



# Linux 云计算-阶段 3

## 必备知识

达内 Linux 云计算学院

2017 年 7 月 22 日



# Shell 编程

## Day01

### 1. 编写脚本时，如何将错误的信息重定向导出，生成一份日志文件？

根据情况，看看是否需要覆盖或追加，选择 `2>` 或者 `2>>` 可以将错误信息重定向导出，从而生成一份日志文件。

重定向符号的使用详情如下表：

重定向符号	功能描述
<code>&gt;</code>	标准正确重定向（覆盖）
<code>2&gt;</code>	标准错误重定向（覆盖）
<code>&gt;&gt;</code>	标准正确重定向（追加）
<code>2&gt;&gt;</code>	标准错误重定向（追加）
<code>&amp;&gt;</code>	标准正确和错误重定向（覆盖）
<code>&lt;</code>	重定向导入

### 2. 简单描述执行脚本的不同方式及其区别（假设脚本名称为 `test.sh`）？

对于有执行权限的脚本可以使用相对路径或绝对路径执行

对于没有执行权限的脚本可以使用 `bash` 或 `source` 执行脚本

使用 `source` 执行脚本时不会开启子进程，其他方式会开启子进程

执行效果如下：

```
# ./test.sh           //需要脚本有可执行权限，执行脚本会开启子进程
# bash test.sh        //不需要脚本有可执行权限，执行脚本会开启子进程
# source test.sh      //执行脚本，不开启子进程
```

### 3. 编写 shell 脚本时，定义变量名有哪些规则要求？

变量名称可以使用数字、字母、下划线（`_`）

变量名称不能以数字开始，变量名称不可以使用特殊符号

### 4. 编写 shell 时，如何让脚本支持位置参数？

使用位置变量可以让脚本能够读取位置参数，如 `$1`, `$2`, `$3`, `$4` 等等。

### 5. Shell 如何判断上一个命令的执行结果是否成功？

使用 `$?` 可以判断上一个命令的执行结果是否成功，通常 `0` 代表成功，非 `0` 代表失败



## Day02

### 6. 请写出下面运算指令的执行结果分别是多少？

```
# i=5

# expr 3 + $i

# echo ${i*2}

# let i++; echo $i

# echo "scale=2;12/$i" |bc
```

结果依次为：

```
8
10
6
2.00
```

### 7. 判断 Linux 当前登录用户的数量，登录数量大于 3，则发送邮件报警，邮件内容在 mail.txt 文件中？

```
# num=$(who |wc -l)

# [ $num -gt 3 ] && mail -s Warning root < mail.txt
```

其他 Shell 支持的判断条件如下表：

判断符号	功能描述
==	测试字符串是否相同
!=	测试字符串是否不同
-z	测试字符串是否为空
-eq	测试数字是否相等
-ne	测试数字是否不相等
-gt	测试数字是否大于
-ge	测试数字是否大于等于
-lt	测试数字是否小于
-le	测试数字是否小于等于
-e	测试文件或目录是否存在
-f	测试是否存在且为文件
-d	测试是否存在且为目录



8. 编写脚本，判断当前用户是否为 root，如果是则安装 httpd 软件，否则提示“当前用户不是管理员”？

```
#!/bin/bash

if [ $USER == root ];then

    yum -y install httpd

else

    echo "当前用户不是管理员"

fi
```

## Day03

9. 编写脚本，循环读取用户输入的用户名和密码，判断用户名是否为 tom，密码是否为 123456，匹配则脚本提示登录成功，否则提示重试，失败 3 次则脚本退出？

for 循环的版本：

```
# cat test.sh

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 == "root" ] && [ $pass -eq 123456 ];then

        echo "登录成功"

        exit

    else

        echo "用户名或密码错误，请重新输入:"

    fi

done
```

while 循环的版本：

```
# cat test.sh

i=1

while [ $i -le 3 ]
```



```
do

    read -p "请输入用户名:"    user

    read -p "请输入密码:"      pass

    if [ $user == "root" ] && [ $pass -eq 123456 ];then

        echo "登录成功"

        exit

    else

        echo "用户名或密码错误, 请重新输入:"

    fi

    let i++

done
```

#### 10. 在编写 shell 脚本时 break 和 continue 命令有什么区别?

break 和 continue 都是用来中断循环语句的命令。break 会结束整个循环体; continue 仅结束当前循环, 跳入下一次循环中。

## Day04

#### 11. 编写一个生成 6 位随机密码的脚本?

```
# cat test.sh

#!/bin/bash

str="abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"

pass=""

for i in {1..6}

do

    num=$((RANDOM%62))

    tmp=${str:$num:1}

    pass=$pass$tmp

done
```



```
done
```

```
echo $pass
```

