Shell脚本,就是利用Shell的命令解释的功能，对一个纯文本的文件进行解析，然后执行这些功能，也可以说Shell脚本就是一系列命令的集合。

Shell可以直接使用在win/Unix/Linux上面，并且可以调用大量系统内部的功能来解释执行程序，如果熟练掌握Shell脚本，可以让我们操作计算机变得更加轻松，也会节省很多时间。

本篇文档整理了来自网络的250个shell脚本，希望对大家有所帮助。

> 文档整理：微信公众号【程序员面试吧】
> 文档内容来自网络，仅作免费交流分享

1.Dos 攻击防范（自动屏蔽攻击 IP）

```
#!/bin/bash
DATE=$(date +%d/%b/%Y:%H:%M)
LOG_FILE=/usr/local/nginx/logs/demo2.access.log
ABNORMAL_IP=$(tail -n5000 $LOG_FILE |grep $DATE |awk '{a[$1]++}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 $IP -j DROP
        echo "$(date +'%F_%T') $IP" >> /tmp/drop_ip.log
    fi
done
```

2.Linux 系统发送告警脚本

```
# yum install mailx
# vi /etc/mail.rc
set from=baojingtongzhi@163.com smtp=smtp.163.com
set smtp-auth-user=baojingtongzhi@163.com smtp-auth-password=123456
set smtp-auth=login
```

3.MySQL 数据库备份单循环

```
#!/bin/bash
DATE=$(date +%F_%H-%M-%S)
HOST=localhost
USER=backup
PASS=123.com
BACKUP_DIR=/data/db_backup
DB_LIST=$(mysql -h$HOST -u$USER -p$PASS -s -e "show databases;" 2>/dev/null
|egrep -v "Database|information_schema|mysql|performance_schema|sys")

for DB in $DB_LIST; do
    BACKUP_NAME=$BACKUP_DIR/${DB}_${DATE}.sql
    if ! mysqldump -h$HOST -u$USER -p$PASS -B $DB > $BACKUP_NAME 2>/dev/null;
then
        echo "$BACKUP_NAME 备份失败!"
    fi
done
```

4.MySQL 数据库备份多循环

```
#!/bin/bash
DATE=$(date +%F_%H-%M-%S)
HOST=localhost
USER=backup
PASS=123.com
BACKUP_DIR=/data/db_backup
DB_LIST=$(mysql -h$HOST -u$USER -p$PASS -s -e "show databases;" 2>/dev/null
|egrep -v "Database|information_schema|mysql|performance_schema|sys")

for DB in $DB_LIST; do
    BACKUP_DB_DIR=$BACKUP_DIR/${DB}_${DATE}
    [ ! -d $BACKUP_DB_DIR ] && mkdir -p $BACKUP_DB_DIR &>/dev/null
    TABLE_LIST=$(mysql -h$HOST -u$USER -p$PASS -s -e "use $DB;show tables;"
2>/dev/null)
    for TABLE in $TABLE_LIST; do
        BACKUP_NAME=$BACKUP_DB_DIR/${TABLE}.sql
        if ! mysqldump -h$HOST -u$USER -p$PASS $DB $TABLE > $BACKUP_NAME
2>/dev/null; then
            echo "$BACKUP_NAME 备份失败!"
        fi
    done
done
```

5.Nginx 访问访问日志按天切割

```
#!/bin/bash
LOG_DIR=/usr/local/nginx/logs
YESTERDAY_TIME=$(date -d "yesterday" +%F)
LOG_MONTH_DIR=$LOG_DIR/$(date +"%Y-%m")
LOG_FILE_LIST="default.access.log"

for LOG_FILE in $LOG_FILE_LIST; do
    [ ! -d $LOG_MONTH_DIR ] && mkdir -p $LOG_MONTH_DIR
    mv $LOG_DIR/$LOG_FILE $LOG_MONTH_DIR/${LOG_FILE}_${YESTERDAY_TIME}
done

kill -USR1 $(cat /var/run/nginx.pid)
```

6.Nginx 访问日志分析脚本

```
#!/bin/bash
# 日志格式: $remote_addr - $remote_user [$time_local] "$request" $status
$body_bytes_sent "$http_referer" "$http_user_agent" "$http_x_forwarded_for"
LOG_FILE=$1
echo "统计访问最多的10个IP"
awk '{a[$1]++}END{print "UV:",length(a);for(v in a)print v,a[v]}' $LOG_FILE |sort
-k2 -nr |head -10
echo "----------------------"

echo "统计时间段访问最多的IP"
awk '$4>="[01/Dec/2018:13:20:25" && $4<="[27/Nov/2018:16:20:49"{a[$1]++}END{for(v
in a)print v,a[v]}' $LOG_FILE |sort -k2 -nr|head -10
echo "----------------------"

echo "统计访问最多的10个页面"
awk '{a[$7]++}END{print "PV:",length(a);for(v in a){if(a[v]>10)print v,a[v]}}'
$LOG_FILE |sort -k2 -nr
echo "----------------------"

echo "统计访问页面状态码数量"
awk '{a[$7" "$9]++}END{for(v in a){if(a[v]>5)print v,a[v]}}'
```

7.查看网卡实时流量脚本

```
#!/bin/bash
NIC=$1
echo -e " In ------ Out"
while true; do
    OLD_IN=$(awk '$0~"'$NIC'"{print $2}' /proc/net/dev)
    OLD_OUT=$(awk '$0~"'$NIC'"{print $10}' /proc/net/dev)
    sleep 1
    NEW_IN=$(awk  '$0~"'$NIC'"{print $2}' /proc/net/dev)
    NEW_OUT=$(awk '$0~"'$NIC'"{print $10}' /proc/net/dev)
    IN=$(printf "%.1f%s" "$((($NEW_IN-$OLD_IN)/1024))" "KB/s")
    OUT=$(printf "%.1f%s" "$((($NEW_OUT-$OLD_OUT)/1024))" "KB/s")
    echo "$IN $OUT"
    sleep 1
done
```

8.服务器系统配置初始化脚本

```
#/bin/bash
# 设置时区并同步时间
ln -s /usr/share/zoneinfo/Asia/Shanghai /etc/localtime
if ! crontab -l |grep ntpdate &>/dev/null ; then
    (echo "* 1 * * * ntpdate time.windows.com >/dev/null 2>&1";crontab -l)
|crontab
fi

# 禁用selinux
sed -i '/SELINUX/{s/permissive/disabled/}' /etc/selinux/config

# 关闭防火墙
if egrep "7.[0-9]" /etc/redhat-release &>/dev/null; then
    systemctl stop firewalld
    systemctl disable firewalld
elif egrep "6.[0-9]" /etc/redhat-release &>/dev/null; then
    service iptables stop
    chkconfig iptables off
fi

# 历史命令显示操作时间
if ! grep HISTTIMEFORMAT /etc/bashrc; then
    echo 'export HISTTIMEFORMAT="%F %T `whoami` "' >> /etc/bashrc
fi

# SSH超时时间
if ! grep "TMOUT=600" /etc/profile &>/dev/null; then
    echo "export TMOUT=600" >> /etc/profile
fi

# 禁止root远程登录
sed -i 's/#PermitRootLogin yes/PermitRootLogin no/' /etc/ssh/sshd_config

# 禁止定时任务向发送邮件
sed -i 's/^MAILTO=root/MAILTO=""/' /etc/crontab

# 设置最大打开文件数
if ! grep "* soft nofile 65535" /etc/security/limits.conf &>/dev/null; then
    cat >> /etc/security/limits.conf << EOF
    * soft nofile 65535
    * hard nofile 65535
EOF
fi

# 系统内核优化
cat >> /etc/sysctl.conf << EOF
net.ipv4.tcp_syncookies = 1
net.ipv4.tcp_max_tw_buckets = 20480
net.ipv4.tcp_max_syn_backlog = 20480
net.core.netdev_max_backlog = 262144
net.ipv4.tcp_fin_timeout = 20
EOF

# 减少SWAP使用
echo "0" > /proc/sys/vm/swappiness
```

```
# 安装系统性能分析工具及其他
yum install gcc make autoconf vim sysstat net-tools iostat if
```

9.监控 100 台服务器磁盘利用率脚本

```
#!/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 $NF,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
```

> 1-9来自微信公众号：我的小碗汤

10.并发从数台机器中获取 hostname，并记录返回信息花费的时长，重定向到一个文件 hostname.txt 中，在全部完成后输出花费时长最短的那台机器的 CPU 信息。

```
#!bin/bash


# 所以主机，以空格分隔
ALL_HOSTS=(IP 地址 IP 地址)

for host in ${ALL_HOSTS[*]}
do
{
    start_time=$(date +'%s')
    ssh $host "hostname" &>/dev/null
    sleep 2
    stop_time=$(date +'%s')
    time_consuming=$((stop_time-start_time))
    echo "$host: $time_consuming" >>hostname.txt
}&
done

wait

host=$(sort -n -k 2 hostname.txt | head -1 | awk -F':' '{print $1}')

ssh $host "top -b -n 1"
```

11.统计 /proc 目类下 Linux 进程相关数量信息，输出总进程数，running 进程数，stoped 进程数，sleeing 进程数，zombie 进程数。

输出所有 zombie 的进程到 zombie.txt 杀死所有 zombie 进程。

```
#!/bin/bash

ALL_PROCESS=$(ls /proc/ | egrep '[0-9]+')

running_count=0
stoped_count=0
sleeping_count=0
zombie_count=0

for pid in ${ALL_PROCESS[*]}
do
    test -f /proc/$pid/status && state=$(egrep "State" /proc/$pid/status | awk
'{print $2}')
    case "$state" in
        R)
            running_count=$((running_count+1))
        ;;
        T)
            stoped_count=$((stoped_count+1))
        ;;
        S)
            sleeping_count=$((sleeping_count+1))
        ;;
        Z)
            zombie_count=$((zombie_count+1))
            echo "$pid" >>zombie.txt
            kill -9 "$pid"
        ;;
    esac
done


echo -e "total:
$((running_count+stoped_count+sleeping_count+zombie_count))\nrunning:
$running_count\nstoped: $stoped_count\nsleeping: $sleeping_count\nzombie:
$zombie_count"
```

12.把当前目录（包含子目录）下所有后缀为 ".sh" 的文件后缀变更为 ".shell"，之后删除每个文件的第二行。

```
#!/bin/bash


ALL_SH_FILE=$(find . -type f -name "*.sh")
for file in ${ALL_SH_FILE[*]}
do
    filename=$(echo $file | awk -F'.sh' '{print $1}')
    new_filename="${filename}.shell"
    mv "$file" "$new_filename"
    sed -i '2d' "$new_filename"
done
```

13.判断目录 /tmp/jstack 是否存在，不存在则新建一个目录，若存在则删除目录下所有内容。

每隔 1 小时打印 inceptor server 的 jstack 信息，并以 jstack_${当前时间} 命名文件，每当目录下超过 10 个文件后，删除最旧的文件。

```
#!/bin/bash


DIRPATH='/tmp/jstack'
CURRENT_TIME=$(date +'%F'-'%H:%M:%S')

if [ ! -d "$DIRPATH" ];then
    mkdir "$DIRPATH"
else
    rm -rf "$DIRPATH"/*
fi

cd "$DIRPATH"

while true
do
    sleep 3600
    # 这里需要将inceptor改后自己的java进程名称
    pid=$(ps -ef | grep 'inceptor' | grep -v grep | awk '{print $2}')
    jstack $pid >> "jstack_${CURRENT_TIME}"
    dir_count=$(ls | wc -l)
    if [ "$dir_count" -gt 10 ];then
       rm -f $(ls -tr | head -1)
    fi
done
```

14.从 test.log 中截取当天的所有 gc 信息日志，并统计 gc 时间的平均值和时长最长的时间。

```
#!/bin/bash

awk '{print $2}' hive-server2.log | tr -d ':' | awk '{sum+=$1} END {print "avg:
", sum/NR}' >>capture_hive_log.log
awk '{print $2}' hive-server2.log | tr -d ':' | awk '{max = 0} {if ($1+0 > max+0)
max=$1} END {print "Max: ", max}'>>capture_hive_log.log
```

15.查找 80 端口请求数最高的前 20 个 IP 地址，判断中间最小的请求数是否大于 500，如大于 500，则输出系统活动情况报告到 alert.txt，如果没有，则在 600s 后重试，直到有输出为止。

```
#!/bin/bash


state="true"

while $state
do
    SMALL_REQUESTS=$(netstat -ant | awk -F'[ :]+' '/:22/{count[$4]++} END {for(ip
in count) print count[ip]}' | sort -n | head -20 | head -1)
    if [ "$SMALL_REQUESTS" -gt 500 ];then
        sar -A > alert.txt
        state="false"
    else
        sleep 6
        continue
    fi
done
```

16.将当前目录下大于 10K 的文件转移到 /tmp 目录，再按照文件大小顺序，从大到小输出文件名。

```
#!/bin/bash


# 目标目录
DIRPATH='/tmp'
# 查看目录
FILEPATH='.'

find "$FILEPATH" -size +10k -type f | xargs -i mv {} "$DIRPATH"
ls -lS "$DIRPATH" | awk '{if(NR>1) print $NF}'
```

17.企业微信告警

此脚本通过企业微信应用，进行微信告警，可用于 Zabbix 监控。

```
# -*- coding: utf-8 -*-


import requests
import json


class DLF:
    def __init__(self, corpid, corpsecret):
        self.url = "https://qyapi.weixin.qq.com/cgi-bin"
        self.corpid = corpid
        self.corpsecret = corpsecret
```

```
        self._token = self._get_token()

    def _get_token(self):
        '''
        获取企业微信API接口的access_token
        :return:
        '''
        token_url = self.url + "/gettoken?corpid=%s&corpsecret=%s" %(self.corpid,
self.corpsecret)
        try:
            res = requests.get(token_url).json()
            token = res['access_token']
            return token
        except Exception as e:
            return str(e)

    def _get_media_id(self, file_obj):
        get_media_url = self.url + "/media/upload?access_token=
{}&type=file".format(self._token)
        data = {"media": file_obj}

        try:
            res = requests.post(url=get_media_url, files=data)
            media_id = res.json()['media_id']
            return media_id
        except Exception as e:
            return str(e)

    def send_text(self, agentid, content, touser=None, toparty=None):
        send_msg_url = self.url + "/message/send?access_token=%s" %
(self._token)
        send_data = {
            "touser": touser,
            "toparty": toparty,
            "msgtype": "text",
            "agentid": agentid,
            "text": {
                "content": content
            }
        }

        try:
            res = requests.post(send_msg_url, data=json.dumps(send_data))
        except Exception as e:
            return str(e)

    def send_image(self, agentid, file_obj, touser=None, toparty=None):
        media_id = self._get_media_id(file_obj)
        send_msg_url = self.url + "/message/send?access_token=%s" %
(self._token)
        send_data = {
            "touser": touser,
            "toparty": toparty,
            "msgtype": "image",
            "agentid": agentid,
            "image": {
                "media_id": media_id
           }
```

```
        }

        try:
            res = requests.post(send_msg_url, data=json.dumps(send_data))
        except Exception as e:
            return str(e)
```

18.FTP 客户端

通过 ftplib 模块操作 ftp 服务器，进行上传下载等操作。

```
# -*- coding: utf-8 -*-

from ftplib import FTP
from os import path
import copy


class FTPClient:
    def __init__(self, host, user, passwd, port=21):
        self.host = host
        self.user = user
        self.passwd = passwd
        self.port = port
        self.res = {'status': True, 'msg': None}
        self._ftp = None
        self._login()

    def _login(self):
        '''
        登录FTP服务器
        :return: 连接或登录出现异常时返回错误信息
        '''
        try:
            self._ftp = FTP()
            self._ftp.connect(self.host, self.port, timeout=30)
            self._ftp.login(self.user, self.passwd)
        except Exception as e:
            return e

    def upload(self, localpath, remotepath=None):
        '''
        上传ftp文件
        :param localpath: local file path
        :param remotepath: remote file path
        :return:
        '''
        if not localpath: return 'Please select a local file. '
        # 读取本地文件
        # fp = open(localpath, 'rb')

        # 如果未传递远程文件路径，则上传到当前目录，文件名称同本地文件
        if not remotepath:
            remotepath = path.basename(localpath)

        # 上传文件
        self._ftp.storbinary('STOR ' + remotepath, localpath)
```

```
        # fp.close()

    def download(self, remotepath, localpath=None):
        '''
        localpath
        :param localpath: local file path
        :param remotepath: remote file path
        :return:
        '''

        if not remotepath: return 'Please select a remote file. '
        # 如果未传递本地文件路径，则下载到当前目录，文件名称同远程文件
        if not localpath:
            localpath = path.basename(remotepath)
        # 如果localpath是目录的话就和remotepath的basename拼接
        if path.isdir(localpath):
            localpath = path.join(localpath, path.basename(remotepath))

        # 写入本地文件
        fp = open(localpath, 'wb')

        # 下载文件
        self._ftp.retrbinary('RETR ' + remotepath, fp.write)
        fp.close()

    def nlst(self, dir='/'):
        '''
        查看目录下的内容
        :return: 以列表形式返回目录下的所有内容
        '''
        files_list = self._ftp.nlst(dir)
        return files_list

    def rmd(self, dir=None):
        '''
        删除目录
        :param dir: 目录名称
        :return: 执行结果
        '''
        if not dir: return 'Please input dirname'
        res = copy.deepcopy(self.res)
        try:
            del_d = self._ftp.rmd(dir)
            res['msg'] = del_d
        except Exception as e:
            res['status'] = False
            res['msg'] = str(e)

        return res

    def mkd(self, dir=None):
        '''
        创建目录
        :param dir: 目录名称
        :return: 执行结果
        '''
        if not dir: return 'Please input dirname'
        res = copy.deepcopy(self.res)
```

```
        try:
            mkd_d = self._ftp.mkd(dir)
            res['msg'] = mkd_d
        except Exception as e:
            res['status'] = False
            res['msg'] = str(e)

        return res

    def del_file(self, filename=None):
        '''
        删除文件
        :param filename: 文件名称
        :return: 执行结果
        '''
        if not filename: return 'Please input filename'
        res = copy.deepcopy(self.res)
        try:
            del_f = self._ftp.delete(filename)
            res['msg'] = del_f
        except Exception as e:
            res['status'] = False
            res['msg'] = str(e)

        return res

    def get_file_size(self, filenames=[]):
        '''
        获取文件大小,单位是字节
        判断文件类型
        :param filename: 文件名称
        :return: 执行结果
        '''
        if not filenames: return {'msg': 'This is an empty directory'}
        res_l = []
        for file in filenames:
            res_d = {}
            # 如果是目录或者文件不存在就会报错
            try:
                size = self._ftp.size(file)
                type = 'f'
            except:
                # 如果是路径的话size显示 - , file末尾加/ （/dir/）
                size = '-'
                type = 'd'
                file = file + '/'

            res_d['filename'] = file
            res_d['size'] = size
            res_d['type'] = type
            res_l.append(res_d)

        return res_l

    def rename(self, old_name=None, new_name=None):
        '''
        重命名
        :param old_name: 旧的文件或者目录名称
```

```
        :param new_name: 新的文件或者目录名称
        :return: 执行结果
        '''
        if not old_name or not new_name: return 'Please input old_name and
new_name'
        res = copy.deepcopy(self.res)
        try:
            rename_f = self._ftp.rename(old_name, new_name)
            res['msg'] = rename_f
        except Exception as e:
            res['status'] = False
            res['msg'] = str(e)

        return res

    def close(self):
        '''
        退出ftp连接
        :return:
        '''
        try:
            # 向服务器发送quit命令
            self._ftp.quit()
        except Exception:
            return 'No response from server'
        finally:
            # 客户端单方面关闭连接
            self._ftp.close()
```

