
BASEVM=rh7_template
read -p "Enter VM number: " VMNUM
if [ $VMNUM -le 9 ];then
VMNUM=0$VMNUM
fi

if [ -z "${VMNUM}" ]; then
    echo "You must input a number."
    exit 65
elif [[  ${VMNUM} =~ [a-z]  ]; then
    echo "You must input a number."
    exit 66
elif [ ${VMNUM} -lt 1 -o ${VMNUM} -gt 99 ]; then
    echo "Input out of range"
    exit 67
fi

NEWVM=rh7_node${VMNUM}

if [ -e $IMG_DIR/${NEWVM}.img ]; then
    echo "File exists."
    exit 68
fi

echo -en "Creating Virtual Machine disk image......\t"
qemu-img create -f qcow2 -b $IMG_DIR/.${BASEVM}.img $IMG_DIR/${NEWVM}.img &> /dev/null

echo -e "\e[32;1m[OK]\e[0m"
#virsh dumpxml ${BASEVM} > /tmp/myvm.xml

cat /var/lib/libvirt/images/.rhel7.xml > /tmp/myvm.xml
sed -i "/<name>${BASEVM}/s/${BASEVM}/${NEWVM}/" /tmp/myvm.xml
sed -i "/uuid/s/<uuid>.*<\/uuid>/<uuid>$(uuidgen)<\/uuid>/" /tmp/myvm.xml
sed -i "/${BASEVM}\.img/s/${BASEVM}/${NEWVM}/" /tmp/myvm.xml

# 修改 MAC 地址,本例使用的是常量,每位使用该脚本的用户需要根据实际情况修改这些值
# 最好这里可以使用便利,这样更适合于批量操作,可以克隆更多虚拟机
sed -i "/mac /s/a1/0c/" /tmp/myvm.xml

echo -en "Defining new virtual machine......\t\t"
virsh define /tmp/myvm.xml &> /dev/null
echo -e "\e[32;1m[OK]\e[0m"
```

## 24）点名器脚本

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

```
# 编写一个点名器脚本

# 该脚本,需要提前准备一个 user.txt 文件
# 该文件中需要包含所有姓名的信息,一行一个姓名,脚本每次随机显示一个姓名
while :
do
#统计 user 文件中有多少用户
line=`cat user.txt |wc -l`
num=$[RANDOM%line+1]
sed -n "${num}p"  user.txt
sleep 0.2
clear
done
```

**25**）查看有多少远程的 **IP** 在连接本机

```
#!/bin/bash

# 查看有多少远程的 IP 在连接本机(不管是通过 ssh 还是 web 还是 ftp 都统计)
# 使用 netstat -atn 可以查看本机所有连接的状态,-a 查看所有,
# -t仅显示 tcp 连接的信息,-n 数字格式显示
# Local Address(第四列是本机的 IP 和端口信息)
# Foreign Address(第五列是远程主机的 IP 和端口信息)
# 使用 awk 命令仅显示第 5 列数据,再显示第 1 列 IP 地址的信息
# sort 可以按数字大小排序,最后使用 uniq 将多余重复的删除,并统计重复的次数
netstat -atn  |  awk  '{print $5}'  | awk  '{print $1}' | sort -nr  |  uniq -c
```

**26**）对 **100** 以内的所有正整数相加求和**(1+2+3+4...+100)**

```
#!/bin/bash

# 对 100 以内的所有正整数相加求和(1+2+3+4...+100)
#seq 100 可以快速自动生成 100 个整数
sum=0
for i in `seq 100`
do
    sum=$[sum+i]
done
echo "总和是:$sum"
```

**27**）统计 **13:30** 到 **14:30** 所有访问 **apache** 服务器的请求有多少个

```
#!/bin/bash

# 统计 13:30 到 14:30 所有访问 apache 服务器的请求有多少个

# awk 使用-F 选项指定文件内容的分隔符是/或者:
# 条件判断$7:$8 大于等于 13:30,并且要求,$7:$8 小于等于 14:30
# 最后使用 wc -l 统计这样的数据有多少行,即多少个
```

```
awk -F "[ /:]" '$7":"$8>="13:30" && $7":"$8<="14:30"' /var/log/httpd/access_log |wc -l
```

**28**）统计 **13:30** 到 **14:30** 所有访问本机 **Aapche** 服务器的远程 **IP** 地址是什么

```
#!/bin/bash

# 统计 13:30 到 14:30 所有访问本机 Aapche 服务器的远程 IP 地址是什么
# awk 使用-F 选项指定文件内容的分隔符是/或者:
# 条件判断$7:$8 大于等于 13:30,并且要求,$7:$8 小于等于 14:30
# 日志文档内容里面,第 1 列是远程主机的 IP 地址,使用 awk 单独显示第 1 列即可
awk -F "[ /:]" '$7":"$8>="13:30" && $7":"$8<="14:30"{print $1}' /var/log/httpd/access_log
```

**29**）打印国际象棋棋盘

```
#!/bin/bash

# 打印国际象棋棋盘
# 设置两个变量,i 和 j,一个代表行,一个代表列,国际象棋为 8*8 棋盘
# i=1 是代表准备打印第一行棋盘,第 1 行棋盘有灰色和蓝色间隔输出,总共为 8 列
# i=1,j=1 代表第 1 行的第 1 列;i=2,j=3 代表第 2 行的第 3 列
# 棋盘的规律是 i+j 如果是偶数,就打印蓝色色块,如果是奇数就打印灰色色块
# 使用 echo -ne 打印色块,并且打印完成色块后不自动换行,在同一行继续输出其他色块
for i in {1..8}
do
    for j in {1..8}
    do
      sum=$[i+j]
    if [  $[sum%2] -eq 0 ];then
       echo -ne "\033[46m  \033[0m"
    else
      echo -ne "\033[47m  \033[0m"
    fi
    done
    echo
done
```

**30**）统计每个远程 **IP** 访问了本机 **apache** 几次**?**

```
#!/bin/bash

# 统计每个远程 IP 访问了本机 apache 几次?
awk  '{ip[$1]++}END{for(i in ip){print ip[i],i}}'  /var/log/httpd/access_log
```

**31**）统计当前 **Linux** 系统中可以登录计算机的账户有多少个

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

```
# 统计当前 Linux 系统中可以登录计算机的账户有多少个
#方法 1:
grep "bash$" /etc/passwd | wc -l
#方法 2:
awk -f: '/bash$/{x++}end{print x}'  /etc/passwd
```

## 32）统计/var/log 有多少个文件,并显示这些文件名

