## 02/ shell编程

Shell是用户与内核进行交互操作的一种接口，目前最流行的Shell称为bash Shell

Shell也是一门编程语言**<解释型的编程语言>**，即shell脚本<就是在用linux的shell命令编程>

一个系统可以存在多个shell，可以通过cat /etc/shells命令查看系统中安装的shell，不同的shell可能支持的命令语法是不相同的

### 2.1 基本格式

代码写在普通文本文件中，通常以 .sh为后缀名

vi hello.sh

|  |
| --- |
| #!/bin/bash ## 表示用哪一种shell解析器来解析执行我们的这个脚本程序  echo "hello world" ## 注释也可以写在这里  ## 这是一行注释 |

执行脚本

sh hello.sh

或给脚本添加x权限，直接执行

chmod 755 hello.sh

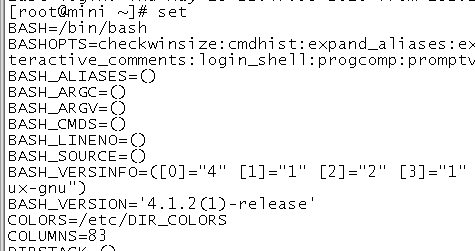
./hello.sh

### 2.2 基本语法

#### 2.2.1 系统变量

Linux Shell中的变量分为“系统变量”和“用户自定义变量”

可以通过set命令查看系统变量



系统变量：$HOME、$PWD、$SHELL、$USER等等

#### 2.2.2 自定义变量

**1、语法**

变量＝值 （例如STR=abc）

等号两侧不能有空格

变量名称一般习惯为大写

使用变量： $arg

双引号和单引号有区别，

双引号仅将空格脱意，

[root@hadoop01 soft]# a="heiehi hei"

[root@hadoop01 soft]# b="$a"

[root@hadoop01 soft]# echo $b

heiehi hei

[root@hadoop01 soft]#

单引号会将变量引用比如$param脱意

[root@hadoop01 soft]# a="heiehi hei"

[root@hadoop01 soft]# b='$a'

[root@hadoop01 soft]# echo $b

$a

**2、示例**

STR="hello world"

A=9

echo $A

echo $STR

如果想打印 hello worlds is greater 怎么办？

*echo $STRs is greate 行吗？*

不行，正确写法是：

echo ${*STR}s is greate*

unset A 撤销变量 A

readonly B=2 声明静态的变量 B=2，不能 unset

export A #可把变量提升为当前shell进程中的全局环境变量，可供其他子shell程序使用

注意理解export：

|  |
| --- |
| [root@shizhan01 scripts]# vi a.sh  #!/bin/bash  a="a in a.sh"  echo $a  /root/scripts/b.sh |

|  |
| --- |
| [root@shizhan01 scripts]# vi b.sh  #!/bin/bash  b="b in b.sh"  echo $b  echo $a |

然后执行 ./a.sh ，会发现 b脚本中并没有把a脚本中定义的a变量打印出来

如果要在b中打印出a脚本的变量a，需要在a脚本中把变量a做export定义

此时，a变量就成了a.sh脚本所在bash进程的全局变量，该进程的所有子进程都能访问到变量a

另一种方式：

如果在a.sh脚本中用如下方式调用b.sh

|  |
| --- |
| . ./b.sh ## 注意：重点关注最前面那个 “.”号 |

或者

|  |
| --- |
| source ./b.sh ## |

则，b.sh就在a.sh所在的bash进程空间中运行

**总结：**

**1、a.sh中直接调用b.sh，会让b.sh在a所在的bash进程的“子进程”空间中执行**

**2、而子进程空间只能访问父进程中用export定义的变量**

**3、一个shell进程无法将自己定义的变量提升到父进程空间中去**

**4、“.”号执行脚本时，会让脚本在调用者所在的shell进程空间中执行**

**3、反引号赋值**

A=`ls -la` ## 反引号，运行里面的命令，并把结果返回给变量A

A=$(ls -la) ## 等价于反引号

**4、特殊变量**

$? 表示上一个命令退出的状态码

$$ 表示当前进程编号

$0 表示当前脚本名称

$n 表示n位置的输入参数（n代表数字，n>=1）

$# 表示参数的个数，常用于循环

$\*和$@ 都表示参数列表  .`

***注：$\*与$@区别***

$\* 和 $@ 都表示传递给函数或脚本的所有参数

* 不被双引号""包含时——

$\* 和 $@ 都以$1 $2 … $n 的形式组成参数列表

* 当它们被双引号""包含时——

"$\*" 会将所有的参数作为一个整体，以**"**$1 $2 … $n**"**的形式组成一个整串；

"$@" 会将各个参数分开，以"$1""$2" … "$n" 的形式组成一个参数列表

### 2.3 运算符

#### 2.3.1 算数运算

##### 1、用expr

格式 expr m + n 或$((m+n)) 注意expr运算符间要有空格

例如计算（2＋3 ）×4 的值

1 .分步计算  
 S=`expr 2 + 3`  
 expr $S \\* 4 ## \*号需要转义

2.一步完成计算

expr `expr 2 + 3 ` \\* 4

echo `expr \`expr 2 + 3\` \\* 4`

##### 用(())

Centos7好像不支持这一种语法

[root@hadoop01 soft]# h=((1 + 2))

-bash: syntax error near unexpected token `('

((1+2))