19.SSH 客户端

此脚本仅用于通过 **key** 连接，如需要密码连接，简单修改下即可。

```
# -*- coding: utf-8 -*-

import paramiko

class SSHClient:
    def __init__(self, host, port, user, pkey):
        self.ssh_host = host
        self.ssh_port = port
        self.ssh_user = user
        self.private_key = paramiko.RSAKey.from_private_key_file(pkey)
        self.ssh = None
        self._connect()

    def _connect(self):
        self.ssh = paramiko.SSHClient()
        self.ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
        try:
            self.ssh.connect(hostname=self.ssh_host, port=self.ssh_port,
username=self.ssh_user, pkey=self.private_key, timeout=10)
        except:
            return 'ssh connect fail'

    def execute_command(self, command):
        stdin, stdout, stderr = self.ssh.exec_command(command)
```

```
        out = stdout.read()
        err = stderr.read()
        return out, err

    def close(self):
        self.ssh.close()
```

20.Saltstack 客户端

通过 api 对 Saltstack 服务端进行操作，执行命令。

```
#!/usr/bin/env python
# -*- coding:utf-8 -*-


import requests
import json
import copy


class SaltApi:
    """
    定义salt api接口的类
    初始化获得token
    """
    def __init__(self):
        self.url = "http://172.85.10.21:8000/"
        self.username = "saltapi"
        self.password = "saltapi"
        self.headers = {"Content-type": "application/json"}
        self.params = {'client': 'local', 'fun': None, 'tgt': None, 'arg': None}
        self.login_url = self.url + "login"
        self.login_params = {'username': self.username, 'password':
self.password, 'eauth': 'pam'}
        self.token = self.get_data(self.login_url, self.login_params)['token']
        self.headers['X-Auth-Token'] = self.token

    def get_data(self, url, params):
        '''
        请求url获取数据
        :param url: 请求的url地址
        :param params: 传递给url的参数
        :return: 请求的结果
        '''
        send_data = json.dumps(params)
        request = requests.post(url, data=send_data, headers=self.headers)
        response = request.json()
        result = dict(response)
        return result['return'][0]

    def get_auth_keys(self):
        '''
        获取所有已经认证的key
        :return:
        '''
        data = copy.deepcopy(self.params)
        data['client'] = 'wheel'
```

```
        data['fun'] = 'key.list_all'
        result = self.get_data(self.url, data)
        try:
            return result['data']['return']['minions']
        except Exception as e:
            return str(e)

    def get_grains(self, tgt, arg='id'):
        """
        获取系统基础信息
        :tgt: 目标主机
        :return:
        """
        data = copy.deepcopy(self.params)
        if tgt:
            data['tgt'] = tgt
        else:
            data['tgt'] = '*'
        data['fun'] = 'grains.item'
        data['arg'] = arg
        result = self.get_data(self.url, data)

        return result


    def execute_command(self, tgt, fun='cmd.run', arg=None, tgt_type='list',
salt_async=False):
        """
        执行saltstack 模块命令，类似于salt '*' cmd.run 'command'
        :param tgt: 目标主机
        :param fun: 模块方法 可为空
        :param arg: 传递参数 可为空
        :return: 执行结果
        """
        data = copy.deepcopy(self.params)

        if not tgt: return {'status': False, 'msg': 'target host not exist'}
        if not arg:
            data.pop('arg')
        else:
            data['arg'] = arg
        if tgt != '*':
            data['tgt_type'] = tgt_type
        if salt_async: data['client'] = 'local_async'
        data['fun'] = fun
        data['tgt'] = tgt
        result = self.get_data(self.url, data)

        return result


    def jobs(self, fun='detail', jid=None):
        """
        任务
        :param fun: active, detail
        :param jod: Job ID
        :return: 任务执行结果
        """
```

```
        data = {'client': 'runner'}
        data['fun'] = fun
        if fun == 'detail':
            if not jid: return {'success': False, 'msg': 'job id is none'}
            data['fun'] = 'jobs.lookup_jid'
            data['jid'] = jid
        else:
            return {'success': False, 'msg': 'fun is active or detail'}
        result = self.get_data(self.url, data)

        return result
```

21.vCenter 客户端

通过官方 SDK 对 vCenter 进行日常操作，此脚本是我用于 cmdb 平台的，自动获取主机信息，存入数据库。

```
from pyVim.connect import SmartConnect, Disconnect, SmartConnectNoSSL
from pyVmomi import vim
from asset import models
import atexit


class Vmware:
    def __init__(self, ip, user, password, port, idc, vcenter_id):
        self.ip = ip
        self.user = user
        self.password = password
        self.port = port
        self.idc_id = idc
        self.vcenter_id = vcenter_id

    def get_obj(self, content, vimtype, name=None):
        '''
        列表返回,name 可以指定匹配的对象
        '''
        container = content.viewManager.CreateContainerView(content.rootFolder,
vimtype, True)
        obj = [ view for view in container.view ]
        return obj

    def get_esxi_info(self):
        # 宿主机信息
        esxi_host = {}
        res = {"connect_status": True, "msg": None}

        try:
            # connect this thing
            si = SmartConnectNoSSL(host=self.ip, user=self.user,
pwd=self.password, port=self.port, connectionPoolTimeout=60)
        except Exception as e:
            res['connect_status'] = False
            try:
                res['msg'] = ("%s Caught vmodl fault : " + e.msg) % (self.ip)
```

```
            except Exception as e:
                res['msg'] = '%s: connection error' % (self.ip)
            return res
        # disconnect this thing
        atexit.register(Disconnect, si)
        content = si.RetrieveContent()
        esxi_obj = self.get_obj(content, [vim.HostSystem])

        for esxi in esxi_obj:
            esxi_host[esxi.name] = {}
            esxi_host[esxi.name]['idc_id'] = self.idc_id
            esxi_host[esxi.name]['vcenter_id'] = self.vcenter_id
            esxi_host[esxi.name]['server_ip'] = esxi.name
            esxi_host[esxi.name]['manufacturer'] = esxi.summary.hardware.vendor
            esxi_host[esxi.name]['server_model'] = esxi.summary.hardware.model

            for i in esxi.summary.hardware.otherIdentifyingInfo:
                if isinstance(i, vim.host.SystemIdentificationInfo):
                    esxi_host[esxi.name]['server_sn'] = i.identifierValue

            # 系统名称
            esxi_host[esxi.name]['system_name'] =
esxi.summary.config.product.fullName
            # cpu总核数
            esxi_cpu_total = esxi.summary.hardware.numCpuThreads
            # 内存总量 GB
            esxi_memory_total = esxi.summary.hardware.memorySize / 1024 / 1024 /
1024

            # 获取硬盘总量 GB
            esxi_disk_total = 0
            for ds in esxi.datastore:
                esxi_disk_total += ds.summary.capacity / 1024 / 1024 / 1024

            # 默认配置4核8G100G，根据这个配置计算剩余可分配虚拟机
            default_configure = {
                'cpu': 4,
                'memory': 8,
                'disk': 100
            }

            esxi_host[esxi.name]['vm_host'] = []
            vm_usage_total_cpu = 0
            vm_usage_total_memory = 0
            vm_usage_total_disk = 0

            # 虚拟机信息
            for vm in esxi.vm:
                host_info = {}
                host_info['vm_name'] = vm.name
                host_info['power_status'] = vm.runtime.powerState
                host_info['cpu_total_kernel'] = str(vm.config.hardware.numCPU) +
'核'
                host_info['memory_total'] = str(vm.config.hardware.memoryMB) +
'MB'
                host_info['system_info'] = vm.config.guestFullName

                disk_info = ''
```

```
                disk_total = 0
                for d in vm.config.hardware.device:
                    if isinstance(d, vim.vm.device.VirtualDisk):
                        disk_total += d.capacityInKB / 1024 / 1024
                        disk_info += d.deviceInfo.label + ": " +
str((d.capacityInKB) / 1024 / 1024) + ' GB' + ','

                host_info['disk_info'] = disk_info
                esxi_host[esxi.name]['vm_host'].append(host_info)

                # 计算当前宿主机可用容量：总量 - 已分配的
                if host_info['power_status'] == 'poweredOn':
                    vm_usage_total_cpu += vm.config.hardware.numCPU
                    vm_usage_total_disk += disk_total
                    vm_usage_total_memory += (vm.config.hardware.memoryMB /
1024)

            esxi_cpu_free = esxi_cpu_total - vm_usage_total_cpu
            esxi_memory_free = esxi_memory_total - vm_usage_total_memory
            esxi_disk_free = esxi_disk_total - vm_usage_total_disk

            esxi_host[esxi.name]['cpu_info'] = 'Total: %d核, Free: %d核' %
(esxi_cpu_total, esxi_cpu_free)
            esxi_host[esxi.name]['memory_info'] = 'Total: %dGB, Free: %dGB' %
(esxi_memory_total, esxi_memory_free)
            esxi_host[esxi.name]['disk_info'] = 'Total: %dGB, Free: %dGB' %
(esxi_disk_total, esxi_disk_free)

            # 计算cpu 内存 磁盘按照默认资源分配的最小值，即为当前可分配资源
            if esxi_cpu_free < 4 or esxi_memory_free < 8 or esxi_disk_free <
100:
                free_allocation_vm_host = 0
            else:
                free_allocation_vm_host = int(min(
                    [
                        esxi_cpu_free / default_configure['cpu'],
                        esxi_memory_free / default_configure['memory'],
                        esxi_disk_free / default_configure['disk']
                    ]
                ))
            esxi_host[esxi.name]['free_allocation_vm_host'] =
free_allocation_vm_host
        esxi_host['connect_status'] = True
        return esxi_host

    def write_to_db(self):
        esxi_host = self.get_esxi_info()
        # 连接失败
        if not esxi_host['connect_status']:
            return esxi_host

        del esxi_host['connect_status']

        for machine_ip in esxi_host:
            # 物理机信息
            esxi_host_dict = esxi_host[machine_ip]
            # 虚拟机信息
            virtual_host = esxi_host[machine_ip]['vm_host']
```

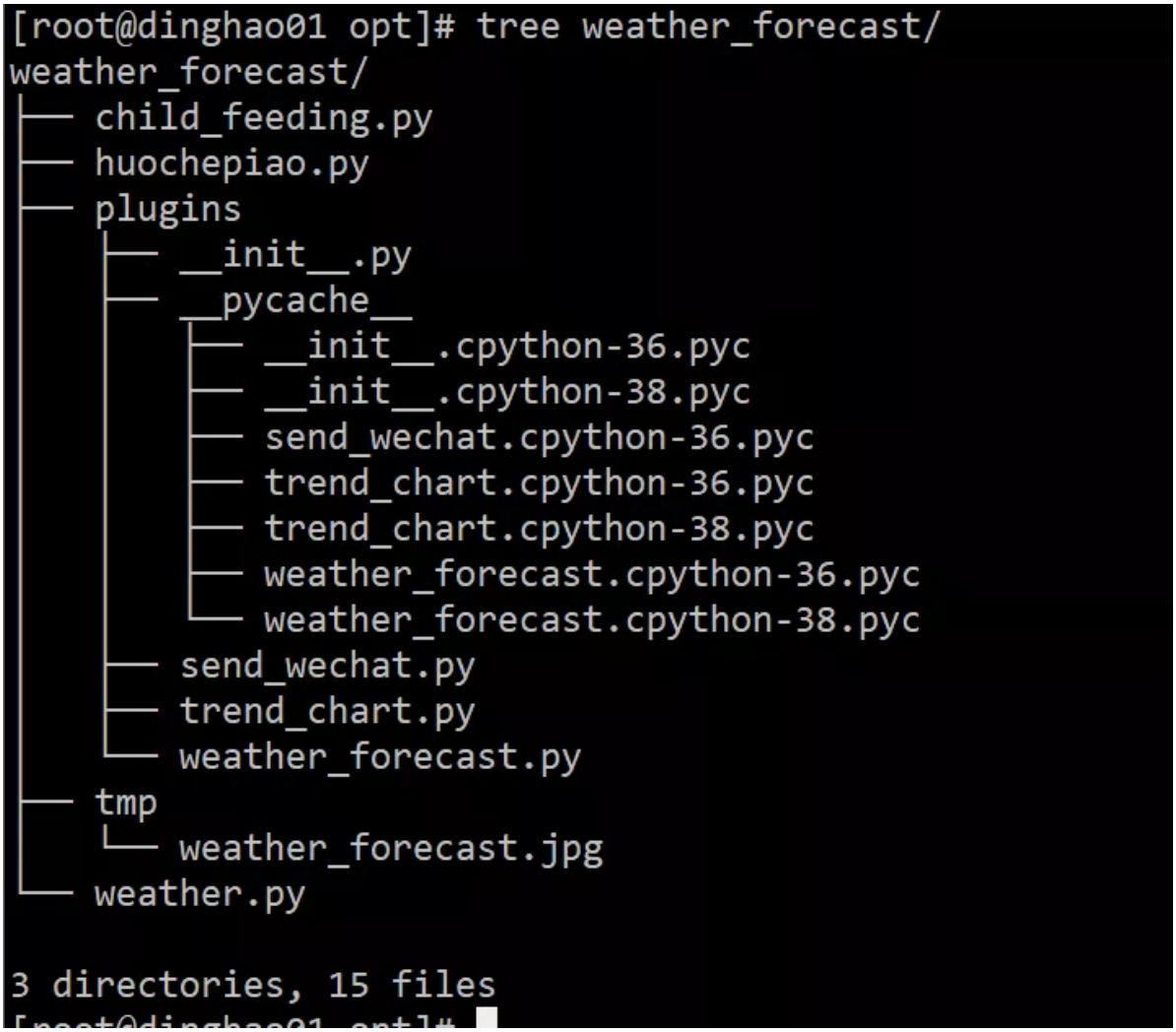
```
            del esxi_host[machine_ip]['vm_host']

            obj = models.EsxiHost.objects.create(**esxi_host_dict)
            obj.save()

            for host_info in virtual_host:
                host_info['management_host_id'] = obj.id
                obj2 = models.virtualHost.objects.create(**host_info)
                obj2.save()
```

22.获取域名 ssl 证书过期时间

用于 zabbix 告警

```
import re
import sys
import time
import subprocess
from datetime import datetime
from io import StringIO

def main(domain):
    f = StringIO()
    comm = f"curl -Ivs https://{domain} --connect-timeout 10"

    result = subprocess.getstatusoutput(comm)
    f.write(result[1])

    try:
        m = re.search('start date: (.*?)\n.*?expire date: (.*?)\n.*?common name:
(.*?)\n.*?issuer: CN=(.*?)\n', f.getvalue(), re.S)
        start_date = m.group(1)
        expire_date = m.group(2)
        common_name = m.group(3)
        issuer = m.group(4)
    except Exception as e:
        return 999999999

    # time 字符串转时间数组
    start_date = time.strptime(start_date, "%b %d %H:%M:%S %Y GMT")
    start_date_st = time.strftime("%Y-%m-%d %H:%M:%S", start_date)
    # datetime 字符串转时间数组
    expire_date = datetime.strptime(expire_date, "%b %d %H:%M:%S %Y GMT")
    expire_date_st = datetime.strftime(expire_date,"%Y-%m-%d %H:%M:%S")

    # 剩余天数
    remaining = (expire_date-datetime.now()).days

    return remaining

if __name__ == "__main__":
    domain = sys.argv[1]
    remaining_days = main(domain)
    print(remaining_days)
```

23.发送今天的天气预报以及未来的天气趋势图



此脚本用于给老婆大人发送今天的天气预报以及未来的天气趋势图，现在微信把网页端禁止了，没法发送到微信了，我是通过企业微信进行通知的，需要把你老婆大人拉到企业微信，无兴趣的小伙伴跳过即可。

```
    # -*- coding: utf-8 -*-


    import requests
    import json
    import datetime

    def weather(city):
        url = "http://wthrcdn.etouch.cn/weather_mini?city=%s" % city

        try:
            data = requests.get(url).json()['data']
            city = data['city']
            ganmao = data['ganmao']

            today_weather = data['forecast'][0]
            res = "老婆今天是{}\n今天天气概况\n城市: {:<10}\n时间: {:<10}\n高温: {:
<10}\n低温: {:<10}\n风力: {:<10}\n风向: {:<10}\n天气: {:<10}\n\n稍后会发送近期温度趋势
图，请注意查看。\
            ".format(
                ganmao,
                city,
                datetime.datetime.now().strftime('%Y-%m-%d'),
                today_weather['high'].split()[1],
```

```
                today_weather['low'].split()[1],
                today_weather['fengli'].split('[')[2].split(']')[0],
                today_weather['fengxiang'],today_weather['type'],
            )

            return {"source_data": data, "res": res}
        except Exception as e:
            return str(e)
```
+ 获取天气预报趋势图
```python
# -*- coding: utf-8 -*-


import matplotlib.pyplot as plt
import re
import datetime


def Future_weather_states(forecast, save_path, day_num=5):
    '''
    展示未来的天气预报趋势图
    :param forecast: 天气预报预测的数据
    :param day_num: 未来几天
    :return: 趋势图
    '''
    future_forecast = forecast
    dict={}

    for i in range(day_num):
        data = []
        date = future_forecast[i]["date"]
        date = int(re.findall("\d+",date)[0])
        data.append(int(re.findall("\d+", future_forecast[i]["high"])[0]))
        data.append(int(re.findall("\d+", future_forecast[i]["low"])[0]))
        data.append(future_forecast[i]["type"])
        dict[date] = data

    data_list = sorted(dict.items())
    date=[]
    high_temperature = []
    low_temperature = []
    for each in data_list:
        date.append(each[0])
        high_temperature.append(each[1][0])
        low_temperature.append(each[1][1])
        fig = plt.plot(date,high_temperature,"r",date,low_temperature,"b")

    current_date = datetime.datetime.now().strftime('%Y-%m')
    plt.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False
    plt.xlabel(current_date)
    plt.ylabel("℃")
    plt.legend(["高温", "低温"])
    plt.xticks(date)
    plt.title("最近几天温度变化趋势")
    plt.savefig(save_path)
```