```
#!/bin/bash

# 统计/var/log 有多少个文件,并显示这些文件名
# 使用 ls 递归显示所有,再判断是否为文件,如果是文件则计数器加 1
cd  /var/log
sum=0
for i in `ls -r *`
do
   if [ -f $i ];then
       let sum++
         echo "文件名:$i"
     fi
done
echo "总文件数量为:$sum"
```

## 33）自动为其他脚本添加解释器信息

```
Docker+K8s+Jenkins 主流技术全解视频资料【干货免费分享】

#!/bin/bash

# 自动为其他脚本添加解释器信息#!/bin/bash,如脚本名为 test.sh 则效果如下:
# ./test.sh  abc.sh  自动为 abc.sh 添加解释器信息
# ./test.sh  user.sh  自动为 user.sh 添加解释器信息
# 先使用 grep 判断对象脚本是否已经有解释器信息,如果没有则使用 sed 添加解释器以及描述信息
if  !  grep  -q  "^#!"  $1; then
sed  '1i #!/bin/bash'  $1
sed  '2i #Description: '
fi
# 因为每个脚本的功能不同,作用不同,所以在给对象脚本添加完解释器信息,以及 Description 后还希望
# 继续编辑具体的脚本功能的描述信息,这里直接使用 vim 把对象脚本打开,并且光标跳转到该文件的第 2 行
vim +2 $1
```

## 34）自动化部署 varnish 源码包软件

```
#!/bin/bash

# 自动化部署 varnish 源码包软件
# 本脚本需要提前下载 varnish-3.0.6.tar.gz 这样一个源码包软件,该脚本即可用自动源码安装部署软件

yum -y install gcc readline-devel pcre-devel
useradd -s /sbin/nologin varnish
```

```
tar -xf varnish-3.0.6.tar.gz
cd varnish-3.0.6

# 使用 configure,make,make install 源码安装软件包
./configure --prefix=/usr/local/varnish
make && make install

# 在源码包目录下,将相应的配置文件拷贝到 Linux 系统文件系统中
# 默认安装完成后,不会自动拷贝或安装配置文件到 Linux 系统,所以需要手动 cp 复制配置文件
# 并使用 uuidgen 生成一个随机密钥的配置文件
cp redhat/varnish.initrc /etc/init.d/varnish
cp redhat/varnish.sysconfig /etc/sysconfig/varnish
cp redhat/varnish_reload_vcl /usr/bin/
ln -s /usr/local/varnish/sbin/varnishd /usr/sbin/
ln -s /usr/local/varnish/bin/* /usr/bin
mkdir /etc/varnish
cp /usr/local/varnish/etc/varnish/default.vcl /etc/varnish/
uuidgen > /etc/varnish/secret
```

## 35）编写 **nginx** 启动脚本

```
#!/bin/bash

# 编写 nginx 启动脚本
# 本脚本编写完成后,放置在/etc/init.d/目录下,就可以被 Linux 系统自动识别到该脚本
# 如果本脚本名为/etc/init.d/nginx,则 service nginx start 就可以启动该服务
# service nginx stop 就可以关闭服务
# service nginx restart 可以重启服务
# service nginx status 可以查看服务状态
program=/usr/local/nginx/sbin/nginx
pid=/usr/local/nginx/logs/nginx.pid
start(){
if [ -f $pid ];then
  echo  "nginx 服务已经处于开启状态"
else
  $program
fi
stop(){
if [ -! -f $pid ];then
  echo "nginx 服务已经关闭"
else
  $program -s stop
  echo "关闭服务 ok"
fi
}
status(){
if [ -f $pid ];then
  echo "服务正在运行..."
else
  echo "服务已经关闭"
fi
}

case $1 in
start)
```

```
  start;;
stop)
  stop;;
restart)
  stop
  sleep 1
  start;;
status)
  status;;
*)
  echo  "你输入的语法格式错误"
esac
```

## 36）自动对磁盘分区、格式化、挂载

```
 #!/bin/bash

# 自动对磁盘分区、格式化、挂载
# 对虚拟机的 vdb 磁盘进行分区格式化,使用<<将需要的分区指令导入给程序 fdisk
# n(新建分区),p(创建主分区),1(分区编号为 1),两个空白行(两个回车,相当于将整个磁盘分一个区)
# 注意:1 后面的两个回车(空白行)是必须的!
fdisk /dev/vdb << EOF
n
p
1



wq
EOF
#格式化刚刚创建好的分区
mkfs.xfs   /dev/vdb1
#创建挂载点目录
if [ -e /data ]; then
exit
fi
mkdir /data
#自动挂载刚刚创建的分区,并设置开机自动挂载该分区
echo '/dev/vdb1     /data    xfs    defaults        1 2'  >> /etc/fstab
mount -a
```

## 37）自动优化 Linux 内核参数

```
#!/bin/bash

# 自动优化 Linux 内核参数
#脚本针对 RHEL7
cat >> /usr/lib/sysctl.d/00-system.conf <<EOF
fs.file-max=65535
net.ipv4.tcp_timestamps = 0
net.ipv4.tcp_synack_retries = 5
net.ipv4.tcp_syn_retries = 5
net.ipv4.tcp_tw_recycle = 1
```

```
net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_fin_timeout = 30
#net.ipv4.tcp_keepalive_time = 120
net.ipv4.ip_local_port_range = 1024  65535
kernel.shmall = 2097152
kernel.shmmax = 2147483648
kernel.shmmni = 4096
kernel.sem = 5010 641280 5010 128
net.core.wmem_default=262144
net.core.wmem_max=262144
net.core.rmem_default=4194304
net.core.rmem_max=4194304
net.ipv4.tcp_fin_timeout = 10
net.ipv4.tcp_keepalive_time = 30
net.ipv4.tcp_window_scaling = 0
net.ipv4.tcp_sack = 0
EOF

sysctl -p
```

## 38）切割 Nginx 日志文件(防止单个文件过大,后期处理很困难)

```
#mkdir  /data/scripts
#vim   /data/scripts/nginx_log.sh
#!/bin/bash

# 切割 Nginx 日志文件(防止单个文件过大,后期处理很困难)
logs_path="/usr/local/nginx/logs/"
mv ${logs_path}access.log ${logs_path}access_$(date -d "yesterday" +"%Y%m%d").log
kill -USR1  `cat /usr/local/nginx/logs/nginx.pid`

# chmod +x  /data/scripts/nginx_log.sh
# crontab  -e                    #脚本写完后,将脚本放入计划任务每天执行一次脚本
0  1  *  *   *   /data/scripts/nginx_log.sh
```

## 39）检测 MySQL 数据库连接数量