## 12. 请写出正则表达式的符号及其功能描述。

基本正则列表

正则符号	功能描述
<code>^</code>	匹配行首
<code>\$</code>	匹配行尾
<code>[]</code>	集合，匹配集合中的任意单个字符
<code>[^]</code>	对集合取反
<code>.</code>	匹配任意单个字符
<code>*</code>	匹配前一个字符任意次数
<code>\{n, m\}</code>	匹配前一个字符 $n$ 到 $m$ 次
<code>\{n\}</code>	匹配前一个字符 $n$ 次
<code>\{n, \}</code>	匹配前一个字符 $n$ 次以上
<code>\(\)</code>	保留

扩展正则列表

正则符号	描述
<code>+</code>	最少匹配一次
<code>?</code>	最多匹配一次
<code>\{n, m\}</code>	匹配 $n$ 到 $m$ 次
<code>()</code>	组合为整体，保留
<code> </code>	或者
<code>\b</code>	单词边界

Day05



13. 使用 sed 修改 httpd.conf 配置文件中的 DocumentRoot，将默认网站根路径从 /var/www/html 修改为/data/web?

```
# sed -i '/DocumentRoot/s#/var/www/html#/data/web#' /etc/httpd/conf/httpd.conf
```

其他 sed 常用指令列表如下：

指令	描述
a	追加新内容到文件
I	插入新内容到文件
c	修改内容（行修改）
s	替换内容（关键词替换）
d	删除（行删除）
p	打印

## Day6

14. 使用 awk 分析 Apache 日志，统计每个客户端 IP 访问服务器的次数？

```
# awk '{IP[$1]++} END{ for(i in IP){print i,IP[i]} }' /var/log/httpd/access_log
```

15. 如何使用 awk 提取 Linux 主机的参数信息，如 CPU 负载，内存剩余容量，根分区剩余容量，本机 IP 地址，本机能登录的用户个数等信息？

```
# df | awk '/\//{print $4}' //根分区剩余容量
# free | awk '/Mem/{print $4}' //内存剩余容量
# uptime | awk '{print $10}' //CPU 负载信息
# ifconfig enp3s0 | awk '/pack/{print $5}' //本机 IP 地址
# awk -F: '/bash$/ {x++} END{print "本机能登录的用户量:", x}' /etc/passwd
```



# 服务器运维技术

## Day01

### 1. squid 与 varnish 的差异是什么？

Squid 使用硬盘做缓存，性能和速度相对较差，但 squid 功能较全面，可以做正向代理，反向代理，透明代理。

Varnish 可以使用内存做缓存，性能和速度更高，但 varnish 只是一个反向代理服务器软件。

Squid 和 varnish 都属于代理软件，都可以通过缓存加速 HTTP 访问速度，使用代理将数据缓存在不同地区，结合 DNS 分离解析，实现不同客户访问不同的代理服务器，加速 Web 的访问速度。

## Day02

### 2. Nginx 如何实现网站的用户认证？

Nginx 默认就支持用户认证功能，在配置文件中添加 auth\_basic 和 auth\_basic\_user\_file 即可，其中 auth\_basic\_user\_file 定义了存储用户名和密码的文件名称，最后通过 htpasswd 命令创建对应的账户和密码即可。

```
# cat /usr/local/nginx/conf/nginx.conf
... ..

server {
    ... ..

    auth_basic "Prompt:";
    auth_basic_user_file "/usr/local/nginx/pass";           //指定密码文件
}

# htpasswd -cm /usr/local/nginx/pass tom                  //创建密码文件及账户
```

### 3. 制作 HTTPS 加密网站时，如何生成私钥和证书？

```
# openssl genrsa -out cert.key                               //生成私钥
# openssl req-new -x509 -key cert.key-out cert.pem          //生成证书
```





#### 4. Nginx 实现 HTTPS 加密网站，编译安装时需要启用什么模块

编译时使用 `./configure --with-http_ssl_module` 命令启动 `ngx_http_ssl_module` 模块。

#### 5. 配置 nginx 反向代理，实现 Web 集群调度器，应该使用什么语句？

Nginx 可以使用 `upstream` 定义集群，默认算法为轮询调度，可以修改为 `ip_hash` 和最少连接算法。

具体配置文件模板如下：

```
# cat /usr/local/nginx/conf/nginx.conf
... ..
http{
    upstream servers {
        #ip_hash;
        server 192.168.2.100 weight=2 max_fails=2 fail_timeout=30;
        server 192.168.2.200;

    server {
        listen 80;
        server_name localhost;
        location / {
            proxy_pass http://servers;
        }
    }
}
```