+ 发送到企业微信
```python
# -*- coding: utf-8 -*-


import requests
import json


class DLF:
    def __init__(self, corpid, corpsecret):
        self.url = "https://qyapi.weixin.qq.com/cgi-bin"
        self.corpid = corpid
        self.corpsecret = corpsecret
        self._token = self._get_token()

    def _get_token(self):
        '''
        获取企业微信API接口的access_token
        :return:
        '''
        token_url = self.url + "/gettoken?corpid=%s&corpsecret=%s" %
(self.corpid, self.corpsecret)
        try:
            res = requests.get(token_url).json()
            token = res['access_token']
            return token
        except Exception as e:
            return str(e)

    def _get_media_id(self, file_obj):
        get_media_url = self.url + "/media/upload?access_token=
{}&type=file".format(self._token)
        data = {"media": file_obj}

        try:
            res = requests.post(url=get_media_url, files=data)
            media_id = res.json()['media_id']
            return media_id
        except Exception as e:
            return str(e)

    def send_text(self, agentid, content, touser=None, toparty=None):
        send_msg_url = self.url + "/message/send?access_token=%s" %
(self._token)
        send_data = {
            "touser": touser,
            "toparty": toparty,
            "msgtype": "text",
            "agentid": agentid,
            "text": {
                "content": content
            }
        }

        try:
            res = requests.post(send_msg_url, data=json.dumps(send_data))
        except Exception as e:
```

```
                return str(e)

        def send_image(self, agentid, file_obj, touser=None, toparty=None):
            media_id = self._get_media_id(file_obj)
            send_msg_url = self.url + "/message/send?access_token=%s" %
(self._token)
            send_data = {
                "touser": touser,
                "toparty": toparty,
                "msgtype": "image",
                "agentid": agentid,
                "image": {
                    "media_id": media_id
               }
            }

            try:
                res = requests.post(send_msg_url, data=json.dumps(send_data))
            except Exception as e:
                return str(e)
```

+ main脚本

```
# -*- coding: utf-8 -*-


from plugins.weather_forecast import weather
from plugins.trend_chart import Future_weather_states
from plugins.send_wechat import DLF
import os


# 企业微信相关信息
corpid = "xxx"
corpsecret = "xxx"
agentid = "xxx"
# 天气预报趋势图保存路径
_path = os.path.dirname(os.path.abspath(__file__))
save_path = os.path.join(_path ,'./tmp/weather_forecast.jpg')

# 获取天气预报信息
content = weather("大兴")

# 发送文字消息
dlf = DLF(corpid, corpsecret)
dlf.send_text(agentid=agentid, content=content['res'], toparty='1')

# 生成天气预报趋势图
Future_weather_states(content['source_data']['forecast'], save_path)
# 发送图片消息
file_obj = open(save_path, 'rb')
dlf.send_image(agentid=agentid, toparty='1', file_obj=file_obj)
```

24.SVN 完整备份

通过 **hotcopy** 进行 SVN 完整备份，备份保留 7 天。

```
#!/bin/bash
# Filename   :  svn_backup_repos.sh
# Date       :  2020/12/14
# Author     :  JakeTian
# Email      :  JakeTian@***.com
# Crontab    :  59 23 * * * /bin/bash $BASE_PATH/svn_backup_repos.sh >/dev/null
2>&1
# Notes      :  将脚本加入crontab中，每天定时执行
# Description:  SVN完全备份


set -e

SRC_PATH="/opt/svndata"
DST_PATH="/data/svnbackup"
LOG_FILE="$DST_PATH/logs/svn_backup.log"
SVN_BACKUP_C="/bin/svnadmin hotcopy"
SVN_LOOK_C="/bin/svnlook youngest"
TODAY=$(date +'%F')
cd $SRC_PATH
ALL_REPOS=$(find ./ -maxdepth 1 -type d ! -name 'httpd' -a ! -name 'bak' | tr -d
'./')

# 创建备份目录，备份脚本日志目录
test -d $DST_PATH || mkdir -p $DST_PATH
test -d $DST_PATH/logs || mkdir $DST_PATH/logs
test -d $DST_PATH/$TODAY || mkdir $DST_PATH/$TODAY

# 备份repos文件
for repo in $ALL_REPOS
do
    $SVN_BACKUP_C $SRC_PATH/$repo $DST_PATH/$TODAY/$repo

    # 判断备份是否完成
    if $SVN_LOOK_C $DST_PATH/$TODAY/$repo;then
        echo "$TODAY: $repo Backup Success" >> $LOG_FILE
    else
        echo "$TODAY: $repo Backup Fail" >> $LOG_FILE
    fi
done

# # 备份用户密码文件和权限文件
cp -p authz access.conf $DST_PATH/$TODAY

# 日志文件转储
mv $LOG_FILE $LOG_FILE-$TODAY

# 删除七天前的备份
seven_days_ago=$(date -d "7 days ago" +'%F')
rm -rf $DST_PATH/$seven_days_ago
```

25.zabbix 监控用户密码过期

用于 Zabbix 监控 Linux 系统用户（shell 为 /bin/bash 和 /bin/sh）密码过期，密码有效期剩余 7 天触发加自动发现用户。

```
#!/bin/bash


diskarray=(`awk -F':' '$NF ~ /\/bin\/bash/||/\/bin\/sh/{print $1}' /etc/passwd`)
length=${#diskarray[@]}

printf "{\n"
printf  '\t'"\"data\":["
for ((i=0;i<$length;i++))
do
    printf '\n\t\t{'
    printf "\"{#USER_NAME}\":\"${diskarray[$i]}\"}"
    if [ $i -lt $[$length-1] ];then
            printf ','
    fi
done
printf  "\n\t]\n"
printf "}\n"
```

检查用户密码过期

```
#!/bin/bash

export LANG=en_US.UTF-8

SEVEN_DAYS_AGO=$(date -d '-7 day' +'%s')
user="$1"

# 将Sep 09, 2018格式的时间转换成unix时间
expires_date=$(sudo chage -l $user | awk -F':' '/Password expires/{print $NF}' |
sed -n 's/^ //p')
if [[ "$expires_date" != "never" ]];then
    expires_date=$(date -d "$expires_date" +'%s')

    if [ "$expires_date" -le "$SEVEN_DAYS_AGO" ];then
        echo "1"
    else
        echo "0"
    fi
else
    echo "0"
fi
```

26.构建本地YUM

通过 rsync 的方式同步 yum，通过 nginx 只做 http yum 站点；

但是 centos6 的镜像最近都不能用了，国内貌似都禁用了，如果找到合适的自行更换地址。

```
#!/bin/bash
# 更新yum镜像


RsyncCommand="rsync -rvutH -P --delete --delete-after --delay-updates --
bwlimit=1000"
DIR="/app/yumData"
LogDir="$DIR/logs"
```

```
Centos6Base="$DIR/Centos6/x86_64/Base"
Centos7Base="$DIR/Centos7/x86_64/Base"
Centos6Epel="$DIR/Centos6/x86_64/Epel"
Centos7Epel="$DIR/Centos7/x86_64/Epel"
Centos6Salt="$DIR/Centos6/x86_64/Salt"
Centos7Salt="$DIR/Centos7/x86_64/Salt"
Centos6Update="$DIR/Centos6/x86_64/Update"
Centos7Update="$DIR/Centos7/x86_64/Update"
Centos6Docker="$DIR/Centos6/x86_64/Docker"
Centos7Docker="$DIR/Centos7/x86_64/Docker"
Centos6Mysql5_7="$DIR/Centos6/x86_64/Mysql/Mysql5.7"
Centos7Mysql5_7="$DIR/Centos7/x86_64/Mysql/Mysql5.7"
Centos6Mysql8_0="$DIR/Centos6/x86_64/Mysql/Mysql8.0"
Centos7Mysql8_0="$DIR/Centos7/x86_64/Mysql/Mysql8.0"
MirrorDomain="rsync://rsync.mirrors.ustc.edu.cn"

# 目录不存在就创建
check_dir(){
    for dir in $*
    do
        test -d $dir || mkdir -p $dir
    done
}

# 检查rsync同步结果
check_rsync_status(){
    if [ $? -eq 0 ];then
        echo "rsync success" >> $1
    else
        echo "rsync fail" >> $1
    fi
}


check_dir $DIR $LogDir $Centos6Base $Centos7Base $Centos6Epel $Centos7Epel
$Centos6Salt $Centos7Salt $Centos6Update $Centos7Update $Centos6Docker
$Centos7Docker $Centos6Mysql5_7 $Centos7Mysql5_7 $Centos6Mysql8_0
$Centos7Mysql8_0


# Base yumrepo
#$RsyncCommand "$MirrorDomain"/repo/centos/6/os/x86_64/ $Centos6Base >>
"$LogDir/centos6Base.log" 2>&1
# check_rsync_status "$LogDir/centos6Base.log"
$RsyncCommand "$MirrorDomain"/repo/centos/7/os/x86_64/ $Centos7Base >>
"$LogDir/centos7Base.log" 2>&1
check_rsync_status "$LogDir/centos7Base.log"

# Epel yumrepo
# $RsyncCommand "$MirrorDomain"/repo/epel/6/x86_64/ $Centos6Epel >>
"$LogDir/centos6Epel.log" 2>&1
# check_rsync_status "$LogDir/centos6Epel.log"
$RsyncCommand "$MirrorDomain"/repo/epel/7/x86_64/ $Centos7Epel >>
"$LogDir/centos7Epel.log" 2>&1
check_rsync_status "$LogDir/centos7Epel.log"

# SaltStack yumrepo
```

```
# $RsyncCommand "$MirrorDomain"/repo/salt/yum/redhat/6/x86_64/ $Centos6Salt >>
"$LogDir/centos6Salt.log" 2>&1
# ln -s $Centos6Salt/archive/$(ls $Centos6Salt/archive | tail -1)
$Centos6Salt/latest
# check_rsync_status "$LogDir/centos6Salt.log"
$RsyncComman "$MirrorDomain"/repo/salt/yum/redhat/7/x86_64/ $Centos7Salt >>
"$LogDir/centos7Salt.log" 2>&1
check_rsync_status "$LogDir/centos7Salt.log"
# ln -s $Centos7Salt/archive/$(ls $Centos7Salt/archive | tail -1)
$Centos7Salt/latest

# Docker yumrepo
$RsyncCommand "$MirrorDomain"/repo/docker-ce/linux/centos/7/x86_64/stable/
$Centos7Docker >> "$LogDir/centos7Docker.log" 2>&1
check_rsync_status "$LogDir/centos7Docker.log"

# centos update yumrepo
# $RsyncCommand "$MirrorDomain"/repo/centos/6/updates/x86_64/ $Centos6Update >>
"$LogDir/centos6Update.log" 2>&1
# check_rsync_status "$LogDir/centos6Update.log"
$RsyncCommand "$MirrorDomain"/repo/centos/7/updates/x86_64/ $Centos7Update >>
"$LogDir/centos7Update.log" 2>&1
check_rsync_status "$LogDir/centos7Update.log"

# mysql 5.7 yumrepo
# $RsyncCommand "$MirrorDomain"/repo/mysql-repo/yum/mysql-5.7-
community/el/6/x86_64/ "$Centos6Mysql5_7" >> "$LogDir/centos6Mysql5.7.log" 2>&1
# check_rsync_status "$LogDir/centos6Mysql5.7.log"
$RsyncCommand "$MirrorDomain"/repo/mysql-repo/yum/mysql-5.7-
community/el/7/x86_64/ "$Centos7Mysql5_7" >> "$LogDir/centos7Mysql5.7.log" 2>&1
check_rsync_status "$LogDir/centos7Mysql5.7.log"

# mysql 8.0 yumrepo
# $RsyncCommand "$MirrorDomain"/repo/mysql-repo/yum/mysql-8.0-
community/el/6/x86_64/ "$Centos6Mysql8_0" >> "$LogDir/centos6Mysql8.0.log" 2>&1
# check_rsync_status "$LogDir/centos6Mysql8.0.log"
$RsyncCommand "$MirrorDomain"/repo/mysql-repo/yum/mysql-8.0-
community/el/7/x86_64/ "$Centos7Mysql8_0" >> "$LogDir/centos7Mysql8.0.log" 2>&1
check_rsync_status "$LogDir/centos7Mysql8.0.log"
```

> 10-26来自养乐多，转自公众号「杰哥的IT之旅」

27.备份当前日期文件

```
#!/bin/bash
#一月前
historyTime=$(date "+%Y-%m-%d %H" -d '1 month ago')
echo ${historyTime}
historyTimeStamp=$(date -d "$historyTime" +%s)
echo ${historyTimeStamp}

#一周前
$(date "+%Y-%m-%d %H" -d '7 day ago')

#本月一月一日
date_this_month=`date +%Y%m01`
```

```
#一天前
date_today=`date -d '1 day ago' +%Y%m%d`

#一小时前
$(date "+%Y-%m-%d %H" -d '-1 hours')
```

28.DOS攻击防范（自动屏蔽攻击IP）

```
#!/bin/bash
DATE=$(date +%d/%b/%Y:%H:%M)
#nginx日志
LOG_FILE=/usr/local/nginx/logs/demo2.access.log
#分析ip的访问情况
ABNORMAL_IP=$(tail -n5000 $LOG_FILE |grep $DATE |awk '{a[$1]++}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 $IP -j DROP
        echo "$(date +'%F_%T') $IP" >> /tmp/drop_ip.log
    fi
done
```

29.批量创建多少个用户并设置密码

```
#!/bin/bash
USER_LIST=$@
USER_FILE=./user.info
for USER in $USER_LIST;do
  if ! id $USER &>/dev/null; then
    PASS=$(echo $RANDOM |md5sum |cut -c 1-8)
    useradd $USER
    echo $PASS | passwd --stdin $USER &>/dev/null
    echo "$USER   $PASS" >> $USER_FILE
    echo "$USER User create successful."
  else
    echo "$USER User already exists!"
  fi
done
```

30.快速在Ubuntu 20.04上架设LAMP服务器及WordPress博客

详情见：https://www.linuxmi.com/ubuntu-20-04-lamp-wordpress.html

```
#/bin/sh

install_dir="/var/www/html"
#Creating Random WP Database Credenitals
db_name="wp`date +%s`"
db_user=$db_name
db_password=`date |md5sum |cut -c '1-12'`
sleep 1
mysqlrootpass=`date |md5sum |cut -c '1-12'`
sleep 1
```

```
#### Install Packages for https and mysql
apt -y install apache2
apt -y install mysql-server


#### Start http
rm /var/www/html/index.html
systemctl enable apache2
systemctl start apache2

#### Start mysql and set root password

systemctl enable mysql
systemctl start mysql

/usr/bin/mysql -e "USE mysql;"
/usr/bin/mysql -e "UPDATE user SET Password=PASSWORD($mysqlrootpass) WHERE
user='root';"
/usr/bin/mysql -e "FLUSH PRIVILEGES;"
touch /root/.my.cnf
chmod 640 /root/.my.cnf
echo "[client]">>/root/.my.cnf
echo "user=root">>/root/.my.cnf
echo "password="$mysqlrootpass>>/root/.my.cnf
####Install PHP
apt -y install php
apt -y php-mysql
apt -y php-gd

sed -i '0,/AllowOverride\ None/! {0,/AllowOverride\ None/ s/AllowOverride\
None/AllowOverride\ All/}' /etc/apache2/apache2.conf #Allow htaccess usage

systemctl restart apache2

####Download and extract latest WordPress Package
if test -f /tmp/latest.tar.gz
then
echo "WP is already downloaded."
else
echo "Downloading WordPress"
cd /tmp/ && wget "http://wordpress.org/latest.tar.gz";
fi

/bin/tar -C $install_dir -zxf /tmp/latest.tar.gz --strip-components=1
chown www-data: $install_dir -R

#### Create WP-config and set DB credentials
/bin/mv $install_dir/wp-config-sample.php $install_dir/wp-config.php

/bin/sed -i "s/database_name_here/$db_name/g" $install_dir/wp-config.php
/bin/sed -i "s/username_here/$db_user/g" $install_dir/wp-config.php
/bin/sed -i "s/password_here/$db_password/g" $install_dir/wp-config.php

cat << EOF >> $install_dir/wp-config.php
define('FS_METHOD', 'direct');
EOF
```

```
cat << EOF >> $install_dir/.htaccess
# BEGIN WordPress
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteBase /
RewriteRule ^index.php$ - [L]
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule . /index.php [L]
</IfModule>
# END WordPress
EOF

chown www-data: $install_dir -R

##### Set WP Salts
grep -A50 'table_prefix' $install_dir/wp-config.php > /tmp/wp-tmp-config
/bin/sed -i '/**#@/,/$p/d' $install_dir/wp-config.php
/usr/bin/lynx --dump -width 200 https://api.wordpress.org/secret-key/1.1/salt/ >>
$install_dir/wp-config.php
/bin/cat /tmp/wp-tmp-config >> $install_dir/wp-config.php && rm /tmp/wp-tmp-
config -f
/usr/bin/mysql -u root -e "CREATE DATABASE $db_name"
/usr/bin/mysql -u root -e "CREATE USER '$db_name'@'localhost' IDENTIFIED WITH
mysql_native_password BY '$db_password';"
/usr/bin/mysql -u root -e "GRANT ALL PRIVILEGES ON $db_name.* TO
'$db_user'@'localhost';"

######Display generated passwords to log file.
echo "Database Name: " $db_name
echo "Database User: " $db_user
echo "Database Password: " $db_password
echo "Mysql root password: " $mysqlrootpass
```