```
#!/bin/bash

# 检测 MySQL 数据库连接数量

# 本脚本每 2 秒检测一次 MySQL 并发连接数,可以将本脚本设置为开机启动脚本,或在特定时间段执行
# 以满足对 MySQL 数据库的监控需求,查看 MySQL 连接是否正常
# 本案例中的用户名和密码需要根据实际情况修改后方可使用
log_file=/var/log/mysql_count.log
user=root
passwd=123456
while :
do
    sleep 2
    count=`mysqladmin  -u  "$user"  -p  "$passwd"   status |  awk '{print $4}'`
    echo "`date +%Y-%m-%d` 并发连接数为:$count" >> $log_file
```

```
done
```

## 40）根据 md5 校验码,检测文件是否被修改

```
#!/bin/bash

# 根据 md5 校验码,检测文件是否被修改
# 本示例脚本检测的是/etc 目录下所有的 conf 结尾的文件,根据实际情况,您可以修改为其他目录或文件
# 本脚本在目标数据没有被修改时执行一次,当怀疑数据被人篡改,再执行一次
# 将两次执行的结果做对比,MD5 码发生改变的文件,就是被人篡改的文件
for i in $(ls /etc/*.conf)
do
  md5sum "$i" >> /var/log/conf_file.log
done
```

## 41）检测 MySQL 服务是否存活

```
#!/bin/bash

# 检测 MySQL 服务是否存活

# host 为你需要检测的 MySQL 主机的 IP 地址,user 为 MySQL 账户名,passwd 为密码
# 这些信息需要根据实际情况修改后方可使用
host=192.168.51.198
user=root
passwd=123456
mysqladmin -h '$host' -u '$user' -p'$passwd' ping &>/dev/null
if [ $? -eq 0 ]
then
        echo "MySQL is UP"
else
        echo "MySQL is down"
fi
```

## 42）备份 MySQL 的 shell 脚本(mysqldump版本)

```
#!/bin/bash

# 备份 MySQL 的 shell 脚本(mysqldump版本)

# 定义变量 user(数据库用户名),passwd(数据库密码),date(备份的时间标签)
# dbname(需要备份的数据库名称,根据实际需求需要修改该变量的值,默认备份 mysql 数据库)

user=root
passwd=123456
dbname=mysql
date=$(date +%Y%m%d)

# 测试备份目录是否存在,不存在则自动创建该目录
[ ! -d /mysqlbackup ] && mkdir /mysqlbackup
```

```
# 使用 mysqldump 命令备份数据库
mysqldump -u "$user" -p "$passwd" "$dbname" > /mysqlbackup/"$dbname"-${date}.sql
```

## 43）将文件中所有的小写字母转换为大写字母

```
#!/bin/bash

# 将文件中所有的小写字母转换为大写字母
# $1是位置参数,是你需要转换大小写字母的文件名称
# 执行脚本,给定一个文件名作为参数,脚本就会将该文件中所有的小写字母转换为大写字母
tr "[a-z]" "[A-Z]" < $1
```

## 44）非交互自动生成 SSH 密钥文件

```
#!/bin/bash

# 非交互自动生成 SSH 密钥文件
# -t 指定 SSH 密钥的算法为 RSA 算法;-N 设置密钥的密码为空;-f 指定生成的密钥文件>存放在哪里
rm  -rf  ~/.ssh/{known_hosts,id_rsa*}
ssh-keygen -t RSA -N '' -f ~/.ssh/id_rsa
```

## 45）检查特定的软件包是否已经安装

```
#!/bin/bash

# 检查特定的软件包是否已经安装
if [ $# -eq 0 ];then
  echo "你需要制定一个软件包名称作为脚本参数"
  echo "用法:$0 软件包名称 ..."
fi
# $@提取所有的位置变量的值,相当于$*
for package in "$@"
do
    if rpm -q ${package} &>/dev/null ;then
    echo -e "${package}\033[32m 已经安装\033[0m"
    else
    echo -e "${package}\033[34;1m 未安装\033[0m"
    fi
done
```

## 46）监控 HTTP 服务器的状态(测试返回码)

```
#!/bin/bash

# 监控 HTTP 服务器的状态(测试返回码)

# 设置变量,url为你需要检测的目标网站的网址(IP 或域名),比如百度
```

```
url=http://http://183.232.231.172/index.html

# 定义函数 check_http:
# 使用 curl 命令检查 http 服务器的状态
# -m 设置curl不管访问成功或失败,最大消耗的时间为 5 秒,5 秒连接服务为相应则视为无法连接
# -s 设置静默连接,不显示连接时的连接速度、时间消耗等信息
# -o 将 curl 下载的页面内容导出到/dev/null(默认会在屏幕显示页面内容)
# -w 设置curl命令需要显示的内容%{http_code},指定curl返回服务器的状态码
check_http()
{
        status_code=$(curl -m 5 -s -o /dev/null -w %{http_code} $url)
}


while :
do
        check_http
        date=$(date +%Y%m%d-%H:%M:%S)


# 生成报警邮件的内容
        echo "当前时间为:$date
        $url 服务器异常,状态码为${status_code}.
        请尽快排查异常." > /tmp/http$$.pid


# 指定测试服务器状态的函数,并根据返回码决定是发送邮件报警还是将正常信息写入日志
        if [ $status_code -ne 200 ];then
                mail -s Warning root < /tmp/http$$.pid
        else
                echo "$url 连接正常" >> /var/log/http.log
        fi
        sleep 5
done
```

## 47）自动添加防火墙规则,开启某些服务或端口(适用于 RHEL7)

```
#!/bin/bash

# 自动添加防火墙规则,开启某些服务或端口(适用于 RHEL7)
#
# 设置变量定义需要添加到防火墙规则的服务和端口号
# 使用 firewall-cmd --get-services 可以查看 firewall 支持哪些服务
service="nfs http ssh"
port="80 22 8080"

# 循环将每个服务添加到防火墙规则中
for i in $service
do
    echo "Adding $i service to firewall"
    firewall-cmd  --add-service=${i}
done

#循环将每个端口添加到防火墙规则中
for i in $port
```

```
do
    echo "Adding $i Port to firewall"
    firewall-cmd --add-port=${i}/tcp
done
#将以上设置的临时防火墙规则,转换为永久有效的规则(确保重启后有效)
firewall-cmd  --runtime-to-permanent
```

## 48）使用脚本自动创建逻辑卷

```
#!/bin/bash

# 使用脚本自动创建逻辑卷


# 清屏,显示警告信息,创建将磁盘转换为逻辑卷会删除数据
clear
echo -e "\033[32m           !!!!!!警告(Warning)!!!!!!\033[0m"
echo
echo "++++++++++++++++++++++++++++++++++++++++++++++++++"
echo "脚本会将整个磁盘转换为 PV,并删除磁盘上所有数据!!!"
echo "This Script will destroy all data on the Disk"
echo "++++++++++++++++++++++++++++++++++++++++++++++++++"
echo
read -p "请问是否继续 y/n?:" sure