### Day03

#### 6. 创建 LNMP 环境所需要安装的软件包列表有哪些，她们的作用分别是什么？

Linux：操作系统、

Nginx：Web 服务器，接受用户的 HTTP 请求

mariadb-server：数据库服务器

mariadb：数据库客户端软件

mariadb-devel：依赖包

php：PHP 解释器



php-fpm: PHP 服务

php-mysql: PHP 连接 mysql 的扩展包。

## 7. Nginx 如何实现页面的动静分离?

Nginx 可以使用 location 匹配用户的请求, 根据正则表达式判断用户访问的是静态页面还是动态页面。

样例配置如下:

```
# cat /usr/local/nginx/conf/nginx.conf
```

```
... ..
```

```
server{  
    listen 80;  
    server_name localhost;  
    location / {                                //识别静态页面(除了 PHP 外的其他所有页面)  
        root html;  
        index index.html;  
    }  
    location ~* \.php$ {                        //识别动态页面  
        root html;  
        fastcgi_pass 127.0.0.1:9000;  
        include fastcgi.conf;  
    }  
}
```

## 8. Nginx 如何识别不同的浏览器返回不同的页面, 如 UC、IE 浏览器?

Nginx 可以使用 http\_user\_agent 识别用户的浏览器, 另外可以使用 rewrite 设置不同浏览器返回不同的页面。

样例配置如下:

```
rewrite ^/ http://www.tmooc.cn;                //访问任何页面跳转网页到 www.tmooc.cn  
if ($http_user_agent ~* msic){  
    rewrite ^/(.*) /msie/$1;                    //如果用户的浏览器是 MSIE, 则跳转页面  
}
```

## 9. Nginx 如何将所有对 a.html 页面的请求跳转到 b.html 页面?



```
rewrite a\.html b.html;
```

//访问 a.html 跳转到 b.html

## Day04

### 10. 说说你是如何优化 nginx?

我们可以从以下几个方面着手优化 nginx:

1. 优化并发数量 (修改 nginx 配置并且修改 Linux 系统配置)
2. 隐藏 nginx 服务器版本信息
3. 增加浏览器缓存机制
4. 启用页面压缩功能
5. 设计自定义报错页面

代码示例如下:

```
# ulimit -Hn 100000                                //max open files
# ulimit -Sn 100000                                //max open files
# cat /usr/local/nginx/conf/nginx.conf
worker_processes 2;                                //与 CPU 核心数量一致
worker_connections 60000;                          //每个 worker 最大并发连接数
server_tokens off;                                  //不显示 Nginx 具体版本号
location ~* \.(jpg|jpeg|gif|png|css|js|ico|xml)$ {
    expires 30d;                                     //静态数据缓存 30 天
}
error_page 404 /404.html;                          //自定义报错页面
```

### 11. Tomcat 默认端口号是多少?

8080 端口

### 12. 简单说明 tomcat 架构

Connector 负责监听端口接受用户的请求, 并将请求转发给 Engine, Engine 判断用户访问的目标网站, 将请求调度给具体的虚拟主机, <Host>定义虚拟主机名称、定义网站的根路径, 定义网站相关属性配置>



## Day05

### 13. 简单描述 memcached 的功能？

Memcached 是一款：缓存数据库、KV 数据库、noSQL 数据库，  
她可以缓存 SQL 数据库的数据，用来加速 SQL 数据库的访问速度，加速整体网站的访问速度，

使用 memcached 缓存可以共享网站的 Session 会话信息，  
PHP 连接 memcached 数据库是需要安装 php-pecl-memcache。

## Day06

### 14. Redis 与 memcached 的区别

1. Redis 支持持久化数据保存
2. Redis 支持更多的数据类型<字符型、hash 表、list 列表等>
3. Redis 软件本身支持主从同步

## Day07

### 15. 常用 subversion 命令：

# svnadmincreate /var/svn/project1	//创建仓库
# svnimport ./ file:///var/svn/project1/init.d -m "Init"	//导入代码到仓库
# svn co svn://10.47.214.131/ code	//将代码从服务器下载到本地副本
# svn ci -m "test"	//提交代码到服务器
# svn update	//更新本地副本上的代码
# svn log svn://10.47.214.131	//查看版本库的日志
# svn list svn://10.47.214.131	//列出服务器文件列表
# svn add test.sh	//新建文件
# svn mkdir test	//创建目录
# svn cat svn://10.47.214.131/test.sh	//查看服务器仓库中的代码文件内容
# svn diff	//对比本地副本与服务器仓库的差异



```
# svn revert test.sh
```