31.每天自动备份 MySQL 数据库

```
#!/bin/sh

# Database info
DB_USER="batsing"
DB_PASS="batsingpw"
DB_HOST="localhost"
DB_NAME="timepusher"

# 一些变量
BIN_DIR="/usr/bin"             #mysql bin路径
BCK_DIR="/mnt/mysqlBackup"     #备份文件目录
DATE=`date +%F`

# TODO
# /usr/bin/mysqldump --opt -ubatsing -pbatsingpw -hlocalhost timepusher >
/mnt/mysqlBackup/db_`date +%F`.sql
$BIN_DIR/mysqldump --opt -u$DB_USER -p$DB_PASS -h$DB_HOST $DB_NAME >
$BCK_DIR/db_$DATE.sql

#还原数据库
#用mysql-front导入前一天的 *.sql 文件即可恢复数据
```

32.MySQL 数据库备份单循环

```
#!/bin/bash
DATE=$(date +%F_%H-%M-%S)
HOST=localhost
USER=backup
PASS=123.com
BACKUP_DIR=/data/db_backup
DB_LIST=$(mysql -h$HOST -u$USER -p$PASS -s -e "show databases;" 2>/dev/null
|egrep -v "Database|information_schema|mysql|performance_schema|sys")

for DB in $DB_LIST; do
    BACKUP_NAME=$BACKUP_DIR/${DB}_${DATE}.sql
    if ! mysqldump -h$HOST -u$USER -p$PASS -B $DB > $BACKUP_NAME 2>/dev/null;
then
        echo "$BACKUP_NAME 备份失败!"
    fi
done
```

33.MySQL 数据库备份多循环

```
#!/bin/bash
DATE=$(date +%F_%H-%M-%S)
HOST=localhost
USER=backup
PASS=123.com
BACKUP_DIR=/data/db_backup
DB_LIST=$(mysql -h$HOST -u$USER -p$PASS -s -e "show databases;" 2>/dev/null
|egrep -v "Database|information_schema|mysql|performance_schema|sys")

for DB in $DB_LIST; do
    BACKUP_DB_DIR=$BACKUP_DIR/${DB}_${DATE}
    [ ! -d $BACKUP_DB_DIR ] && mkdir -p $BACKUP_DB_DIR &>/dev/null
    TABLE_LIST=$(mysql -h$HOST -u$USER -p$PASS -s -e "use $DB;show tables;"
2>/dev/null)
    for TABLE in $TABLE_LIST; do
        BACKUP_NAME=$BACKUP_DB_DIR/${TABLE}.sql
        if ! mysqldump -h$HOST -u$USER -p$PASS $DB $TABLE > $BACKUP_NAME
2>/dev/null; then
            echo "$BACKUP_NAME 备份失败!"
        fi
    done
done
```

34.Nginx 日志按要求切割

```
#!/bin/bash
#安装目录下日志文件
base_log_path='/usr/local/openresty/nginx/logs/access.log'
base_error_path='/usr/local/openresty/nginx/logs/error.log'

#需要保存的目录位置
log_path='/data_lytdev_dir/nginx/logs/'
```

```
#获取月份
log_month=$(date -d yesterday +"%Y%m")

#获取前一天日期 (第二天凌晨备份,即保存的日志就是当天时间的日志)
log_day=$(date -d yesterday +"%d")

#在指定位置创建文件夹
mkdir -p $log_path/$log_month

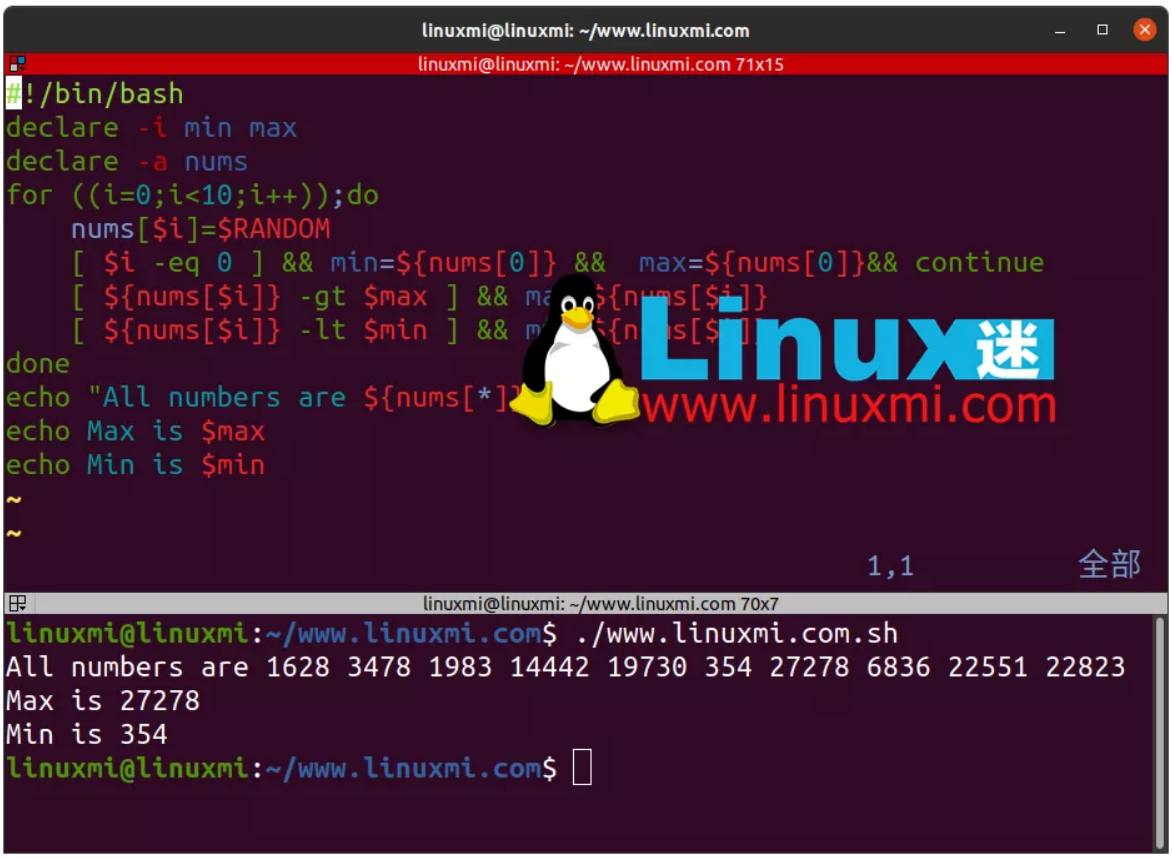
#将安装目录下的日志文件，移动到指定存储位置
mv $base_log_path $log_path/$log_month/access_$log_day.log
mv $base_error_path $log_path/$log_month/error_$log_day.log

#再使用信号控制切割日志
#USR1 表示nginx信号控制,切割日志
kill -USR1 `cat /usr/local/openresty/nginx/logs/nginx.pid`


#每天凌晨1点切割日志
* 1 * * * /usr/local/openresty/nginx/logs/log_rotate.sh
```

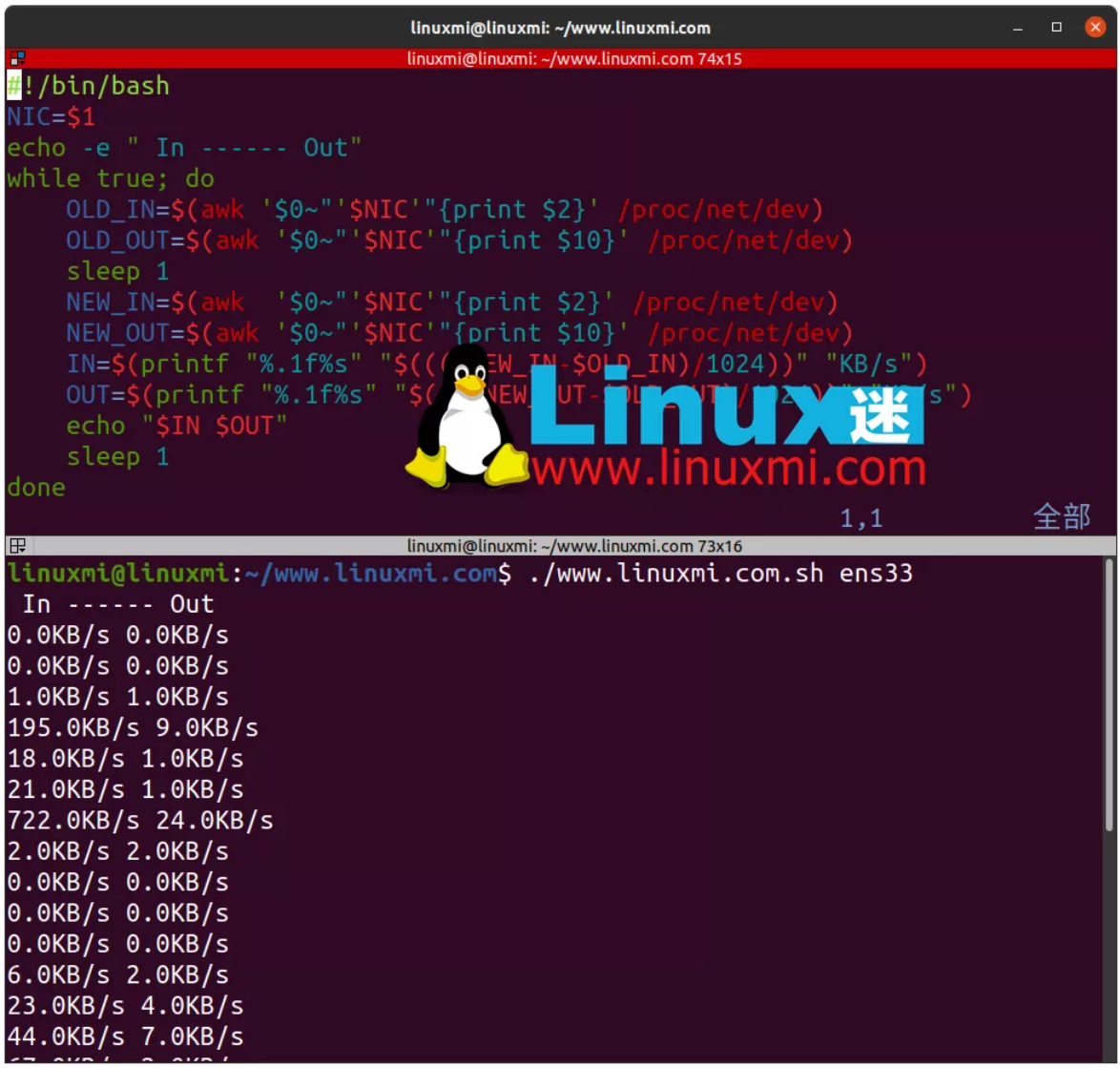
35.生成10个随机数保存于数组中并找出其最大值和最小值

```
#!/bin/bash
declare -i min max
declare -a nums
for ((i=0;i<10;i++));do
    nums[$i]=$RANDOM
    [ $i -eq 0 ] && min=${nums[0]} &&  max=${nums[0]}&& continue
    [ ${nums[$i]} -gt $max ] && max=${nums[$i]}
    [ ${nums[$i]} -lt $min ] && min=${nums[$i]}
done
echo "All numbers are ${nums[*]}"
echo Max is $max
echo Min is $min
```



36.查看网卡实时流量

```
#!/bin/bash
NIC=$1
echo -e " In ------ Out"
while true; do
    OLD_IN=$(awk '$0~"'$NIC'"{print $2}' /proc/net/dev)
    OLD_OUT=$(awk '$0~"'$NIC'"{print $10}' /proc/net/dev)
    sleep 1
    NEW_IN=$(awk  '$0~"'$NIC'"{print $2}' /proc/net/dev)
    NEW_OUT=$(awk '$0~"'$NIC'"{print $10}' /proc/net/dev)
    IN=$(printf "%.1f%s" "$((($NEW_IN-$OLD_IN)/1024))" "KB/s")
    OUT=$(printf "%.1f%s" "$((($NEW_OUT-$OLD_OUT)/1024))" "KB/s")
    echo "$IN $OUT"
    sleep 1
done
```



> 27-36来自：Linux迷，https://www.linuxmi.com/shell-script-10.html

37.服务器系统配置初始化

```
#/bin/bash
# 安装系统性能分析工具及其他
yum install gcc make autoconf vim sysstat net-tools iostat iftop iotp wget lrzsz
lsof unzip openssh-clients net-tool vim ntpdate -y
# 设置时区并同步时间
ln -s /usr/share/zoneinfo/Asia/Shanghai /etc/localtime
if ! crontab -l |grep ntpdate &>/dev/null ; then
    (echo "* 1 * * * ntpdate time.windows.com >/dev/null 2>&1";crontab -l)
|crontab
fi

# 禁用selinux
sed -i '/SELINUX/{s/permissive/disabled/}' /etc/selinux/config

# 关闭防火墙
if egrep "7.[0-9]" /etc/redhat-release &>/dev/null; then
    systemctl stop firewalld
    systemctl disable firewalld
elif egrep "6.[0-9]" /etc/redhat-release &>/dev/null; then
    service iptables stop
    chkconfig iptables off
```

```
fi

# 历史命令显示操作时间
if ! grep HISTTIMEFORMAT /etc/bashrc; then
    echo 'export HISTTIMEFORMAT="%Y-%m-%d %H:%M:%S  `whoami` "' >> /etc/bashrc
fi

# SSH超时时间
if ! grep "TMOUT=600" /etc/profile &>/dev/null; then
    echo "export TMOUT=600" >> /etc/profile
fi

# 禁止root远程登录 切记给系统添加普通用户，给su到root的权限
sed -i 's/#PermitRootLogin yes/PermitRootLogin no/' /etc/ssh/sshd_config

# 禁止定时任务向发送邮件
sed -i 's/^MAILTO=root/MAILTO=""/' /etc/crontab

# 设置最大打开文件数
if ! grep "* soft nofile 65535" /etc/security/limits.conf &>/dev/null; then
cat >> /etc/security/limits.conf << EOF
    * soft nofile 65535
    * hard nofile 65535
EOF
fi

# 系统内核优化
cat >> /etc/sysctl.conf << EOF
net.ipv4.tcp_syncookies = 1
net.ipv4.tcp_max_tw_buckets = 20480
net.ipv4.tcp_max_syn_backlog = 20480
net.core.netdev_max_backlog = 262144
net.ipv4.tcp_fin_timeout = 20
EOF

# 减少SWAP使用
echo "0" > /proc/sys/vm/swappiness
```

38.批量创建多个用户并设置密码

```
#!/bin/bash
USER_LIST=$@
USER_FILE=./user.info
for USER in $USER_LIST;do
 if ! id $USER &>/dev/null; then
  PASS=$(echo $RANDOM |md5sum |cut -c 1-8)
  useradd $USER
  echo $PASS | passwd --stdin $USER &>/dev/null
  echo "$USER   $PASS" >> $USER_FILE
  echo "$USER User create successful."
 else
  echo "$USER User already exists!"
 fi
done
```

39.一键查看服务器利用率

```
#!/bin/bash
function cpu(){

 util=$(vmstat | awk '{if(NR==3)print $13+$14}')
 iowait=$(vmstat | awk '{if(NR==3)print $16}')
 echo "CPU -使用率：${util}% ,等待磁盘IO相应使用率：${iowait}:${iowait}%"

}
function memory (){

 total=`free -m |awk '{if(NR==2)printf "%.1f",$2/1024}'`
    used=`free -m |awk '{if(NR==2) printf "%.1f",($2-$NF)/1024}'`
    available=`free -m |awk '{if(NR==2) printf "%.1f",$NF/1024}'`
    echo "内存 - 总大小: ${total}G , 使用: ${used}G , 剩余: ${available}G"
}
disk(){

 fs=$(df -h |awk '/^\/dev/{print $1}')
    for p in $fs; do
        mounted=$(df -h |awk '$1=="'$p'"{print $NF}')
        size=$(df -h |awk '$1=="'$p'"{print $2}')
        used=$(df -h |awk '$1=="'$p'"{print $3}')
        used_percent=$(df -h |awk '$1=="'$p'"{print $5}')
        echo "硬盘 - 挂载点: $mounted , 总大小: $size , 使用: $used , 使用率:
$used_percent"
    done

}
function tcp_status() {
    summary=$(ss -antp |awk '{status[$1]++}END{for(i in status) printf
i":"status[i]" "}')
    echo "TCP连接状态 - $summary"
}
cpu
memory
disk
tcp_status
```

40.找出占用CPU 内存过高的进程

```
#!/bin/bash
echo "-------------------CUP占用前10排序---------------------------------"
ps -eo user,pid,pcpu,pmem,args --sort=-pcpu  |head -n 10
echo "-------------------内存占用前10排序---------------------------------"
ps -eo user,pid,pcpu,pmem,args --sort=-pmem  |head -n 10
```

41.查看网卡的实时流量

```
#!/bin/bash
eth0=$1
echo  -e    "流量进入--流量传出     "
while true; do
 old_in=$(cat /proc/net/dev |grep $eth0 |awk '{print $2}')
 old_out=$(cat /proc/net/dev |grep $eth0 |awk '{print $10}')
 sleep 1
 new_in=$(cat /proc/net/dev |grep $eth0 |awk '{print $2}')
 new_out=$(cat /proc/net/dev |grep $eth0 |awk '{print $10}')
 in=$(printf "%.1f%s" "$((($new_in-$old_in)/1024))" "KB/s")
 out=$(printf "%.1f%s" "$((($new_out-$old_out)/1024))" "KB/s")
 echo "$in $out"
done
```