# 测试用户输入的是否为 y,如果不是则退出脚本
[ $sure != y ] && exit

# 提示用户输入相关参数(磁盘、卷组名称等数据),并测试用户是否输入了这些值,如果没有输入,则脚本退出
read -p "请输入磁盘名称,如/dev/vdb:" disk
[ -z $disk ] && echo "没有输入磁盘名称" && exit
read -p "请输入卷组名称:" vg_name
[ -z $vg_name ] && echo "没有输入卷组名称" && exit
read -p "请输入逻辑卷名称:" lv_name
[ -z $lv_name ] && echo "没有输入逻辑卷名称" && exit
read -p "请输入逻辑卷大小:" lv_size
[ -z $lv_size ] && echo "没有输入逻辑卷大小" && exit
# 使用命令创建逻辑卷
pvcreate $disk
vgcreate $vg_name $disk
lvcreate -L ${lv_size}M -n ${lv_name}  ${vg_name}
```

## 49）显示 CPU 厂商信息

```
#!/bin/bash

# 显示 CPU 厂商信息
awk '/vendor_id/{print $3}' /proc/cpuinfo | uniq
```

## 50）删除某个目录下大小为 0 的文件

```
#!/bin/bash

# 删除某个目录下大小为 0 的文件

#/var/www/html 为测试目录,脚本会清空该目录下所有 0 字节的文件
dir="/var/www/html"
find $dir -type f -size 0 -exec rm -rf {} \;
```

## 51）查找 Linux 系统中的僵尸进程

```
#!/bin/bash

# 查找 Linux 系统中的僵尸进程
# awk 判断 ps 命令输出的第 8 列为 Z 是,显示该进程的 PID 和进程命令
ps aux | awk '{if($8 == "Z"){print $2,$11}}'
```

## 52）提示用户输入年份后判断该年是否为闰年

```
#!/bin/bash

# 提示用户输入年份后判断该年是否为闰年


# 能被4整除并且并不能被100整除的年份是闰年
# 能被400整除的年份也是闰年
read -p "请输入一个年份:" year


if [ "$year" = "" ];then
    echo "没有输入年份"
    exit
fi
#使用正则测试变量 year 中是否包含大小写字母
if [[ "$year" =~ [a-Z] ]];then
    echo "你输入的不是数字"
    exit
fi
# 判断是否为闰年
if [ $[year % 4] -eq 0 ] && [ $[year % 100] -ne 0 ];then
    echo "$year年是闰年"
elif [ $[year % 400] -eq 0 ];then
    echo "$year年是闰年"
else
    echo "$year年不是闰年"
fi
```

## 53）生成随机密码(urandom 版本)

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

```
# 生成随机密码(urandom 版本)

# /dev/urandom 文件是 Linux 内置的随机设备文件
# cat /dev/urandom 可以看看里面的内容,ctrl+c 退出查看
# 查看该文件内容后,发现内容有些太随机,包括很多特殊符号,我们需要的密码不希望使用这些符号
# tr -dc '_A-Za-z0-9' < /dev/urandom
# 该命令可以将随机文件中其他的字符删除,仅保留大小写字母,数字,下划线,但是内容还是太多
# 我们可以继续将优化好的内容通过管道传递给 head 命令,在大量数据中仅显示头 10 个字节
# 注意 A 前面有个下划线
tr -dc '_A-Za-z0-9' </dev/urandom | head -c 10
```

## 54）生成随机密码(字串截取版本)

```
#!/bin/bash

# 生成随机密码(字串截取版本)

# 设置变量 key,存储密码的所有可能性(密码库),如果还需要其他字符请自行添加其他密码字符
# 使用$#统计密码库的长度
key="0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"
num=${#key}
# 设置初始密码为空
pass=''
# 循环 8 次,生成随机密码
# 每次都是随机数对密码库的长度取余,确保提取的密码字符不超过密码库的长度
# 每次循环提取一位随机密码,并将该随机密码追加到 pass 变量的最后
for i in {1..8}
do
  index=$[RANDOM%num]
  pass=$pass${key:$index:1}
done
echo $pass
```

## 55）生成随机密码(UUID 版本,16 进制密码)

```
#!/bin/bash

# 生成随机密码(UUID 版本,16 进制密码)
uuidgen
```

## 56）生成随机密码(进程 ID 版本,数字密码)

```
#!/bin/bash

# 生成随机密码(进程 ID 版本,数字密码)
echo $$
```

## 57

）测试用户名与密码是否正确

```
#!/bin/bash

# 测试用户名与密码是否正确
#用户名为 tom 并且密码为 123456,则提示登录成功,否则提示登录失败
read -p "请输入用户名:"  user
read -p "请输入密码:"    pass
if [ "$user" == 'tom' -a "$pass" == '123456' ];then
  echo "Login successful"
else
  echo "Login Failed"
fi
```

**58**）循环测试用户名与密码是否正确

```
#!/bin/bash

# 循环测试用户名与密码是否正确
# 循环测试用户的账户名和密码,最大测试 3 次,输入正确提示登录成功,否则提示登录失败
# 用户名为 tom 并且密码为 123456
for i in {1..3}
do
  read -p "请输入用户名:" user
  read -p "请输入密码:"   pass
if [ "$user" == 'tom' -a "$pass" == '123456' ];then
    echo "Login successful"
     exit
fi
done
echo "Login Failed"
```

**59**）**Shell** 脚本的 **fork** 炸弹

```
#!/bin/bash

# Shell 脚本的 fork 炸弹


# 快速消耗计算机资源,致使计算机死机
# 定义函数名为.(点), 函数中递归调用自己并放入后台执行
.() { .|.& };.
```

**60**）批量下载有序文件**(pdf**、图片、视频等等**)**

```
#!/bin/bash

# 批量下载有序文件(pdf、图片、视频等等)
```

```
# 本脚本准备有序的网络资料进行批量下载操作(如 01.jpg,02.jpg,03.jpg)
# 设置资源来源的域名连接
url="http://www.baidu.com/"
echo  "开始下载..."
sleep 2
type=jpg
for i in `seq 100`
     echo "正在下载$i.$type"
  curl $url/$i.$type -o /tmp/${i}$type
     sleep 1
done
#curl 使用-o 选项指定下载文件另存到哪里.
```

## 61）显示当前计算机中所有账户的用户名称