//回滚数据（还原数据）

```
# svn merge -r 10:5 test.sh
```

//还原数据到第 5 个版本

```
# svn co -r2 svn://10.47.214.131 code2
```

//下载特定版本的代码到本地

## 16. 将源码包转换为 RPM 需要什么工具：

```
# yum -y install rpm-build
```



# Python 开发技术

## Day01

### 1. 如何让 python 支持自动 tab 补齐的方法?

```
import readline
import rlcompleter
readline.parse_and_bind("tab:complete")
```

### 2. python 常见数据类型有哪些?

数字, 字符, 列表, 元组, 字典  
定义各种数据类型的操作示例如下:

```
# python
>>> test = 12345 //十进制数字
>>> test = 011 //八进制数字
>>> test = 0xff //十六进制数字
>>> test = 0b11 //二进制数字
>>> test = "hello the world" //字符
>>> test = ["tom", "jerry", 1, 8] //列表
>>> test = ("jerry", "tom", 88, 22) //元组
>>> test = {"name": "tom", "age": 11, "sex": "female"} //字典
>>> test[0] //提取数据
>>> test[:5]
>>> test[1:-2]
```

### 3. 使用 if 语句判断两个数字的大小?

```
# cat test.py
#!/usr/bin/python
```



```
A = 1  
  
B = 10  
  
If (A > B):  
    Print "A is bigger"  
  
else  
    Print "B is bigger"
```

## Day02

### 4. 使用 while 统计 1 到 100 的和

```
# cat test.py  
#!/usr/bin/python  
  
i=1  
  
sum=0  
  
while i<=100:  
    sum=sum+i  
  
    i+=1  
  
print "sum is %d" %sum
```

### 5. 使用 for 统计 1 到 100 的和

```
# cat test.py  
#!/usr/bin/python  
  
Sum=0  
  
for i in range(1,101):  
    sum=sum+i  
  
print "sum is %d" %sum
```

### 6. 使用 Python 对文件进行基本操作的指令有哪些？

常见操作包括：打开、读取、写入、关闭、更新数据等操作



```
# cat test.py

#!/usr/bin/python

data=open("/etc/hosts","r")           //只读打开文件

data.readline()                       //读取文件行

data.read()                           //读取文件剩余全部内容

data.seek(0)                          //重新定位文件行号

data.close()                          //关闭文件

data=open("/root/new.txt","w")        //可写方式新建文件

data.write("hello the world")         //写入数据到文件

data.writeline("hello python")        //写入数据到文件（可一次写入多行）

data.close()                          //关闭文件
```

## 7. python 如果定义和调用函数

```
# cat test.py

#!/usr/bin/python

def calc(x=1,y=2):                    //定义函数（设定形参）

    print x+y

calc(5,8)                             //调用函数（输入实参）

calc()                                //调用函数（使用默认参数）
```

## Day03

## 8. 简单描述 Python 常见错误有哪些？

NameError	未声明/初始化对象
IndexError	序列中没有没有此索引
SyntaxError	语法错误
KeyboardInterrupt	用户中断执行
EOFError	没有内建输入，到达 EOF 标记
IOError	输入/输出操作失败

## 9. 编写一个异常处理的案例？





```
# cat test.py
```

```
try:
```

```
    res = 10 / int(raw_input('input a number: '))           //捕获该指令的异常
```

```
except BaseException, e:
```

```
    print 'Error:', e                                       //如果遇到 BaseException 错误，则报错
```

```
else:
```

```
    print "OK"                                              //无异则都显示 OK
```

```
finally:
```

```
    print "Done"                                           //无论是否异常，都显示 Done
```

## 10. 使用线程模块实现 ping 一个网段，判断主机是否存活？

```
# cat mtping.py
```

```
#!/usr/bin/env python
```

```
import subprocess
```

```
import threading
```

```
import sys
```

```
def ping(ip):
```

```
    result = subprocess.call("ping -c2 %s &> /dev/null" % ip, shell=True)
```

```
    if result:
```

```
        print "%s:down" % ip
```

```
    else:
```

```
        print "%s:up" % ip
```

```
if __name__ == '__main__':
```

```
    if len(sys.argv) != 2:
```

```
        print "Usage: %s subnet" % sys.argv[0]
```

```
        sys.exit(1)
```

```
    net_list = sys.argv[1].split('.')
```



```
net = '.'.join(net_list[:-1])  
  
ips = ("%s.%s" % (net, i) for i in range(1, 255))  
  
for ip in ips:  
  
    t = threading.Thread(target=ping, args=(ip,))  
  
    t.start()
```