42.监控多台服务器磁盘利用率脚本

```
#!/bin/bash
HOST_INFO=host.info
for IP in $(awk '/^[^#]/{print $1}' $HOST_INFO); do
 #取出用户名和端口
    USER=$(awk -v ip=$IP 'ip==$1{print $2}' $HOST_INFO)
    PORT=$(awk -v ip=$IP 'ip==$1{print $3}' $HOST_INFO)
 #创建临时文件，保存信息
    TMP_FILE=/tmp/disk.tmp
 #通过公钥登录获取主机磁盘信息
    ssh -p $PORT $USER@$IP 'df -h' > $TMP_FILE
 #分析磁盘占用空间
    USE_RATE_LIST=$(awk 'BEGIN{OFS="="}/^\/dev/{print $NF,int($5)}' $TMP_FILE)
 #循环磁盘列表，进行判断
    for USE_RATE in $USE_RATE_LIST; do
  #取出等号（=）右边的值 挂载点名称
        PART_NAME=${USE_RATE%=*}
  #取出等号（=）左边的值  磁盘利用率
        USE_RATE=${USE_RATE#*=}
  #进行判断
        if [ $USE_RATE -ge 80 ]; then
            echo "Warning: $PART_NAME Partition usage $USE_RATE%!"
    echo "服务器$IP的磁盘空间占用过高，请及时处理" | mail -s "空间不足警告" 你的qq@qq.com
  else
   echo "服务器$IP的$PART_NAME目录空间良好"
        fi
    done
done
```

43.批量检测网站是否异常并邮件通知

```
#!/bin/bash
URL_LIST="www.baidu.com www.ctnrs.com www.der-matech.net.cn www.der-matech.com.cn
www.der-matech.cn www.der-matech.top www.der-matech.org"
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 -eq 200 ]; then
```

```
            echo "$URL OK"
            break
        else
            echo "$URL retry $FAIL_COUNT"
            let FAIL_COUNT++
        fi
    done
    if [ $FAIL_COUNT -eq 3 ]; then
        echo "Warning: $URL Access failure!"
  echo "网站$URL坏掉，请及时处理" | mail -s "$URL网站高危" 1794748404@qq.com
    fi
done
```

44.批量主机远程执行命令脚本

```
#!/bin/bash
COMMAND=$*
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}' $HOST_INFO)
    PASS=$(awk -v ip=$IP 'ip==$1{print $4}' $HOST_INFO)
    expect -c "
       spawn ssh -p $PORT $USER@$IP
       expect {
          \"(yes/no)\" {send \"yes\r\"; exp_continue}
          \"password:\" {send \"$PASS\r\"; exp_continue}
          \"$USER@*\" {send \"$COMMAND\r exit\r\"; exp_continue}
       }
    "
    echo "--------------------"
done
```

45.一键部署LNMP网站平台脚本

```
#!/bin/bash
NGINX_V=1.15.6
PHP_V=5.6.36
TMP_DIR=/tmp

INSTALL_DIR=/usr/local

PWD_C=$PWD

echo
echo -e "\tMenu\n"
echo -e "1. Install Nginx"
echo -e "2. Install PHP"
echo -e "3. Install MySQL"
echo -e "4. Deploy LNMP"
echo -e "9. Quit"

function command_status_check() {
 if [ $? -ne 0 ]; then
  echo $1
```

```
  exit
 fi
}

function install_nginx() {
    cd $TMP_DIR
    yum install -y gcc gcc-c++ make openssl-devel pcre-devel wget
    wget http://nginx.org/download/nginx-${NGINX_V}.tar.gz
    tar zxf nginx-${NGINX_V}.tar.gz
    cd nginx-${NGINX_V}
    ./configure --prefix=$INSTALL_DIR/nginx \
    --with-http_ssl_module \
    --with-http_stub_status_module \
    --with-stream
    command_status_check "Nginx - 平台环境检查失败！"
    make -j 4
    command_status_check "Nginx - 编译失败！"
    make install
    command_status_check "Nginx - 安装失败！"
    mkdir -p $INSTALL_DIR/nginx/conf/vhost
    alias cp=cp ; cp -rf $PWD_C/nginx.conf $INSTALL_DIR/nginx/conf
    rm -rf $INSTALL_DIR/nginx/html/*
    echo "ok" > $INSTALL_DIR/nginx/html/status.html
    echo '<?php echo "ok"?>' > $INSTALL_DIR/nginx/html/status.php
    $INSTALL_DIR/nginx/sbin/nginx
    command_status_check "Nginx - 启动失败！"
}

function install_php() {
 cd $TMP_DIR
    yum install -y gcc gcc-c++ make gd-devel libxml2-devel \
        libcurl-devel libjpeg-devel libpng-devel openssl-devel \
        libmcrypt-devel libxslt-devel libtidy-devel
    wget http://docs.php.net/distributions/php-${PHP_V}.tar.gz
    tar zxf php-${PHP_V}.tar.gz
    cd php-${PHP_V}
    ./configure --prefix=$INSTALL_DIR/php \
    --with-config-file-path=$INSTALL_DIR/php/etc \
    --enable-fpm --enable-opcache \
    --with-mysql --with-mysqli --with-pdo-mysql \
    --with-openssl --with-zlib --with-curl --with-gd \
    --with-jpeg-dir --with-png-dir --with-freetype-dir \
    --enable-mbstring --enable-hash
    command_status_check "PHP - 平台环境检查失败！"
    make -j 4
    command_status_check "PHP - 编译失败！"
    make install
    command_status_check "PHP - 安装失败！"
    cp php.ini-production $INSTALL_DIR/php/etc/php.ini
    cp sapi/fpm/php-fpm.conf $INSTALL_DIR/php/etc/php-fpm.conf
    cp sapi/fpm/init.d.php-fpm /etc/init.d/php-fpm
    chmod +x /etc/init.d/php-fpm
    /etc/init.d/php-fpm start
    command_status_check "PHP - 启动失败！"
}

read -p "请输入编号：" number
case $number in
```

```
    1)
        install_nginx;;
    2)
        install_php;;
    3)
        install_mysql;;
    4)
        install_nginx
        install_php
        ;;
    9)
        exit;;
esac
```

46.监控MySQL主从同步状态是否异常脚本

```
#!/bin/bash
HOST=localhost
USER=root
PASSWD=123.com
IO_SQL_STATUS=$(mysql -h$HOST -u$USER -p$PASSWD -e 'show slave status\G'
2>/dev/null |awk '/Slave_.*_Running:/{print $1$2}')
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!" |mail -s "Master-Slave Staus" xxx@163.com
    fi
done
```

47.MySql数据库备份脚本

分库备份

```
mysqldump -uroot -pxxx -B A > A.sql
#!/bin/bash
DATE=$(date +%F_%H-%M-%S)
HOST=localhost
USER=backup
PASS=123.com
BACKUP_DIR=/data/db_backup
DB_LIST=$(mysql -h$HOST -u$USER -p$PASS -s -e "show databases;" 2>/dev/null
|egrep -v "Database|information_schema|mysql|performance_schema|sys")

for DB in $DB_LIST; do
    BACKUP_NAME=$BACKUP_DIR/${DB}_${DATE}.sql
    if ! mysqldump -h$HOST -u$USER -p$PASS -B $DB > $BACKUP_NAME 2>/dev/null;
then
        echo "$BACKUP_NAME 备份失败!"
    fi
done
```

分表备份

```
mysqldump -uroot -pxxx -A t > t.sql
#!/bin/bash
DATE=$(date +%F_%H-%M-%S)
HOST=localhost
USER=backup
PASS=123.com
BACKUP_DIR=/data/db_backup
DB_LIST=$(mysql -h$HOST -u$USER -p$PASS -s -e "show databases;" 2>/dev/null
|egrep -v "Database|information_schema|mysql|performance_schema|sys")

for DB in $DB_LIST; do
    BACKUP_DB_DIR=$BACKUP_DIR/${DB}_${DATE}
    [ ! -d $BACKUP_DB_DIR ] && mkdir -p $BACKUP_DB_DIR &>/dev/null
    TABLE_LIST=$(mysql -h$HOST -u$USER -p$PASS -s -e "use $DB;show tables;"
2>/dev/null)
    for TABLE in $TABLE_LIST; do
        BACKUP_NAME=$BACKUP_DB_DIR/${TABLE}.sql
        if ! mysqldump -h$HOST -u$USER -p$PASS $DB $TABLE > $BACKUP_NAME
2>/dev/null; then
            echo "$BACKUP_NAME 备份失败!"
        fi
    done
done
```

48.Nginx访问日志分析

```
#!/bin/bash
# 日志格式: $remote_addr - $remote_user [$time_local] "$request" $status
$body_bytes_sent "$http_referer" "$http_user_agent" "$http_x_forwarded_for"
LOG_FILE=$1
echo "统计访问最多的10个IP"
awk '{a[$1]++}END{print "UV:",length(a);for(v in a)print v,a[v]}' $LOG_FILE |sort
-k2 -nr |head -10
echo "----------------------"

echo "统计时间段访问最多的IP"
awk '$4>="[01/Dec/2018:13:20:25" && $4<="[27/Nov/2018:16:20:49"{a[$1]++}END{for(v
in a)print v,a[v]}' $LOG_FILE |sort -k2 -nr|head -10
echo "----------------------"

echo "统计访问最多的10个页面"
awk '{a[$7]++}END{print "PV:",length(a);for(v in a){if(a[v]>10)print v,a[v]}}'
$LOG_FILE |sort -k2 -nr
echo "----------------------"

echo "统计访问页面状态码数量"
awk '{a[$7" "$9]++}END{for(v in a){if(a[v]>5)print v,a[v]}}' $LOG_FILE |sort -k3
-nr
```

49.Nginx访问日志自动按天（周、月）切割

```
#!/bin/bash
#nginx日志目录
```

```
LOG_DIR=/www/server/nginx/logs
#获取到上一天的时间
YESTERDAY_TIME=$(date -d "yesterday" +%F)
#归档日志取时间
LOG_MONTH_DIR=$LOG_DIR/$(date +"%Y-%m")
#归档日志的名称
LOG_FILE_LIST="access.log"

for LOG_FILE in $LOG_FILE_LIST; do
    [ ! -d $LOG_MONTH_DIR ] && mkdir -p $LOG_MONTH_DIR
    mv $LOG_DIR/$LOG_FILE $LOG_MONTH_DIR/${LOG_FILE}_${YESTERDAY_TIME}
done

kill -USR1 $(cat $LOG_DIR/nginx.pid)
```

50.自动发布Java项目（Tomcat）

```
#!/bin/bash
DATE=$(date +%F_%T)

TOMCAT_NAME=$1
TOMCAT_DIR=/usr/local/$TOMCAT_NAME
ROOT=$TOMCAT_DIR/webapps/ROOT

BACKUP_DIR=/data/backup
WORK_DIR=/tmp
PROJECT_NAME=tomcat-java-demo

# 拉取代码
cd $WORK_DIR
if [ ! -d $PROJECT_NAME ]; then
   git clone https://github.com/lizhenliang/tomcat-java-demo
   cd $PROJECT_NAME
else
   cd $PROJECT_NAME
   git pull
fi

# 构建
mvn clean package -Dmaven.test.skip=true
if [ $? -ne 0 ]; then
   echo "maven build failure!"
   exit 1
fi

# 部署
TOMCAT_PID=$(ps -ef |grep "$TOMCAT_NAME" |egrep -v "grep|$$" |awk 'NR==1{print
$2}')
[ -n "$TOMCAT_PID" ] && kill -9 $TOMCAT_PID
[ -d $ROOT ] && mv $ROOT $BACKUP_DIR/${TOMCAT_NAME}_ROOT$DATE
unzip $WORK_DIR/$PROJECT_NAME/target/*.war -d $ROOT
$TOMCAT_DIR/bin/startup.sh
```

51.自动发布PHP项目

```
#!/bin/bash

DATE=$(date +%F_%T)

WWWROOT=/usr/local/nginx/html/$1


BACKUP_DIR=/data/backup
WORK_DIR=/tmp
PROJECT_NAME=php-demo


# 拉取代码
cd $WORK_DIR
if [ ! -d $PROJECT_NAME ]; then
   git clone https://github.com/lizhenliang/php-demo
   cd $PROJECT_NAME
else
   cd $PROJECT_NAME
   git pull
fi


# 部署
if [ ! -d $WWWROOT ]; then
   mkdir -p $WWWROOT
   rsync -avz --exclude=.git $WORK_DIR/$PROJECT_NAME/* $WWWROOT
else
   rsync -avz --exclude=.git $WORK_DIR/$PROJECT_NAME/* $WWWROOT
fi
```

52.DOS攻击防范（自动屏蔽攻击IP）

```
#!/bin/bash
DATE=$(date +%d/%b/%Y:%H:%M)
#nginx日志
LOG_FILE=/usr/local/nginx/logs/demo2.access.log
#分析ip的访问情况
ABNORMAL_IP=$(tail -n5000 $LOG_FILE |grep $DATE |awk '{a[$1]++}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 $IP -j DROP
        echo "$(date +'%F_%T') $IP" >> /tmp/drop_ip.log
    fi
done
```

53.目录入侵检测与告警

```
#!/bin/bash

MON_DIR=/opt
inotifywait -mqr --format %f -e create $MON_DIR |\
while read files; do
   #同步文件
   rsync -avz /opt /tmp/opt
  #检测文件是否被修改
   #echo "$(date +'%F %T') create $files" | mail -s "dir monitor" xxx@163.com
done
```

> 37-53 作者：南宫乘风 链接：
>
> *blog.csdn.net/heian_99/article/details/104027379*

54.本地选择脚本auto_build.sh

脚本内容如下：

```
#/bin/bash
remote_ip=172.160.111.32
remote_hostname=lyn

case $1 in
    1)  echo -e '\033[0;42m Ethernet dhcp \033[0m'
        VAR="eno1"
    ;;
    2)  echo -e '\033[0;46m wireless dhcp \033[0m'
        VAR="wlo1"
    ;;
esac

HOST_IP=$(ifconfig $VAR | grep "inet" | grep -v inet6| awk '{ print $2}' | awk -
F: '{print $1}')
echo "parse ip is:" $HOST_IP
if [[ ! -n "${HOST_IP}" ]] ;then
 echo -e "\033[0;31m input local ip \033[0m"
 read local_ip
else
 if [[ ! $(echo "${HOST_IP}" | awk -F. '{printf $1}') == "192"  ]] ;then
  local_ip=$HOST_IP
 else
  echo -e "\033[0;31m error ip \033[0m"
  exit 0
 fi
fi
#exit 0
if [[ $2 == 1 ]];then
 build_opt="all_build"
else
 build_opt=
fi
ssh -t ${remote_hostname}@${remote_ip} "/home/lyn/build.sh ip=${local_ip}
${build_opt}"
```

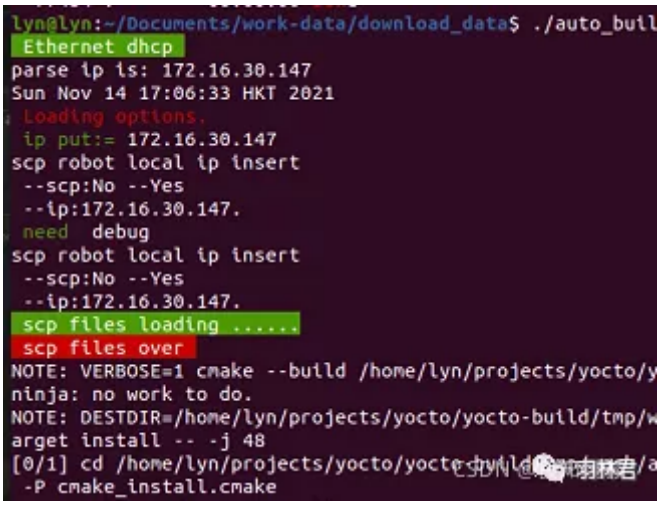
这个部分有几处技术使用：

switch case使用，if else、免密登陆，远程调用脚本。

首先是一个switch case

此处作用是进行ip地址的筛选，因为的在调试过程中，我的电脑有时候用网线连接，有时候会去测试房去测试，用wifi连接，这个时候会进行网络ip地址的区分，当我输入./auto_build.sh 1的时候，脚本会进行解析**eno1**网线分配的ip地址，当我输入./auto_build.sh 2的时候则会解析**wlo1**wifi分配的ip。

在里面我还用了颜色打印，进行关键词的标注，如下所示：



关于颜色打印的部分这个是另一个知识，这是一个转义的实际使用过程，通过特定符号的转义识别，我们在Linux终端去显示不同颜色的打印输出，这个是我们经常使用的操作，例如log等级分级打印时候，error是红色，正常是绿色，普通是白色等。

颜色打印大致介绍如下：

转义序列以控制字符'ESC'开头。该字符的ASCII码十进制表示为27，十六进制表示为0x1B，八进制表示为033。多数转义序列超过两个字符，故通常以'ESC'和左括号'['开头。该起始序列称为控制序列引导符(CSI，Control Sequence Intro)，通常由 '**\033[**' 或 '**\e[**' 代替。

通过转义序列设置终端显示属性时，可采用以下格式：

```
\033[ Param {;Param;...}m

或

\e[ Param {;Param;...}m
```

其中，'\033['或'\e['引导转义序列，'m'表示设置属性并结束转义序列。

因此，通过转义序列设置终端显示属性时，常见格式为：

```
\033[显示方式;前景色;背景色m输出字符串\033[0m

或\e[显示方式;前景色;背景色m输出字符串\033[0m
```

其中，'\033[0m'用于恢复默认的终端输出属性，否则会影响后续的输出。

示例：我在此处使用 echo -e '\033[0;42m Ethernet dhcp \033[0m' 进行网线端口ip分配的打印，通过转义之后，打印颜色为带背景色的绿色显示。具体对应的颜色，大家可以看一下小麦老兄写的这篇文章printf打印还能这么玩。