```
 #!/bin/bash

 # 显示当前计算机中所有账户的用户名称

# 下面使用3种不同的方式列出计算机中所有账户的用户名
# 指定以:为分隔符,打印/etc/passwd 文件的第 1 列
awk -F: '{print $1}' /etc/passwd

# 指定以:为分隔符,打印/etc/passwd 文件的第 1 列
cut -d: -f1 /etc/passwd

# 使用 sed 的替换功能,将/etc/passwd 文件中:后面的所有内容替换为空(仅显示用户名)
sed 's/:.*//' /etc/passwd
```

## 62）制定目录路径,脚本自动将该目录使用 tar 命令打包备份到/data目录

```
#!/bin/bash

# 制定目录路径,脚本自动将该目录使用 tar 命令打包备份到/data目录


[ ! -d /data ] && mkdir /data
[ -z $1 ] && exit
if [ -d $1 ];then
  tar -czf /data/$1.-`date +%Y%m%d`.tar.gz $1
else
    echo "该目录不存在"
fi
```

## 63）显示进度条(回旋镖版)

```
#!/bin/bash

# 显示进度条(回旋镖版)
```

```
while :
do
  clear
  for i in {1..20}
  do
    echo -e "\033[3;${i}H*"
    sleep 0.1
    done
    clear
    for i in {20..1}
    do
    echo -e "\033[3;${i}H*"
    sleep 0.1
    done
    clear
done
```

## 64）安装 LAMP 环境(yum 版本)

```
#!/bin/bash

# 安装 LAMP 环境(yum 版本)
# 本脚本适用于 RHEL7(RHEL6 中数据库为 mysql)
yum makecache &>/dev/null
num=$(yum repolist | awk '/repolist/{print $2}' | sed 's/,//')
if [ $num -lt 0 ];then
  yum -y install httpd
  yum -y install mariadb mariadb-server mariadb-devel
  yum -y install php php-mysql
else
  echo "未配置 yum 源..."
fi
```

## 65）循环关闭局域网中所有主机

```
#!/bin/bash

# 循环关闭局域网中所有主机
# 假设本机为 192.168.4.100,编写脚本关闭除自己外的其他所有主机
# 脚本执行,需要提前给所有其他主机传递 ssh 密钥,满足无密码连接
for i in {1..254}
do
  [ $i -eq 100 ] && continue
  echo "正在关闭 192.168.4.$i..."
  ssh 192.168.4.$i poweroff
done
```

## 66）获取本机 MAC 地址

```
#!/bin/bash

# 获取本机 MAC 地址
ip a s | awk 'BEGIN{print  " 本 机 MAC 地 址 信 息 如 下 :"}/^[0-9]/{print $2;getline;if($0~/link\
{print $2}}' | grep -v lo:
# awk 读取 ip 命令的输出,输出结果中如果有以数字开始的行,先显示该行的地 2 列(网卡名称),
# 接着使用 getline 再读取它的下一行数据,判断是否包含 link/ether
# 如果保护该关键词,就显示该行的第 2 列(MAC 地址)
# lo 回环设备没有 MAC,因此将其屏蔽,不显示
```

## 67）自动配置 rsynd 服务器的配置文件 rsyncd.conf

```
#!/bin/bash

# 自动配置 rsynd 服务器的配置文件 rsyncd.conf

# See rsyncd.conf man page for more options.

[ ! -d /home/ftp ] && mkdir /home/ftp
echo 'uid = nobody
gid = nobody
use chroot = yes
max connections = 4
pid file = /var/run/rsyncd.pid
exclude = lost+found/
transfer logging = yes
timeout = 900
ignore nonreadable = yes
dont compress   = *.gz *.tgz *.zip *.z *.Z *.rpm *.deb *.bz2
[ftp]
    path = /home/ftp
    comment = share' > /etc/rsyncd.conf
```

## 68）修改 Linux 系统的最大打开文件数量

```
#!/bin/bash

# 修改 Linux 系统的最大打开文件数量

# 往/etc/security/limits.conf 文件的末尾追加两行配置参数,修改最大打开文件数量为 65536
cat >> /etc/security/limits.conf <<EOF
* soft nofile  65536
* hard nofile  65536
EOF
```

## 69）设置 Python 支持自动命令补齐功能

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

```
# 设置 Python 支持自动命令补齐功能

# Summary:Enable tab complete for python
# Description:

Needs import readline and rlcompleter module
#
import readline
#
import rlcompleter
#
help(rlcompleter) display detail: readline.parse_and_bind('tab: complete')
#
man python display detail: PYTHONSTARTUP variable

if  [ ! -f /usr/bin/tab.py ];then
  cat >> /usr/bin/tab.py <<EOF
import readline
import rlcompleter
readline.parse_and_bind('tab: complete')
EOF
fi
sed  -i '$a export PYTHONSTARTUP=/usr/bin/tab.py' /etc/profile
source /etc/profile
```

## 70）自动修改计划任务配置文件

```
Docker+K8s+Jenkins 主流技术全解视频资料【干货免费分享】

#!/bin/bash

# 自动修改计划任务配置文件
read -p "请输入分钟信息(00-59):" min
read -p "请输入小时信息(00-24):" hour
read -p "请输入日期信息(01-31):" date
read -p "请输入月份信息(01-12):" month
read -p "请输入星期信息(00-06):" weak
read -p "请输入计划任务需要执行的命令或脚本:" program
echo "$min $hour $date $month $weak $program" >> /etc/crontab
```

## 71）使用脚本循环创建三位数字的文本文件(111-999 的文件)

```
#!/bin/bash

# 使用脚本循环创建三位数字的文本文件(111-999 的文件)

for i in {1..9}
do
  for j in {1..9}
  do
    for k in {1..9}
    do
```

```
      touch /tmp/$i$j$k.txt
    done
    done
done
```

## 72）找出/etc/passwd 中能登录的用户,并将对应在/etc/shadow 中第二列密码提出处理

```
#!/bin/bash

# 找出/etc/passwd 中能登录的用户,并将对应在/etc/shadow 中第二列密码提出处理

user=$(awk -F: '/bash$/{print $1}' /etc/passwd)
for i in $user
do
  awk -F: -v x=$i '$1==x{print $1,$2}' /etc/shadow
done
```

## 73）统计/etc/passwd 中 root 出现的次数

```
#!/bin/bash

# 统计/etc/passwd 中 root 出现的次数

#每读取一行文件内容,即从第 1 列循环到最后 1 列,依次判断是否包含 root 关键词,如果包含则 x++
awk -F: '{i=1;while(i<=NF){if($i~/root/){x++};i++}} END{print "root 出现次数为"x}' /etc/passwd
```

## 74）统计 Linux 进程相关数量信息

```
#!/bin/bash

# 统计 Linux 进程相关数量信息
running=0
sleeping=0
stoped=0
zombie=0
# 在 proc 目录下所有以数字开始的都是当前计算机正在运行的进程的进程 PID
# 每个 PID 编号的目录下记录有该进程相关的信息
for pid in /proc/[1-9]*
do
  procs=$[procs+1]
  stat=$(awk '{print $3}' $pid/stat)
# 每个 pid 目录下都有一个 stat 文件,该文件的第 3 列是该进程的状态信息
    case $stat in
    R)
    running=$[running+1]
    ;;
    T)
    stoped=$[stoped+1]
    ;;
    S)
```

```
    sleeping=$[sleeping+1]
    ;;
    Z)
     zombie=$[zombie+1]
     ;;
    esac
done
echo "进程统计信息如下"
echo "总进程数量为:$procs"
echo "Running 进程数为:$running"
echo "Stoped 进程数为:$stoped"
echo "Sleeping 进程数为:$sleeping"
echo "Zombie 进程数为:$zombie"
```

## 75）从键盘读取一个论坛积分,判断论坛用户等级

```
#!/bin/bash

# 从键盘读取一个论坛积分,判断论坛用户等级

#等级分类如下:
#  大于等于 90         神功绝世
#  大于等于 80,小于 90        登峰造极
#  大于等于 70,小于 80        炉火纯青
#  大于等于 60,小于 70        略有小成
#  小于 60                初学乍练
read -p "请输入积分(0-100):" JF
if [ $JF -ge 90 ] ; then
  echo "$JF 分,神功绝世"
elif [ $JF -ge 80 ] ; then
    echo "$JF 分,登峰造极"
elif [ $JF -ge 70 ] ; then
    echo "$JF 分,炉火纯青"
elif [ $JF -lt 60 ] ; then
    echo "$JF 分,略有小成"
else
    echo "$JF 分,初学乍练"
fi
```

## 76）判断用户输入的数据类型(字母、数字或其他)

```
#!/bin/bash

# 判断用户输入的数据类型(字母、数字或其他)
read -p "请输入一个字符:" KEY
case "$KEY" in
  [a-z]|[A-Z])
    echo "字母"
    ;;
  [0-9])
    echo "数字"
    ;;
```

```
  *)
    echo "空格、功能键或其他控制字符"
esac
```

## 77）显示进度条(数字版)

```
#!/bin/bash

# 显示进度条(数字版)
# echo 使用-e 选项后,在打印参数中可以指定 H,设置需要打印内容的 x,y 轴的定位坐标
# 设置需要打印内容在第几行,第几列
for i in {1..100}
do
    echo -e "\033[6;8H["
    echo -e "\033[6;9H$i%"
    echo -e "\033[6;13H]"
    sleep 0.1
done
```

## 78）打印斐波那契数列

```
#!/bin/bash

# 打印斐波那契数列(该数列的特点是后一个数字,永远都是前 2 个数字之和)

# 斐波那契数列后一个数字永远是前 2 个数字之和
# 如:0  1  1  2  3  5  8  13 ... ...
list=(0 1)
for i in `seq 2 11`
do
  list[$i]=`expr ${list[-1]} + ${list[-2]}`
done
echo ${list[@]}
```

## 79）判断用户输入的是 Yes 或 NO

```
#!/bin/bash

# 判断用户输入的是 Yes 或 NO

read -p  "Are you sure?[y/n]:"  sure
case  $sure  in
  y|Y|Yes|YES)
    echo "you enter $a"
    ;;
    n|N|NO|no)
     echo "you enter $a"
     ;;
    *)
     echo "error";;
```

```
esac
```

## 80）显示本机 Linux 系统上所有开放的端口列表

```
#!/bin/bash

# 显示本机 Linux 系统上所有开放的端口列表

# 从端口列表中观测有没有没用的端口,有的话可以将该端口对应的服务关闭,防止意外的攻击可能性
ss -nutlp | awk '{print $1,$5}' | awk -F"[: ]" '{print "协议:"$1,"端口号:"$NF}' | grep "
[0-9]" | uniq
```

## 81）将 Linux 系统中 UID 大于等于 1000 的普通用户都删除

```
#!/bin/bash

# 将 Linux 系统中 UID 大于等于 1000 的普通用户都删除

# 先用 awk 提取所有 uid 大于等于 1000 的普通用户名称
# 再使用 for 循环逐个将每个用户删除即可
user=$(awk -F: '$3>=1000{print $1}' /etc/passwd)
for i in $user
do
     userdel -r $i
done
```

## 82）使用脚本开启关闭虚拟机

```
#!/bin/bash

# 使用脚本开启关闭虚拟机
# 脚本通过调用virsh命令实现对虚拟机的管理,如果没有该命令,需要安装 libvirt-client 软件包
# $1是脚本的第1个参数,$2是脚本的第2个参数
# 第1个参数是你希望对虚拟机进行的操作指令,第2个参数是虚拟机名称
case $1 in
  list)
    virsh list --all
    ;;
  start)
    virsh start $2
    ;;
  stop)
    virsh destroy $2
    ;;
  enable)
    virsh autostart $2
    ;;
  disable)
    virsh autostart --disable $2
    ;;
```

```
  *)
    echo "Usage:$0 list"
    echo "Usage:$0 [start|stop|enable|disable]  VM_name"
    cat << EOF
    #list       显示虚拟机列表
    #start      启动虚拟机
    #stop       关闭虚拟机
    #enable     设置虚拟机为开机自启
    #disable    关闭虚拟机开机自启功能
    EOF
    ;;
esac
```

## 83）调整虚拟机内存参数的 shell 脚本

```
#!/bin/bash

# 调整虚拟机内存参数的 shell 脚本
# 脚本通过调用 virsh 命令实现对虚拟机的管理,如果没有该命令,需要安装 libvirt-client 软件包
cat << EOF
1.调整虚拟机最大内存数值
2.调整实际分配给虚拟机的内存数值
EOF
read -p "请选择[1-2]:" select
case $select in
  1)
      read -p "请输入虚拟机名称" name
      read -p "请输入最大内存数值(单位:k):" size
      virsh setmaxmem $name --size $size --config
      ;;
  2)
      read -p "请输入虚拟机名称" name
      read -p "请输入实际分配内存数值(单位:k):" size
      virsh setmem $name $size
      ;;
  *)
      echo "Error"
      ;;
esac
```

## 84）查看 KVM 虚拟机中的网卡信息(不需要进入启动或进入虚拟机)