注：打印log时候记得echo 要使用 -e参数。

其次还有组合使用命令实现获取本地ip

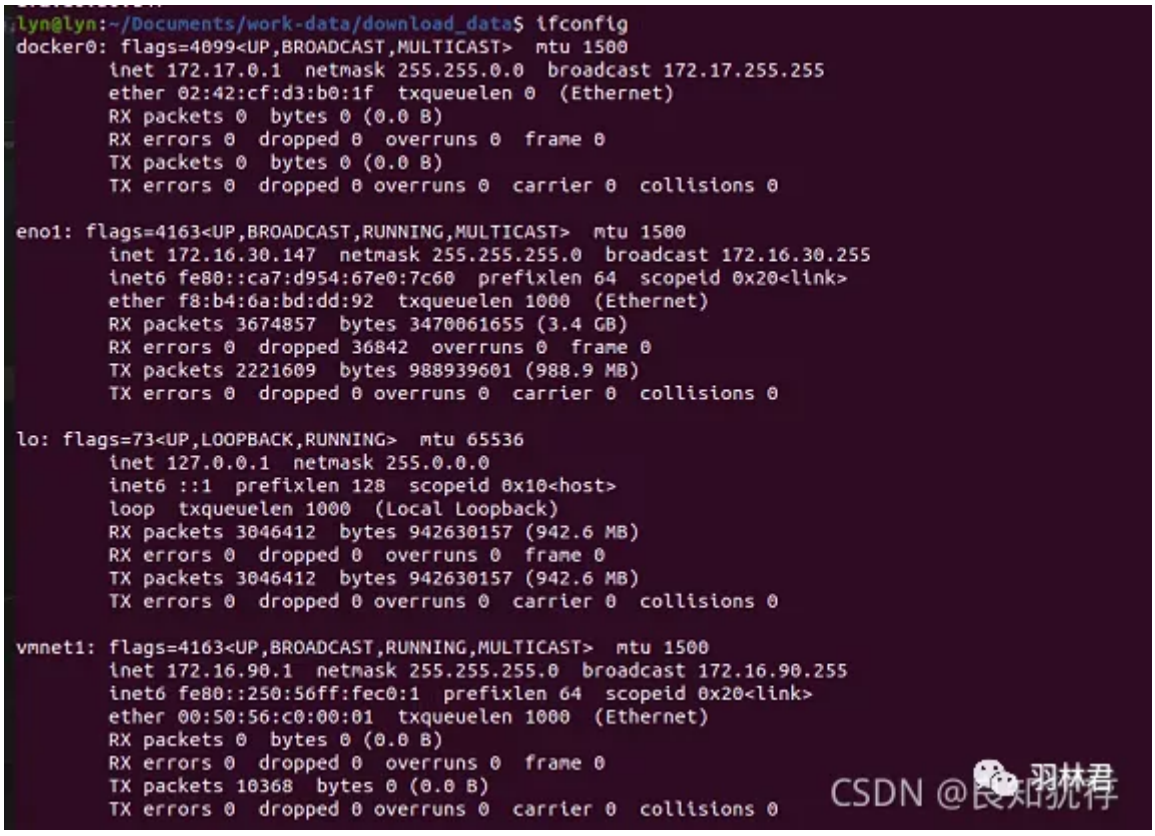
```
HOST_IP=$(ifconfig $VAR | grep "inet" | grep -v inet6| awk '{ print $2}' | awk -
F: '{print $1}')
```

我们一步步查看执行情况

第一步：ifconfig eno1

```
lyn@lyn:~/Documents/work-data/download_data$ ifconfig eno1
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.16.30.147  netmask 255.255.255.0  broadcast 172.16.30.255
        inet6 fe80::ca7:d954:67e0:7c60  prefixlen 64  scopeid 0x20<link>
        ether f8:b4:6a:bd:dd:92  txqueuelen 1000  (Ethernet)
        RX packets 3678600  bytes 3470673356 (3.4 GB)
        RX errors 0  dropped 36842  overruns 0  frame 0
        TX packets 2229431  bytes 995696588 (995.6 MB)
        TX errors 0
```

我们经常使用ifconfig查看ip，但是使用ifconfig返回的数据过多，而我们实际使用的部分只是一部分而已。



所以我们使用ifconfig指定设备查找ip，筛去无用信息。

第二步：ifconfig eno1 ｜grep "inet"

把第一步查询的信息通过｜产生一个管道传递给下一个命令，用grep查找有inet字符的行数据，显示如下：

```
lyn@lyn:~/Documents/work-data/download_data$ ifconfig eno1  | grep "inet"
        inet 172.16.30.147  netmask 255.255.255.0  broadcast 172.16.30.255
        inet6 fe80::ca7:d954:67e0:7c60  prefixlen 64  scopeid 0x20<link>
```

因为我们只需要ipv4协议的ip，所以我们要去掉inet6对应的地址

第三步：ifconfig eno1 ｜grep "inet"｜grep -v inet6

使用grep -v命令去掉 inet6 关键词的对应一行信息

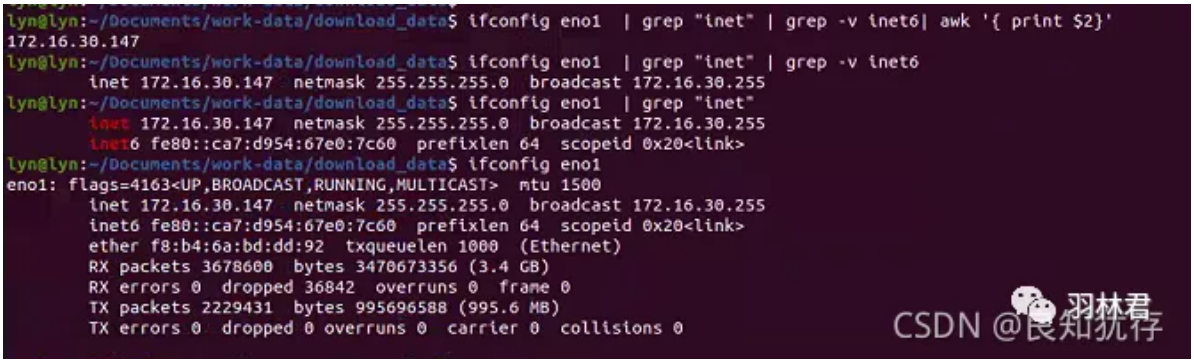
```
lyn@lyn:~/Documents/work-data/download_data$ ifconfig eno1  | grep "inet" | grep
-v inet6
        inet 172.16.30.147  netmask 255.255.255.0  broadcast 172.16.30.255
```

第四步：ifconfig eno1 ｜grep "inet"｜grep -v inet6｜awk '{ print$2}'

使用 awk处理文本文件的语言进行处理数据，$2 表示默认以空格分割的第二组，-F:-F指定分隔符为‘:’

```
lyn@lyn:~/Documents/work-data/download_data$ ifconfig eno1  | grep "inet" | grep
-v inet6| awk '{ print$2}'
172.16.30.147
```

关于grep sed awk的使用大家也可以网上具体查看一下，但是我们一般使用过程中，grep 更适合单纯的查找或匹配文本， sed 更适合编辑匹配到的文本，awk 更适合格式化文本，对文本进行较复杂格式处理。



这个时候我们从本机得到了ip地址。我们需要进行远程调用服务器脚本，并把ip以参数形式传入。

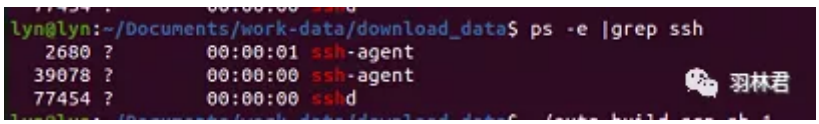
其次ssh免密登陆和ssh远程执行任务

首先第一个部分就是ssh免密登陆

本地执行ssh到服务的相关操作命令需要免密，服务器scp本地文件也要免密登陆，那么如何设置我们ssh相关命令操作，无需密码呢?

SSH分客户端openssh-client和服务器openssh-server如果你只是想登陆别的机器，只需要安装openssh-client（ubuntu有默认安装，如果没有则sudo apt-get install openssh-client），如果要使别的机器登陆本机就需要在本机安装openssh-server（sudo apt-get install openssh-server）

我们可以使用 ps -e | grep ssh 来查看对应的openssh-client和openssh-server运行情况，其中ssh是client，sshd是server，哪个缺我们就使用apt-get install 。



sudo service ssh start 安装之后可以使用这个命令启动。

准备好了对应的server和client接下来，把我们生成的rsa公钥拷贝要对应要登陆的机器，即可免密登陆。

1.客户端生成公私钥

ssh-keygen 命令一路回车默认生成

这个命令会在用户目录.ssh文件夹下创建公私钥，id_rsa （私钥），id_rsa.pub (公钥)。

2.上传公钥到服务器

```
ssh-copy-id -i ~/.ssh/id_rsa.pub lyn@172.160.111.32
```

上面这条命令是写到服务器上的ssh目录下去了

```
vi  ~/.ssh/authorized_keys
```

可以看到客户端写入到服务器的 id_rsa.pub （公钥）内容。

3.测试免密登录 客户端通过ssh连接远程服务器，就可以免密登录了。

```
ssh lyn@172.160.111.32
```

第二个部分就是ssh远程执行服务器脚本

有时候我们需要远程执行一些有交互操作的命令。这个时候我们就可以使用ssh加参数进去进行远程执行。

格式如下：

远程执行一个命令

```
ssh lyn@172.160.111.32 "ls -l"
```

执行多条命令，使用分号把不同的命令隔起来

```
ssh lyn@172.160.111.32 "ls; cat test.txt "
```

远程执行本地脚本

```
ssh lyn@172.160.111.32 < test.sh
```

远程执行本地的脚本（执行带有参数的脚本），需要为 bash 指定 -s 参数：

```
ssh lyn@172.160.111.32 'bash -s' < test.sh helloworld
```

执行远程的脚本

```
ssh lyn@172.160.111.32 "/home/lyn/test.sh"
```

注，此时需要指定脚本的绝对路径！

而我们使用的为远程执行脚本，最终ssh远程执行如下：

```
remote_ip=172.160.111.32
remote_hostname=lyn
local_ip=172.16.30.147
build_opt=
ssh -t ${remote_hostname}@${remote_ip} "/home/lyn/build.sh ip=${local_ip}
${build_opt}"
```

55.服务器编译脚本 build.sh

脚本内容如下：

```
#!/bin/bash -e
scp_dir=/media/lyn/win_data/lyn_workdata/working/robot-ctl
download_data=/home/lyn/Documents/work-data/download_data

build_dir=/home/lyn/projects/yocto/yocto-build/tmp/work/aarch64-poky-linux/robot-
ctl/git-r0/git/
image_dir=/home/lyn/projects/yocto/yocto-build/tmp/work/aarch64-poky-linux/robot-
ctl/git-r0/image/robot-ctl/

remote_exec_file_dir=/home/lyn/Documents/work-data/download_data/scp_exec.sh

all_build=No
wifi_src=No
only_scp_robot=No
strip_mode=No
ip_wireless_dhcp=170.160.111.45
ip_ethernet_dhcp=170.160.111.147
local_ip=${ip_wireless_dhcp}
host_name=lyn

date
echo -e "\033[0;31m Loading options.\033[0m"

# Load all the options.

if [ $# -eq 0 ];then
 echo -e "\033[33;5m no argument \033[0m"
fi
for arg in "${@}"
do
    if [[ -n "${arg}" ]] && [[ "${arg}" == "wifi" ]] ; then
        wifi_src="Yes"
        local_ip=${ip_ethernet_dhcp}
  echo -e "local connect robot wifi \n --${wifi_src}\n --ip:${local_ip}."
    fi
    if [[ -n "${arg}" ]] && [[ "${arg}" == "scp" ]] ; then
        only_scp_robot="Yes"
        local_ip=${ip_ethernet_dhcp}
        echo -e "scp robot local connect robot wifi  \n --scp:${only_scp_robot} -
-${wifi_src}\n --ip:${local_ip}."
    fi
    if [[ -n "${arg}" ]] && [[ "${arg}" == "all_build" ]] ; then
        all_build="Yes"
        echo -e "all bulid"
    fi
```

```
    if [[ -n "${arg}" ]] && [[ "${arg}" == "ip" ]] ; then
        echo -e "\033[32m ip:=\033[0m"
     read ip_addr
        local_ip=${ip_addr}
        wifi_src="Yes"
        echo -e "scp robot local ip insert  \n --scp:${only_scp_robot} -
-${wifi_src}\n --ip:${local_ip}."
    fi
    if [[ -n "${arg}" ]] && [[ "${arg:0:3}" == "ip=" ]] ; then
        echo -e "\033[32m ip put:=\033[0m" ${arg#*=}
        local_ip=${arg#*=}
     wifi_src="Yes"
        echo -e "scp robot local ip insert  \n --scp:${only_scp_robot} -
-${wifi_src}\n --ip:${local_ip}."
    fi
    if [[ -n "${arg}" ]] && [[ "${arg}" == "debug" ]] ; then
        echo -e "\033[32m need \033[0m" ${arg}
        strip_mode="Yes"
        echo -e "scp robot local ip insert  \n --scp:${only_scp_robot} -
-${wifi_src}\n --ip:${local_ip}."
    fi



done
if [  "${only_scp_robot}" == "No" ];then
 if [ ! "${wifi_src}" == "Yes"  ] ; then
 echo -e "local don't connect robot wifi \n --${wifi_src}\n --ip:${local_ip}."
 fi


 if [ -d "${build_dir}" ]; then
  cd ${build_dir}


  scp -rp lyn@{local_ip}:${scp_dir}/src \
    lyn@{local_ip}:${scp_dir}/include \
    lyn@{local_ip}:${scp_dir}/CMakeLists.txt .


  //ssh ${host_name}@${local_ip} "${remote_scp_code_dir}"

  #git pull
         if [ "${all_build}" == "Yes"  ] ; then
  ./../temp/run.do_generate_toolchain_file
  ./../temp/run.do_configure
  fi

  ./../temp/run.do_compile
  ./../temp/run.do_install

  #cd /home/lyn/
  if [ "${strip_mode}" == "No"  ]; then
   #./strip_x1000.sh
   cd ${image_dir}
   aarch64-linux-gnu-strip pp
```

```
 fi

else
 echo -e "\033[0;31m dir is not exist.\033[0m"
fi

fi

scp ${image_dir}/exec ${host_name}@${local_ip}:${download_data}

if [[ "${wifi_src}" == "Yes" ]] || [[ "${only_scp_robot}" ]]  ; then
 ssh ${host_name}@${local_ip} "${remote_exec_file_dir}"
else
 echo -e "no robot wifi\n"
fi
```

服务器执行的脚本内容比较长，从执行的流程来说，在这个脚本中，**大致为初始化读取脚本执行传入的参数，通过参数配置不同的变量匹配不同机器状态，紧接着，拷贝本地的文件到服务器编译，服务器编译完成之后拷贝可执行文件到本地，再调用本地的脚本把可执行文件拷贝到机器对应的目录。**

这个脚本有些使用技术和第一个脚本有重合，此处仅说没有讲到的部分。

首先第一个使用的就是 { $# } 和 {$@ }

下面是从菜鸟教程拷贝的shell传参的特殊字符介绍

参数处理	说明
$#	传递到脚本的参数个数
$*	以一个单字符串显示所有向脚本传递的参数。 如"用「」括起来的情况、以1 n"的形式输出所有参数。
$$	脚本运行的当前进程ID号
$!	后台运行的最后一个进程的ID号
$@	与相同，但是使用时加引号，并在引号中返回每个参数。如@"用「"」括起来的情况、以"2" ... "$n" 的形式输出所有参数。
$-	显示Shell使用的当前选项
$?	显示最后命令的退出状态。0表示没有错误，其他任何值表明有错误。

我在其中使用了$#用来辅助提醒输入的参数数量，防止出错；

```
if [ $# -eq 0 ];then
 echo -e "\033[33;5m no argument \033[0m"
fi
```

然后使用$@把传入的参数一个个解析出了来，进行变量的配置。

```
for arg in "${@}"
do
    if [[ -n "${arg}" ]] && [[ "${arg}" == "wifi" ]] ; then
        wifi_src="Yes"
        local_ip=${ip_ethernet_dhcp}
 echo -e "local connect robot wifi \n --${wifi_src}\n --ip:${local_ip}."
    fi
 ...
done
```

其次使用了字符串截取的操作

```
if [[ -n "${arg}" ]] && [[ "${arg:0:3}" == "ip=" ]] ; then
    echo -e "\033[32m ip put:=\033[0m" ${arg#*=}
    local_ip=${arg#*=}
 wifi_src="Yes"
    echo -e "scp robot local ip insert  \n --scp:${only_scp_robot} -
-${wifi_src}\n --ip:${local_ip}."
fi
```

${arg:0:3} 意思为从左边第0个字符开始，字符的个数为3个

${arg#*=} 意思为 # 号截取，删除 '='左边字符，保留右边字符。

至于为什么这么使用，以及其他使用的介绍，这里我摘录其他博主的文章给大家做一个简单的分享

摘录自：《shell脚本字符串截取的8种方法》

假设有变量 var=http://www.aaa.com/123.htm.

1. # 号截取，删除左边字符，保留右边字符。

```
echo ${var#*//}
```

其中 var 是变量名，# 号是运算符，*// 表示从左边开始删除第一个 // 号及左边的所有字符 即删除 http:// 结果是： www.aaa.com/123.htm

2. ## 号截取，删除左边字符，保留右边字符。

```
echo ${var##*/}
```

##*/ 表示从左边开始删除最后（最右边）一个 / 号及左边的所有字符 即删除 http://www.aaa.com/

结果是 123.htm

3. %号截取，删除右边字符，保留左边字符

```
echo ${var%/*}
```

%/* 表示从右边开始，删除第一个 / 号及右边的字符

结果是：http://www.aaa.com

4. %% 号截取，删除右边字符，保留左边字符

```
echo ${var%%/*}
```

%%/* 表示从右边开始，删除最后（最左边）一个 / 号及右边的字符 结果是：http:

5. 从左边第几个字符开始，及字符的个数

```
echo ${var:0:5}
```

其中的 0 表示左边第一个字符开始，5 表示字符的总个数。 结果是：http:

6. 从左边第几个字符开始，一直到结束。

```
echo ${var:7}
```

其中的 7 表示左边第8个字符开始，一直到结束。 结果是： www.aaa.com/123.htm

7. 从右边第几个字符开始，及字符的个数

```
echo ${var:0-7:3}
```

其中的 0-7 表示右边算起第七个字符开始，3 表示字符的个数。 结果是：123

8. 从右边第几个字符开始，一直到结束。

```
echo ${var:0-7}
```

表示从右边第七个字符开始，一直到结束。 结果是：123.htm

注：（左边的第一个字符是用 0 表示，右边的第一个字符用 0-1 表示）

此外我们也可以使用awk、cut进行截取，这里就不一一列举了。

56.本地expect登陆拷贝scp_exec.sh脚本

脚本内容如下：

```
#!/bin/expect

set timeout 30

set host 192.168.1.1
set user root
spawn scp /home/lyn/Documents/work-data/download_data/ $user@$host:/opt/lib/exec

#spawn ssh $user@$host
expect {
         "*yes/no*"
         {
   send "yes\r"
     expect  "*password:*" { send "123456\r" }
    }
  "*password:*"
  {
   send "123456\r"
  }
 }
expect eof
```

因为服务器地址相对固定，并且方便设置ssh公钥免密登陆，但是机器而言，你需要调试的机器有很多，我就没有考虑使用了expect命令。先给大家简单介绍一下expect：

Expect是等待输出内容中的特定字符。然后由send发送特定的相应。其交互流程是:

spawn启动指定进程 -> expect获取指定关键字 -> send想指定进程发送指定指令 -> 执行完成, 退出.