```
#!/bin/bash

# 查看 KVM 虚拟机中的网卡信息(不需要进入启动或进入虚拟机)

# 该脚本使用 guestmount 工具,可以将虚拟机的磁盘系统挂载到真实机文件系统中
# Centos7.2 中安装 libguestfs-tools-c 可以获得 guestmount 工具
# 虚拟机可以启动或者不启动都不影响该脚本的使用
# 将虚拟机磁盘文件挂载到文件系统后,就可以直接读取磁盘文件中的网卡配置文件中的数据
clear
mountpoint="/media/virtimage"
```

```
[ ! -d $mountpoint ] && mkdir $mountpoint
read -p "输入虚拟机名称:" name
echo "请稍后..."
# 如果有设备挂载到该挂载点,则先 umount 卸载
if mount | grep -q "$mountpoint" ;then
  umount $mountpoint
fi
# 只读的方式,将虚拟机的磁盘文件挂载到特定的目录下,这里是/media/virtimage 目录
guestmount -r -d $name -i $mountpoint
echo
echo "----------------------------------------"
echo -e "\033[32m$name 虚拟机中网卡列表如下:\033[0m"
dev=$(ls /media/virtimage/etc/sysconfig/network-scripts/ifcfg-* |awk -F"[/-]" '{print $9}')
echo $dev
echo "----------------------------------------"
echo
echo
echo "++++++++++++++++++++++++++++++++++++++++"
echo -e "\033[32m 网卡 IP 地址信息如下:\033[0m"
for i in $dev
do
  echo -n "$i:"
  grep -q "IPADDR" /media/virtimage/etc/sysconfig/network-scripts/ifcfg-$i || echo "未配置 IP地
址"
  awk -F= '/IPADDR/{print $2}' /media/virtimage/etc/sysconfig/network-scripts/ifcfg-$i
done
echo "++++++++++++++++++++++++++++++++++++++++"
```

## 85）不登陆虚拟机,修改虚拟机网卡 IP 地址

```
#!/bin/bash

# 不登陆虚拟机,修改虚拟机网卡 IP 地址

# 该脚本使用 guestmount 工具,Centos7.2 中安装 libguestfs-tools-c 可以获得 guestmount 工具
# 脚本在不登陆虚拟机的情况下,修改虚拟机的 IP 地址信息
# 在某些环境下,虚拟机没有 IP 或 IP 地址与真实主机不在一个网段
# 真实主机在没有 virt-manger 图形的情况下,远程连接虚拟机很麻烦
# 该脚本可以解决类似的问题
read -p "请输入虚拟机名称:" name
if virsh domstate $name | grep -q running ;then
  echo "修改虚拟机网卡数据,需要关闭虚拟机"
  virsh destroy $name
fi
mountpoint="/media/virtimage"
[ ! -d $mountpoint ] && mkdir $mountpoint
echo "请稍后..."
if mount | grep -q "$mountpoint" ;then
  umount $mountpoint
fi
guestmount  -d $name -i $mountpoint
read -p "请输入需要修改的网卡名称:" dev
read -p "请输入 IP 地址:" addr
# 判断原本网卡配置文件中是否有 IP 地址,有就修改该 IP,没有就添加一个新的 IP 地址
if grep -q "IPADDR"  $mountpoint/etc/sysconfig/network-scripts/ifcfg-$dev ;then
```

```
  sed -i "/IPADDR/s/=.*/=$addr/"  $mountpoint/etc/sysconfig/network-scripts/ifcfg-$dev
else
  echo "IPADDR=$addr" >> $mountpoint/etc/sysconfig/network-scripts/ifcfg-$dev
fi
# 如果网卡配置文件中有客户配置的 IP 地址,则脚本提示修改 IP 完成
awk -F= -v x=$addr '$2==x{print "完
成..."}'  $mountpoint/etc/sysconfig/network-scripts/ifcfg-$dev
```

## 86）破解虚拟机密码,无密码登陆虚拟机系统

```
#!/bin/bash

# 破解虚拟机密码,无密码登陆虚拟机系统

# 该脚本使用 guestmount 工具,Centos7.2 中安装 libguestfs-tools-c 可以获得 guestmount 工具
read -p "请输入虚拟机名称:" name
if virsh domstate $name | grep -q running ;then
  echo "破解,需要关闭虚拟机"
  virsh destroy $name
fi
mountpoint="/media/virtimage"
[ ! -d $mountpoint ] && mkdir $mountpoint
echo "请稍后..."
if mount | grep -q "$mountpoint" ;then
  umount $mountpoint
fi
guestmount -d $name -i $mountpoint
# 将 passwd 中密码占位符号 x 删除,该账户即可实现无密码登陆系统
sed -i "/^root/s/x//" $mountpoint/etc/passwd
```

## 87）Shell 脚本对信号的处理,执行脚本后,按键盘 Ctrl+C 无法终止的脚本

```
#!/bin/bash

# Shell 脚本对信号的处理,执行脚本后,按键盘 Ctrl+C 无法终止的脚本
# 使用 trap 命令可以拦截用户通过键盘或 kill 命令发送过来的信号
# 使用 kill -l 可以查看 Linux 系统中所有的信号列表,其中 2 代表 Ctrl+C
# trap 当发现有用户 ctrl+C 希望终端脚本时,就执行 echo "暂停 10s";sleep 10 这两条命令
# 另外用户使用命令:[ kill -2 脚本的 PID ] 也可以中断脚本和 Ctrl+C 一样的效果,都会被 trap 拦截
trap 'echo "暂停 10s";sleep 10' 2
while :
do
  echo "go go go"
done
```

## 88）一键部署 memcached

```
#!/bin/bash

# 一键部署 memcached
```

```
# 脚本用源码来安装 memcached 服务器
# 注意:如果软件的下载链接过期了,请更新 memcached 的下载链接
wget http://www.memcached.org/files/memcached-1.5.1.tar.gz
yum -y install gcc
tar -xf  memcached-1.5.1.tar.gz
cd memcached-1.5.1
./configure
make
make install
```

## 89）一键配置 VNC 远程桌面服务器(无密码版本)

```
#!/bin/bash

# 一键配置 VNC 远程桌面服务器(无密码版本)

# 脚本配置的 VNC 服务器,客户端无需密码即可连接
# 客户端仅有查看远程桌面的权限,没有鼠标和键盘的操作权限

rpm --quiet -q tigervnc-server
if [  $? -ne  0 ];then
  yum  -y  tigervnc-server
fi
x0vncserver AcceptKeyEvents=0 AlwaysShared=1 \
AcceptPointerEvents=0 SecurityTypes=None  rfbport=5908
```

## 90）关闭 SELinux

```
#!/bin/bash

# 关闭 SELinux

sed -i  '/^SELINUX/s/=.*/=disabled/' /etc/selinux/config
setenforce 0
```

## 91）查看所有虚拟机磁盘使用量以及CPU使用量信息

```
#!/bin/bash

# 查看所有虚拟机磁盘使用量以及CPU使用量信息

virt-df
read -n1 "按任意键继续" key
virt-top
```

## 92）使用 shell 脚本打印图形

```
```

```
#!/bin/bash

# 使用 shell 脚本打印如下图形:
# 打印第一组图片
# for(())为类 C 语言的语法格式,也可以使用 for i  in;do  ;done 的格式替换
# for((i=1;i<=9;i++))循环会执行 9 次,i 从 1 开始到 9,每循环一次 i 自加 1
clear
for (( i=1; i<=9; i++ ))
do
  for (( j=1; j<=i; j++ ))
  do
    echo -n "$i"
  done
  echo ""
done
read  -n1  "按任意键继续"  key
#打印第二组图片
clear
for (( i=1; i<=5; i++ ))
do
  for (( j=1; j<=i; j++ ))
  do
    echo -n " |"
  done
  echo "_ "
done
read  -n1  "按任意键继续"  key
#打印第三组图片
clear
for (( i=1; i<=5; i++ ))
do
  for (( j=1; j<=i; j++ ))
  do
    echo -n " *"
  done
  echo ""
done
for (( i=5; i>=1; i-- ))
do
  for (( j=1; j<=i; j++ ))
  do
    echo -n " *"
  done
  echo ""
done
```

**93**）根据计算机当前时间,返回问候语,可以将该脚本设置为开机启动

```
#!/bin/bash

# 根据计算机当前时间,返回问候语,可以将该脚本设置为开机启动

# 00-12 点为早晨,12-18 点为下午,18-24 点为晚上
# 使用 date 命令获取时间后,if 判断时间的区间,确定问候语内容
tm=$(date +%H)
```

```
if [ $tm -le 12 ];then
  msg="Good Morning $USER"
elif [ $tm -gt 12 -a $tm -le 18 ];then
    msg="Good Afternoon $USER"
else
    msg="Good Night $USER"
fi
echo "当前时间是:$(date +"%Y-%m-%d %H:%M:%S")"
echo -e "\033[34m$msg\033[0m"
```

## 94）读取用户输入的账户名称,将账户名写入到数组保存

```
#!/bin/bash

# 读取用户输入的账户名称,将账户名写入到数组保存
# 定义数组名称为 name,数组的下标为 i,小标从 0 开始,每输入一个账户名,下标加 1,继续存下一个账户
# 最后,输入 over,脚本输出总结性信息后脚本退出
i=0
while :
do
  read -p "请输入账户名,输入 over 结束:" key
  if [ $key == "over" ];then
    break
    else
    name[$i]=$key
    let i++
    fi
done
echo "总账户名数量:${#name[*]}"
echo "${name[@]}"
```

## 95）判断文件或目录是否存在

```
#!/bin/bash

# 判断文件或目录是否存在
if [ $# -eq 0 ] ;then
echo "未输入任何参数,请输入参数"
echo "用法:$0 [文件名|目录名]"
fi
if [ -f $1 ];then
  echo "该文件,存在"
  ls -l $1
else
  echo "没有该文件"
fi
if [ -d  $1 ];then
     echo "该目录,存在"
     ls -ld  $2
else
     echo "没有该目录"
fi
```

## 96）打印各种格式的时间

```
#!/bin/bash

# 打印各种时间格式

echo "显示星期简称(如:Sun)"
date +%a
echo "显示星期全称(如:Sunday)"
date +%A
echo "显示月份简称(如:Jan)"
date +%b
echo "显示月份全称(如:January)"
date +%B
echo "显示数字月份(如:12)"
date +%m
echo "显示数字日期(如:01 号)"
date +%d
echo "显示数字年(如:01 号)"
date +%Y echo "显示年-月-日"
date +%F
echo "显示小时(24 小时制)"
date +%H
echo "显示分钟(00..59)"
date +%M
echo "显示秒"
date +%S
echo "显示纳秒"
date +%N
echo "组合显示"
date +"%Y%m%d %H:%M:%S"
```

## 97）使用 **egrep** 过滤 **MAC** 地址

```
#!/bin/bash

# 使用 egrep 过滤 MAC 地址

# MAC 地址由 16 进制组成,如 AA:BB:CC:DD:EE:FF
# [0-9a-fA-F]{2}表示一段十六进制数值,{5}表示连续出现5组前置:的十六进制
egrep "[0-9a-fA-F]{2}(:[0-9a-fA-F]{2}){5}" $1
```

## 98）统计双色球各个数字的中奖概率

```
#!/bin/bash

# 统计双色球各个数字的中奖概率

# 往期双色球中奖号码如下:
```

```
# 01 04 11 28 31 32  16
# 04 07 08 18 23 24  02
# 02 05 06 16 28 29  04
# 04 19 22 27 30 33  01
# 05 10 18 19 30 31  03
# 02 06 11 12 19 29  06
# 统计篮球和红球数据出现的概率次数(篮球不分顺序,统计所有篮球混合在一起的概率)
awk '{print $1"\n"$2"\n"$3"\n"$4"\n"$5"\n"$6}' 1.txt | sort | uniq -c | sort
awk '{print $7}' 1.txt | sort | uniq -c | sort
```

## 99）生成签名私钥和证书

```
#!/bin/bash

# 生成签名私钥和证书

read -p "请输入存放证书的目录:" dir
if [ ! -d $dir ];then
  echo "该目录不存在"
  exit
fi
read -p "请输入密钥名称:" name
# 使用 openssl 生成私钥
openssl genrsa -out ${dir}/${name}.key
# 使用 openssl 生成证书 #subj 选项可以在生成证书时,非交互自动填写 Common Name 信息
openssl req -new -x509 -key ${dir}/${name}.key -subj "/CN=common" -out ${dir}/${name}.crt
```

## 100）使用awk编写的wc程序

```
#!/bin/bash

# 使用awk编写的wc程序

# 自定义变量 chars 变量存储字符个数,自定义变量 words 变量存储单词个数
# awk 内置变量 NR 存储行数
# length()为 awk 内置函数,用来统计每行的字符数量,因为每行都会有一个隐藏的$,所以每次统计后都+1
# wc 程序会把文件结尾符$也统计在内,可以使用 cat -A 文件名,查看该隐藏字符
awk '{chars+=length($0)+1;words+=NF} END{print NR,words,chars}' $1
```

作者丨baiduoWang
来源丨https://blog.csdn.net/yugemengjing/article/details/82469785

公众号

欢迎大家加我个人微信

有关大数据、企业中遇到问题我们在群内一起讨论

长按上方扫码二维码，加我微信备注加群，拉你进群