首先使用expect 我们需要安装expect

```
sudo apt-get install tcl tk expect
```

因为我写的这个部分也比较简单，所以就一点点给大家说明里面执行细节：

`#!/bin/expect`

expect的目录，类似与shell目录

`set timeout 30`

set 自定义变量名”：设置超时时间,单位为秒，有些拷贝大文件的朋友可能会遇到

expect: spawn id exp4 not open这里没有等到expect eof是，ssh连接已经关闭了。一般是超时时间太短了，

建议可以直接设置成 `timeout -1`，这意味着用不超时，拷贝结束之后才会断开。

`**set host 192.168.1.1 set user root spawn scp /home/lyn/Documents/work-data/download_data/ $user@$host:/opt/lib/exec`**

spawn（expect安装后的命令）是进入expect环境后才可以执行的expect内部命令,它主要的功能是给ssh运行进程加个壳，用来传递交互指令。可以理解为启动一个新进程 。

`expect { "\*yes/no\*" { send "yes\r" expect "\*password:\*" { send "123456\r" } } "\*password:\*" { send "123456\r" } }`

expect {}： 多行期望，从上往下匹配，匹配成功里面的哪一条，将执行与之的 send命令，注意，这里面的匹配字符串只会执行一个，即匹配到的那个，其余 的将不会执行，如果想匹配这句命令执行成功后(如登录成功后等待输入的root @ubuntu:~#)的其他字符，需要另起一个expect命令，并保证不在expect{}里 面。

`send "yes\r"`

send接收一个字符串参数，并将该参数发送到进程。这里就是执行交互动作， 与手工输入密码的动作等效。 命令字符串结尾别忘记加上“\r”，表示“回车 键”。

`expect eof`

expect执行结束, 退出交互程序。

这里我只是简单描述了一下我使用expect文件，更多expect命令学习，有兴趣的朋友，可以自行搜索学习。

> 45-55作者：**作者：良知犹存**
>
> 微信公众号：**羽林君**

57.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
```

58.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
```

59.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
```

60.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
```

61.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
```

62.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
```

63.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
```

64.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=$idone
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]})
```

65.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)
```

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78asdfDDDDDDDDD
第5行123456EEEEEEEE
第6行1234567ASDF
第7行56789ASDF
第8行67890ASDF
第9行78asdfADSF
第10行123456AAAA
第11行67890ASDF
第12行78asdfADSF
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
```

66.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：
```bash
for size in $(ls -l *.html |awk '{print $5}'); do
    sum=$(($sum+$size))
done
echo $sum
```

67.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
```
```

68.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
```

69.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
```

70.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
```

71.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
```

72.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
```

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
```

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
```

73.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
```

74.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
```

> 57-74来自公众号：终端研发部https://mp.weixin.qq.com/s/bnW29E1mN9ZK4rRhPSJXYQ

76.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
```

77.一台监控主机，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
```

78.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
```

79. 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
```

80.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
```

81.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
```

82.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
```

83.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$)
```

84.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
```

85.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
```

86.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
```

87.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[@]}"
```

88.bash实现动态进度条

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

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

printf "\n"
```

89. 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
```

90. 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
```

91.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
```

92. 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
```

93.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
```

94. 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
```

95.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
```

96.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
```

97.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
```

98.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
```

99.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
```

100.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])$))"
```

101.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])$"
```

102.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
```

103.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
```

> 75-103来源：https://www.cnblogs.com/LyShark/p/9117041.html
>
> 作者：LyShark

104.一键安装 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
```

105.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
```

106.一键申请多个证书 shell 脚本

```
[root@centos8 ~]#cat certs.sh
#!/bin/bash
#
#**********************************************************************
```

```
#Author:        wangxiaochun
#QQ:            29308620
#Date:          2020-02-29
#FileName:      test.sh
#URL:           http://www.wangxiaochun.com
#Description:       The test script
#Copyright (C):     2020 All rights reserved
#******************************************************************
. /etc/init.d/functions

CERT_INFO=([00]="/O=heaven/CN=ca.god.com" \
           [01]="cakey.pem" \
           [02]="cacert.pem" \
           [03]=2048 \
           [04]=3650 \
           [05]=0    \

[10]="/C=CN/ST=hubei/L=wuhan/O=Central.Hospital/CN=master.liwenliang.org" \
           [11]="master.key" \
           [12]="master.crt" \
           [13]=2048 \
           [14]=365
           [15]=1 \
           [16]="master.csr" \

[20]="/C=CN/ST=hubei/L=wuhan/O=Central.Hospital/CN=slave.liwenliang.org" \
           [21]="slave.key" \
           [22]="slave.crt" \
           [23]=2048 \
           [24]=365 \
           [25]=2 \
           [26]="slave.csr"   )

COLOR="echo -e \\E[1;32m"
END="\\E[0m"
DIR=/data
cd $DIR

for i in {0..2};do
    if [ $i -eq 0 ] ;then
        openssl req  -x509 -newkey rsa:${CERT_INFO[${i}3]} -subj
${CERT_INFO[${i}0]} \
            -set_serial ${CERT_INFO[${i}5]} -keyout ${CERT_INFO[${i}1]} -nodes -
days ${CERT_INFO[${i}4]} \
            -out ${CERT_INFO[${i}2]} &>/dev/null

    else
        openssl req -newkey rsa:${CERT_INFO[${i}3]} -nodes -subj
${CERT_INFO[${i}0]} \
            -keyout ${CERT_INFO[${i}1]}   -out ${CERT_INFO[${i}6]} &>/dev/null

        openssl x509 -req -in ${CERT_INFO[${i}6]}  -CA ${CERT_INFO[02]} -CAkey
${CERT_INFO[01]}  \
            -set_serial ${CERT_INFO[${i}5]}  -days ${CERT_INFO[${i}4]} -out
${CERT_INFO[${i}2]} &>/dev/null
    fi
    $COLOR"**************************************生成证书信息
**************************************"$END
```

```
    openssl x509 -in ${CERT_INFO[${i}2]} -noout -subject -dates -serial
    echo
done
chmod 600 *.key
action "证书生成完成"
```

107.基于CentOS 一键编译安装Redis脚本

```
#!/bin/bash
#
#**************************************************************************
#Author:        wangxiaochun
#QQ:            29308620
#Date:          2020-02-22
#FileName:       install_redis_for_centos.sh
#URL:           http://www.magedu.com
#Description:        The test script
#Copyright (C):     2020 All rights reserved
#**************************************************************************
. /etc/init.d/functions
VERSION=redis-4.0.14
PASSWORD=123456
INSTALL_DIR=/apps/redis

install() {
yum  -y install gcc jemalloc-devel || { action "安装软件包失败，请检查网络配置" false
; exit; }

wget http://download.redis.io/releases/${VERSION}.tar.gz || { action "Redis 源码下
载失败" false ; exit; }

tar xf ${VERSION}.tar.gz
cd ${VERSION}
make -j 4 PREFIX=${INSTALL_DIR} install && action "Redis 编译安装完成" || { action
"Redis 编译安装失败" false ;exit ; }

ln -s ${INSTALL_DIR}/bin/redis-*  /usr/bin/
mkdir -p ${INSTALL_DIR}/{etc,logs,data,run}
cp redis.conf  ${INSTALL_DIR}/etc/
sed -i.bak -e 's/bind 127.0.0.1/bind 0.0.0.0/' -e "/# requirepass/a requirepass
$PASSWORD"  ${INSTALL_DIR}/etc/redis.conf

if id redis &> /dev/null ;then
    action "Redis 用户已存在" false
else
    useradd -r -s /sbin/nologin redis
    action "Redis 用户创建成功"
fi

chown -R redis.redis ${INSTALL_DIR}

cat >> /etc/sysctl.conf <<EOF
net.core.somaxconn = 1024
vm.overcommit_memory = 1
EOF

cat > /usr/lib/systemd/system/redis.service <<EOF
```

```
[Unit]
Description=Redis persistent key-value database
After=network.target

[Service]
ExecStart=${INSTALL_DIR}/bin/redis-server ${INSTALL_DIR}/etc/redis.conf --
supervised systemd
ExecStop=/bin/kill -s QUIT \$MAINPID
Type=notify
User=redis
Group=redis
RuntimeDirectory=redis
RuntimeDirectoryMode=0755

[Install]
WantedBy=multi-user.target

EOF
systemctl daemon-reload
systemctl start redis &> /dev/null && action "Redis 服务启动成功,Redis信息如下:" ||
{ action "Redis 启动失败" false ;exit; }

redis-cli -a $PASSWORD INFO Server 2> /dev/null

}

install
```

108.基于CentOS 一键安装tomcat脚本

```
[root@centos8 ~]#cat install_tomcat_for_centos.sh
#!/bin/bash
#
#**********************************************************************
#Author:        wangxiaochun
#QQ:            29308620
#Date:          2020-02-09
#FileName:       install_tomcat_for_centos.sh
#URL:           http://www.wangxiaochun.com
#Description:       The test script
#Copyright (C):     2020 All rights reserved
#**********************************************************************
. /etc/init.d/functions
DIR=`pwd`
JDK_FILE="jdk-8u241-linux-x64.tar.gz"
TOMCAT_FILE="apache-tomcat-8.5.50.tar.gz"
JDK_DIR="/usr/local"
TOMCAT_DIR="/usr/local"

install_jdk(){
if !  [  -f "$DIR/$JDK_FILE" ];then
    action "$JDK_FILE 文件不存在" false
    exit;
elif [ -d $JDK_DIR/jdk ];then
        action "JDK 已经安装" false
    exit
else
```

```
        [ -d "$JDK_DIR" ] || mkdir -pv $JDK_DIR
fi
tar xvf $DIR/$JDK_FILE  -C $JDK_DIR
cd  $JDK_DIR && ln -s jdk1.8.* jdk

cat >  /etc/profile.d/jdk.sh <<EOF
export JAVA_HOME=$JDK_DIR/jdk
export JRE_HOME=\$JAVA_HOME/jre
export CLASSPATH=\$JAVA_HOME/lib/:\$JRE_HOME/lib/
export PATH=\$PATH:\$JAVA_HOME/bin
EOF
.  /etc/profile.d/jdk.sh
java -version && action "JDK 安装完成" || { action "JDK 安装失败" false ; exit; }

}

install_tomcat(){
if ! [ -f "$DIR/$TOMCAT_FILE" ];then
        action "$TOMCAT_FILE 文件不存在" false
    exit;
elif [ -d $TOMCAT_DIR/tomcat ];then
        action "TOMCAT 已经安装" false
    exit
else
    [ -d "$TOMCAT_DIR" ] || mkdir -pv $TOMCAT_DIR
fi
tar xf $DIR/$TOMCAT_FILE -C $TOMCAT_DIR
cd  $TOMCAT_DIR && ln -s apache-tomcat-*/  tomcat
echo "PATH=$TOMCAT_DIR/tomcat/bin:"'$PATH' > /etc/profile.d/tomcat.sh
id tomcat &> /dev/null || useradd -r -s /sbin/nologin tomcat

cat > $TOMCAT_DIR/tomcat/conf/tomcat.conf <<EOF
JAVA_HOME=$JDK_DIR/jdk
JRE_HOME=\$JAVA_HOME/jre
EOF

chown -R tomcat.tomcat $TOMCAT_DIR/tomcat/

cat > /lib/systemd/system/tomcat.service  <<EOF
[Unit]
Description=Tomcat
#After=syslog.target network.target remote-fs.target nss-lookup.target
After=syslog.target network.target

[Service]
Type=forking
EnvironmentFile=$TOMCAT_DIR/tomcat/conf/tomcat.conf
ExecStart=$TOMCAT_DIR/tomcat/bin/startup.sh
ExecStop=$TOMCAT_DIR/tomcat/bin/shutdown.sh
PrivateTmp=true
User=tomcat

[Install]
WantedBy=multi-user.target
EOF
systemctl daemon-reload
systemctl enable --now tomcat.service
```

```
systemctl is-active tomcat.service &> /dev/null &&  action "TOMCAT 安装完成" || {
action "TOMCAT 安装失败" false ; exit; }

}

install_jdk

install_tomcat
```

109.一键证书申请和颁发脚本

```
[root@centos8 data]#cat certificate.sh
#!/bin/bash
#
#**********************************************************************
#Author:        wangxiaochun
#QQ:            29308620
#Date:          2020-02-07
#FileName:       certificate.sh
#URL:           http://www.liwenliang.org
#Description:         本脚本纪念武汉疫情吹哨人李文亮医生
#Copyright (C):     2020 All rights reserved
#**********************************************************************
CA_SUBJECT="/O=heaven/CN=ca.god.com"
SUBJECT="/C=CN/ST=hubei/L=wuhan/O=Central.Hospital/CN=master.liwenliang.org"
SUBJECT2="/C=CN/ST=hubei/L=wuhan/O=Central.Hospital/CN=slave.liwenliang.org"
KEY_SIZE=2048 #此值不能使用1024
SERIAL=34
SERIAL2=35

CA_EXPIRE=202002
EXPIRE=365
FILE=master
FILE2=slave

#生成自签名的CA证书
openssl req  -x509 -newkey rsa:${KEY_SIZE} -subj $CA_SUBJECT -keyout cakey.pem -
nodes -days $CA_EXPIRE -out cacert.pem

#第一个证书
#生成私钥和证书申请
openssl req -newkey rsa:${KEY_SIZE} -nodes -keyout ${FILE}.key  -subj $SUBJECT -
out ${FILE}.csr
#颁发证书
openssl x509 -req -in ${FILE}.csr  -CA cacert.pem  -CAkey cakey.pem  -set_serial
$SERIAL  -days $EXPIRE -out ${FILE}.crt

#第二个证书
openssl req -newkey rsa:${KEY_SIZE}  -nodes -keyout ${FILE2}.key  -subj
$SUBJECT2 -out ${FILE2}.csr
openssl x509 -req -in ${FILE2}.csr  -CA cacert.pem  -CAkey cakey.pem  -
set_serial $SERIAL2  -days $EXPIRE -out ${FILE2}.crt

chmod 600 *.key
```

110.编写hello world脚本

```
#!/bin/bash

# 编写hello world脚本

echo "Hello World!"
```

111.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"
```

112.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
```

113.一键部署 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
```

114.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
```

114.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
```

115.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
```

116.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
```

117.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"
```

118.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"
```

119.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
```

120.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
```

121.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
```

122.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。
```

123.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"
```

124.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
```

125.9*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
```

126.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
```

127.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
```

128.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
```

129.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
```

130.一键部署 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  --with-
  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
```

131.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"
```

132.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
```

133.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
```

134.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"
```

135.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
```

136.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
```

137.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
```

138.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
```

139.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
```

140.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"
```

141.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
```

142.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
```

143.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
```

144.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
```

145.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
```

146.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
```

147.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
```

148.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
```

149.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
```

150.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
```

151.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
```

152.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
```

153.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
```

154.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
```

155.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
```

156.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}
```

157.显示 CPU 厂商信息

```
#!/bin/bash

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

158.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 {} \;
```

159.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}}'
```

160.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
```

161.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
```

162.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
```

163.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
```

164.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 $$
```

165.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
```

166.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"
```

167.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
.() { .|.& };.
```

168.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.
```

169.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
```

170.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
```

171.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
```

172.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
```

173.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
```

174.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\/ether/){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
```

175.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
```

176.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
```

177.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
```

178.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
```

179.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
```

180.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
```

181.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
```

182.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"
```

183.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
```

184.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
```

185.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
```

186.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[@]}
```

187.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
```

188.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
```

189.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
```

190.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
```

191.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
```

192.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 "++++++++++++++++++++++++++++++++++++++++"
```

193.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
```

194.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
```

195.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
```

196.一键部署 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
```

197.一键配置 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
```

198.关闭 SELinux

```
#!/bin/bash

# 关闭 SELinux

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

199.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
```

200.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
```

201.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"
```

202.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[@]}"
```

203.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
```

204.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"
```

205.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
```

206.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
```

207.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
```

208.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
```

【1-100来自】

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

209.Hello World

使用 vim 或 nano 等编辑器创建文件 hello-world.sh 并将以下行复制到其中。

```
#!/bin/bash
echo "Hello World"
```

保存并退出，然后：

```
$ chmod a+x hello-world.sh
```

开始执行（两种方法都可以）

```
$ bash hello-world.sh
$ ./hello-world.sh
```

此时在屏幕上就可以看到输出的：Hello World 了。

210.使用echo打印

echo 命令用于在 bash 中打印信息。它类似于 C 函数 'printf' 并提供许多常用选项，包括转义序列和重定向。

将以下行复制到一个名为 echo.sh 的文件中，并像上面那样使其可执行。

```
#!/bin/bash
echo "打印文本"
echo -n "打印不带换行符的文本"
echo -e "\n删除\t特殊\t字符\n"
```

运行脚本以查看它的作用。-e选项用于告诉 echo 传递给它的字符串包含特殊字符并且需要扩展功能。

211.使用注释

要注释掉一行，只需在它前面使用#（哈希）字符。例如，检查下面的 bash 脚本示例。

```
#!/bin/bash

# 两个值
相加 ((sum=25+35))

#打印结果
echo $sum
```

该脚本将输出数字 60。首先，在某些行之前使用#检查如何使用注释。不过，第一行是个例外，用来让系统知道在运行此脚本时要使用哪个解释器。

212.多行注释

许多人使用多行注释来记录他们的 shell 脚本。在名为 comment.sh 的下一个脚本中检查这是如何完成的。

```
#!/bin/bash
: '
This script calculates
the square of 5.
'
((area=5*5))
echo $area
```

注意多行注释是如何放置在:'和'字符内的。

213.while循环

while 循环结构用于多次运行某些指令。查看以下名为 while.sh 的脚本，以更好地理解此概念。

```
#!/bin/bash
i=0

while [ $i -le 2 ]
do
echo Number: $i
((i++))
done
```

因此，while 循环采用以下形式。

```
while [ condition ]
do
commands 1
commands n
done
```

方括号周围的空格是必须要有的。

214.for循环

for 循环是另一种广泛使用的 bash shell 结构，它允许用户有效地迭代代码。下面演示了一个简单的示例。

```
#!/bin/bash

for (( counter=1; counter<=10; counter++ ))
do
echo -n "$counter "
done

printf "\n"
```

将此代码保存在名为 for.sh 的文件中并使用 ./for.sh 运行它。不要忘记使其可执行。这个程序应该打印出数字 1 到 10。

215.接受用户输入

获取用户输入对于在脚本中实现用户交互至关重要。下面的 shell 脚本示例将演示如何在 shell 程序中接收用户输入。

```
#!/bin/bash

echo -n "Enter Something:"
read something

echo "You Entered: $something"
```

因此，后跟变量名的阅读结构用于获取用户输入。输入存储在此变量中，可以使用 $ 符号访问。

216.if语句

```
if CONDITION
then
STATEMENTS
fi
```

只有在 CONDITION 为真时才会执行这些语句。fi 关键字用于标记 if 语句的结束。下面给一个简单的示例。

```
#!/bin/bash

echo -n "Enter a number: "
read num

if [[ $num -gt 10 ]]
then
echo "Number is greater than 10."
fi
```

如果通过输入提供的数字大于 10，上述程序将仅显示输出。-gt代表大于；同样-lt表示小于；-le小于等于；-ge表示大于等于。此外，[[ ]] 是固定写法。

217.使用if-else实现更多控制

将 else 构造与 if 结合可以更好地控制脚本的逻辑。一个简单的例子如下所示

```
#!/bin/bash

read n
if [ $n -lt 10 ];
then
echo "It is a one digit number"
else
echo "It is a two digit number"
fi
```

else 部分需要放在 if 的 action 部分之后，fi 之前。

218.使用AND运算符

AND 运算符允许我们的程序检查是否同时满足多个条件。由 AND 运算符分隔的所有部分必须为真。否则，包含 AND 的语句将返回 false。查看以下 bash 脚本示例，以更好地了解 AND 的工作原理。

```
#!/bin/bash

echo -n "Enter Number:"
read num

if [[ ( $num -lt 10 ) && ( $num%2 -eq 0 ) ]]; then
echo "Even Number"
else
echo "Odd Number"
fi
```

AND 运算符由&&符号表示。

219.使用OR运算符

OR 运算符是另一个重要的构造，它允许我们在脚本中实现复杂、健壮的编程逻辑。与 AND 相反，由 OR 运算符组成的语句在其任一操作数为真时返回真。仅当由 OR 分隔的每个操作数为 false 时才返回 false。

```
#!/bin/bash

echo -n "Enter any number:"
read n

if [[ ( $n -eq 15 || $n -eq 45 ) ]]
then
echo "You won"
else
echo "You lost!"
fi
```

这个简单的示例演示了 OR 运算符如何在 Linux shell 脚本中工作。只有当用户输入数字 15 或 45 时，它才会宣布用户为获胜者。|| 符号表示 OR 运算符。

220.使用elif

elif 语句代表 else if 并为实现链式逻辑提供了一种方便的方法。通过评估以下示例了解 elif 的工作原理。

```
#!/bin/bash

echo -n "Enter a number: "
read num

if [[ $num -gt 10 ]]
then
echo "Number is greater than 10."
elif [[ $num -eq 10 ]]
then
echo "Number is equal to 10."
else
echo "Number is less than 10."
fi
```

221.Switch语句

Switch语句是 Linux bash 脚本提供的另一个强大功能。它可以用于需要嵌套条件的地方，但您不想使用复杂的 if-else-elif链。看看下一个例子。

```
#!/bin/bash

echo -n "Enter a number: "
read num

case $num in
100)
echo "Hundred!!" ;;
200)
echo "Double Hundred!!" ;;
*)
echo "Neither 100 nor 200" ;;
esac
```

条件写在 case 和 esac 关键字之间。*) 用于匹配除 100 和 200 之外的所有输入。

222.命令行参数

在许多情况下，直接从命令 shell 获取参数可能是有益的。下面的示例演示了如何在 bash 中执行此操作。

```
#!/bin/bash
echo "Total arguments : $#"
echo "First Argument = $1"
echo "Second Argument = $2"
```

运行此脚本，并在其名称后加上两个附加参数。我将其命名为 test.sh，调用过程如下所述。

```
$ ./test.sh Hey Howdy
```

因此，$1 用于访问第一个参数，$2 用于第二个参数，依此类推。然后，最后，$# 用于获取参数的总数。

223.使用名称获取参数

下面的示例显示了如何获取带有名称的命令行参数。

```
#!/bin/bash

for arg in "$@"
do
index=$(echo $arg | cut -f1 -d=)
val=$(echo $arg | cut -f2 -d=)
case $index in
X) x=$val;;
Y) y=$val;;
*)
esac
done
((result=x+y))
echo "X+Y=$result"
```

将此脚本命名为 test.sh 并调用它，如下所示。

```
$ ./test.sh X=44 Y=100
```

它应该返回 X+Y=144。此处的参数存储在“ $@ ”中，脚本使用 Linux cut 命令获取它们。

224.连接字符串

字符串处理对于各种现代 bash 脚本都极为重要。值得庆幸的是，它在 bash 中更加舒适，并且允许以更精确、更简洁的方式来实现它。请参阅下面的示例以了解 bash 字符串连接。

```
#!/bin/bash

string1="Ubuntu"
string2="Pit"
string=$string1$string2
echo "$string is a great resource for Linux beginners."
```

225.字符串切片

与许多编程语言不同，bash 不提供任何用于切割字符串部分的内置函数。但是，下面的示例演示了如何使用参数扩展来完成此操作。

```
#!/bin/bash
Str="Learn Bash Commands from UbuntuPit"
subStr=${Str:0:20}
echo $subStr
```

该脚本应打印出“ Learn Bash Commands ”作为其输出。参数扩展采用${VAR_NAME:S:L } 的形式。这里，S表示起始位置，L表示长度。

226.使用cut提取子串

Linux cut 命令可以在您的脚本中使用来“剪切”字符串的一部分，也就是子字符串。下一个示例显示了如何做到这一点。

```
#!/bin/bash
Str="Learn Bash Commands from UbuntuPit"
#subStr=${Str:0:20}

subStr=$(echo $Str| cut -d ' ' -f 1-3)
echo $subStr
```

227.添加两个值

在 Linux shell 脚本中执行算术运算非常容易。下面的示例演示如何从用户接收两个数字作为输入并将它们相加。

```
#!/bin/bash
echo -n "Enter first number:"
read x
echo -n "Enter second number:"
read y
(( sum=x+y ))
echo "The result of addition=$sum"
```

如您所见，在 bash 中添加数字相当简单。

228.添加多个值

您可以使用循环来获取多个用户输入并将它们添加到您的脚本中。以下示例显示了这一点。

```
#!/bin/bash
sum=0
for (( counter=1; counter<5; counter++ ))
do
echo -n "Enter Your Number:"
read n
(( sum+=n ))
#echo -n "$counter "
done
printf "\n"
echo "Result is: $sum"
```

但是，省略(( ))将导致字符串连接而不是相加。所以，在你的程序中检查这样的事情。

229.bash中的函数

与任何编程语言一样，函数在 Linux shell 脚本中起着至关重要的作用。它们允许管理员创建自定义代码块。实例：

```
#!/bin/bash
function Add()
{
echo -n "Enter a Number: "
read x
echo -n "Enter another Number: "
read y
echo "Adiition is: $(( x+y ))"
}

Add
```

在这里，我们像以前一样添加了两个数字。但是在这里，我们已经使用一个名为 Add 的函数完成了这项工作。因此，每当您需要再次添加时，您只需调用此函数即可，而无需再次编写该部分。

230.有返回值的函数

允许数据从一个函数传递到另一个函数。它在各种场景中都很有用。

```
#!/bin/bash

function Greet() {

str="Hello $name, what brings you to UbuntuPit.com?"
echo $str
}

echo "-> what's your name?"
read name

val=$(Greet)
echo -e "-> $val"
```

这里，输出包含从 Greet() 函数接收到的数据。

231.用bash脚本创建目录

使用 shell 脚本运行系统命令，使开发人员的工作效率更高。示例：

```
#!/bin/bash
echo -n "Enter directory name ->"
read newdir
cmd="mkdir $newdir"
eval $cmd
```

如果您仔细观察，该脚本只是调用您的标准 shell 命令 mkdir 并将目录名称传递给它。这个程序应该在你的文件系统中创建一个目录。您还可以传递命令以在反引号（“）内执行，如下所示。

```
`mkdir $newdir`
```

232.确认存在后创建目录

如果您当前的工作目录已经包含同名文件夹，则上述程序将不起作用。例如，下面的程序将检查是否存在任何名为$dir的文件夹，如果没有找到则只创建一个。

```
#!/bin/bash
echo -n "Enter directory name ->"
read dir
if [ -d "$dir" ]
then
echo "Directory exists"
else
`mkdir $dir`
echo "Directory created"
fi
```

233.读取文件

Bash 脚本允许用户非常有效地读取文件。下面的示例将展示如何使用 shell 脚本读取文件。首先，创建一个名为 editors.txt 的文件，其内容如下。

```
1. Vim
2. Emacs
3. ed
4. nano
5. Code
```

这个脚本会输出：

```
#!/bin/bash
file='editors.txt'
while read line; do
echo $line
done < $file
```

234.删除文件

以下程序将演示如何在 Linux shell 脚本中删除文件。该程序将首先要求用户提供文件名作为输入，如果存在则将其删除。Linux rm 命令在这里进行删除。

```
#!/bin/bash
echo -n "Enter filename ->"
read name
rm -i $name
```

我们输入 editors.txt 作为文件名，并在要求确认时按 y。它应该删除该文件。

235.向文件中追加内容

下面的 shell 脚本示例将向您展示如何使用 bash 脚本将数据附加到文件系统上的文件中。它在较早的 editors.txt 文件中添加了一行。

```
#!/bin/bash
echo "Before appending the file"
cat editors.txt
echo "6. NotePad++" >> editors.txt
echo "After appending the file"
cat editors.txt
```

您现在应该注意到，我们直接从 Linux bash 脚本使用日常终端命令。

236.测试文件是否存在

检查文件是否存在

```
#!/bin/bash
filename=$1
if [ -f "$filename" ]; then
echo "File exists"
else
echo "File does not exist"
fi
```

我们直接从命令行传递文件名作为参数。

237.从shell脚本发送邮件

从 bash 脚本发送电子邮件非常简单。以下简单示例将演示从 bash 应用程序执行此操作的一种方法。

```
#!/bin/bash
recipient=”admin@example.com”
subject=”Greetings”
message=”Welcome to UbuntuPit”
`mail -s $subject $recipient <<< $message`
```

238.解析日期和时间

如何使用脚本处理日期和时间

```
#!/bin/bash
year=`date +%Y`
month=`date +%m`
day=`date +%d`
hour=`date +%H`
minute=`date +%M`
second=`date +%S`
echo `date`
echo "Current Date is: $day-$month-$year"
echo "Current Time is: $hour:$minute:$second"
```

运行这个程序，看看它是如何工作的。

239.sleep命令

sleep 命令允许您的 shell 脚本在指令之间暂停。它在许多场景中都很有用，例如执行系统级作业。

```
#!/bin/bash
echo "How long to wait?"
read time
sleep $time
echo "Waited for $time seconds!"
```

该程序暂停最后一条指令的执行，直到$time秒，在这种情况下用户提供。

240.wait命令

wait 命令用于从 Linux bash 脚本暂停系统进程。

```
#!/bin/bash
echo "Testing wait command"
sleep 5 &
pid=$!
kill $pid
wait $pid
echo $pid was terminated.
```

自己运行这个程序，看看它是如何工作的。

241.显示最后更新的文件

有时您可能需要为某些操作查找最后更新的文件。下面的简单程序向我们展示了如何使用 awk 命令在 bash 中执行此操作。它将列出您当前工作目录中最后更新或创建的文件。

```
#!/bin/bash

ls -lrt | grep ^- | awk 'END{print $NF}'
```

242.批量添加扩展名

下面的示例将对目录中的所有文件应用自定义扩展名。创建一个新目录并将一些文件放在那里以进行演示。我的文件夹共有五个文件，每个文件名为 test 后跟 (0-4)。我已将此脚本编程为在文件末尾添加（.UP） 。你可以添加任何你想要的扩展。

```
#!/bin/bash
dir=$1
for file in `ls $1/*`
do
mv $file $file.UP
done
```

首先，不要从任何常规目录尝试此脚本；相反，从测试目录运行它。另外，您需要提供文件的目录名称作为命令行参数。对当前工作目录使用句点(.)。

243.打印文件或目录数量

查找给定目录中存在的文件或文件夹的数量。利用 Linux find 命令来执行此操作。首先，您需要传递目录名称以从命令行搜索文件。

```
#!/bin/bash

if [ -d "$@" ]; then
echo "Files found: $(find "$@" -type f | wc -l)"
echo "Folders found: $(find "$@" -type d | wc -l)"
else
echo "[ERROR] Please retry with another folder."
exit 1
fi
```

如果指定的目录不可用或有权限问题，程序将要求用户重试。

244.清理日志文件

删除 /var/log 目录中存在的所有日志文件。

```
#!/bin/bash
LOG_DIR=/var/log
cd $LOG_DIR

cat /dev/null > messages
cat /dev/null > wtmp
echo "Logs cleaned up."
```

245.备份文件和目录

Shell 脚本提供了一种强大的方法来备份您的文件和目录。以下示例将备份过去 24 小时内修改过的每个文件或目录。该程序利用 find 命令来执行此操作。

```
#!/bin/bash

BACKUPFILE=backup-$(date +%m-%d-%Y)
archive=${1:-$BACKUPFILE}

find . -mtime -1 -type f -print0 | xargs -0 tar rvf "$archive.tar"
echo "Directory $PWD backed up in archive file \"$archive.tar.gz\"."
exit 0
```

备份过程成功后，它将打印文件和目录的名称。

246..检查当前是不是root用户

```
#!/bin/bash
ROOT_UID=0

if [ "$UID" -eq "$ROOT_UID" ]
then
echo "You are root."
else
echo "你不是root"
fi
exit 0
```

247.从文件中删除重复的行

文件处理需要相当长的时间，并在许多方面影响管理员的工作效率。例如，在文件中搜索重复项可能会成为一项艰巨的任务。但是用shell脚本就简单多了。

```
#! /bin/sh

echo -n "Enter Filename-> "
read filename
if [ -f "$filename" ]; then
sort $filename | uniq | tee sorted.txt
else
echo "No $filename in $pwd...try again"
fi
exit 0
```

上面的脚本逐行通过您的文件并删除任何重复的行。然后它将新内容放入一个新文件中，并保持原始文件不变。

248.系统升级

```
#!/bin/bash

echo -e "\n$(date "+%d-%m-%Y --- %T") --- 开始工作\n"

apt-get update
apt-get -y upgrade

apt-get -y autoremove
apt-get autoclean

echo -e "\n$(date "+%T") \t 脚本终止"
```

请把上面的脚本改写成用yum实现。

209-248 来自：https://mp.weixin.qq.com/s/gerj94lRWajxXlrB2LMZYA

249.一键安装 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
```

250.一键申请多个证书 shell 脚本

```
[root@centos8 ~]#cat certs.sh
#!/bin/bash
#
#**********************************************************************
#Author:        wangxiaochun
#QQ:            29308620
#Date:          2020-02-29
#FileName:       test.sh
#URL:           http://www.wangxiaochun.com
#Description:       The test script
#Copyright (C):     2020 All rights reserved
#**********************************************************************
. /etc/init.d/functions

CERT_INFO=([00]="/O=heaven/CN=ca.god.com" \
           [01]="cakey.pem" \
           [02]="cacert.pem" \
           [03]=2048 \
           [04]=3650 \
           [05]=0    \

[10]="/C=CN/ST=hubei/L=wuhan/O=Central.Hospital/CN=master.liwenliang.org" \
           [11]="master.key" \
           [12]="master.crt" \
           [13]=2048 \
           [14]=365
           [15]=1 \
           [16]="master.csr" \

[20]="/C=CN/ST=hubei/L=wuhan/O=Central.Hospital/CN=slave.liwenliang.org" \
           [21]="slave.key" \
           [22]="slave.crt" \
           [23]=2048 \
           [24]=365 \
           [25]=2 \
           [26]="slave.csr"   )

COLOR="echo -e \\E[1;32m"
END="\\E[0m"
DIR=/data
cd $DIR

for i in {0..2};do
    if [ $i -eq 0 ] ;then
        openssl req  -x509 -newkey rsa:${CERT_INFO[${i}3]} -subj
${CERT_INFO[${i}0]} \
```

```
            -set_serial ${CERT_INFO[${i}5]} -keyout ${CERT_INFO[${i}1]} -nodes -
days ${CERT_INFO[${i}4]} \
            -out ${CERT_INFO[${i}2]} &>/dev/null

    else
        openssl req -newkey rsa:${CERT_INFO[${i}3]} -nodes -subj
${CERT_INFO[${i}0]} \
            -keyout ${CERT_INFO[${i}1]}   -out ${CERT_INFO[${i}6]} &>/dev/null

        openssl x509 -req -in ${CERT_INFO[${i}6]}  -CA ${CERT_INFO[02]} -CAkey
${CERT_INFO[01]}  \
            -set_serial ${CERT_INFO[${i}5]}  -days ${CERT_INFO[${i}4]} -out
${CERT_INFO[${i}2]} &>/dev/null
    fi
    $COLOR"**************************************生成证书信息
**************************************"$END
    openssl x509 -in ${CERT_INFO[${i}2]} -noout -subject -dates -serial
    echo
done
chmod 600 *.key
action "证书生成完成"
```

249-250来自[老王讲 IT (wangxiaochun.com)]