(((2+3)\*4))

count=1

((count++))

echo $count

但是要想取到运算结果，需要用$引用

a=$((1+2))

##### 3、用$[]

a=$[1+2]

echo $a

### 2.5 流程控制

#### 2.5.1 if语法

##### 1、语法格式

if condition   
then   
    statements   
[elif condition   
    then statements. ..]   
[else   
    statements ]   
fi

##### 2、示例

|  |
| --- |
| #!/bin/bash  read -p "please input your name:" NAME ## read命令用于从控制台读取输入数据  ## printf '%s\n' $NAME  if [ $NAME = root ]  then  echo "hello ${NAME}, welcome !"  elif [ $NAME = itcast ]  then  echo "hello ${NAME}, welcome !"  else  echo "SB, get out here !"  fi |

##### 3、判断条件

1/ 条件判断基本语法

[ condition ]  *(注意condition前后要有空格)*

#非空返回true，可使用$?验证（0为true，>1为false）

[ itcast ]

#空返回false

[ ]

注意[ ]内部的=周边的空格，有区别：

|  |
| --- |
| [root@shizhan01 scripts]# if [ a = b ];then echo ok;else echo notok;fi  notok  [root@shizhan01 scripts]# if [ a=b ];then echo ok;else echo notok;fi  ok |

短路（理解为三元运算符）

[ condition ] && echo OK || echo notok

条件满足，执行&&后面的语句；条件不满足，执行|| 后面的语句

2/ 条件判断组合

注：[] 与[[ ]] 的区别：[[ ]] 中逻辑组合可以使用 && || 符号

而[] 里面逻辑组合可以用 -a -o

|  |
| --- |
| [root@mini ~]# if [ a = b && b = c ]; then echo ok;else echo notok;fi  *-bash: [: missing `]'*  *notok*  [root@mini ~]# if [ a = b -a b = b ]; then echo ok;else echo notok;fi  *notok*  [root@mini ~]# if [ a = b -o b = b ]; then echo ok;else echo notok;fi  *ok*  [root@mini ~]# if [[ a = b && b = b ]]; then echo ok;else echo notok;fi  *notok*  [root@mini ~]# if [[ a = b || b = b ]]; then echo ok;else echo notok;fi  *ok* |

3/ 常用判断运算符

**字符串比较**：= !=

-z 字符串长度是为0返回true

-n 字符串长度是不为0返回true

if [ 'aa' = 'bb' ]; then echo ok; else echo notok;fi

if [ -n "aa" ]; then echo ok; else echo notok;fi

if [ -z "" ]; then echo ok; else echo notok;fi

**整数比较：**

-lt 小于

-le 小于等于

-eq 等于

-gt 大于

-ge 大于等于

-ne 不等于

**文件判断：**

-d 是否为目录

if [ -d /bin ]; then echo ok; else echo notok;fi

-f 是否为文件

*if [ -f /bin/ls ]; then echo ok; else echo notok;fi*

-e 是否存在

*if [ -e /bin/ls ]; then echo ok; else echo notok;fi*

#### 2.5.2 while语法

##### 1、方式一

while expression

do

command

…

done

##### 2、方式二

i=1

while ((i<=3))

do

echo $i

let i++

done

#### 2.5.3 case语法

case $1 in

start)

echo "starting"

;;

stop)

echo "stoping"

;;

\*)

echo "Usage: {start|stop}"

esac

#### 2.5.4 for语法

##### 1、方式一

for N in 1 2 3

do

echo $N

done

或

for N in 1 2 3; do echo $N; done

或

for N in {1..3}; do echo $N; done

##### 2、方式二

for ((i = 0; i <= 5; i++))

do

echo "welcome $i times"

done

或

for ((i = 0; i <= 5; i++)); do echo "welcome $i times"; done

### 2.6 函数使用

#### 2.6.1 函数定义

|  |
| --- |
| #!/bin/sh  # func1.sh  hello() ## 函数定义  {  echo "Hello there today's date is `date +%Y-%m-%d`"  # return 2 ###返回值其实是状态码，只能在[0-255]范围内  }  hello  # echo $? 获取函数的return值  echo "now going to the function hello"  echo "back from the function" |

函数调用：

function hello()

或 function hello

或 hello

*注意：*

*1.必须在调用函数地方之前，先声明函数，shell脚本是逐行运行。不会像其它语言一样先预编译*

*2.函数返回值，只能通过$? 系统变量获得，可以显示加：return 返回，如果不加，将以最后一条命令运行结果，作为返回值。 return后跟数值n(0-255)*

*~~脚本调试：~~*

*sh -vx helloWorld.sh*

*或者在脚本中增加set -x*

#### 2.6.2 函数参数

|  |
| --- |
| #!/bin/bash  # fun1.sh  funWithParam(){  echo "第一个参数为 $1 !"  echo "第二个参数为 $2 !"  echo "第十个参数为 $10 !"  echo "第十个参数为 ${10} !"  echo "第十一个参数为 ${11} !"  echo "参数总数有 $# 个!"  echo "作为一个字符串输出所有参数 $\* !"  }  funWithParam 1 2 3 4 5 6 7 8 9 34 73 |

*注意，$10 不能获取第十个参数，获取第十个参数需要${10}。当n>=10时，需要使用${n}*

*1~9之间也可以使用${n}来获取对应的值这样只需要记着取值为${n}即可*

*来获取参数。*

#### 2.6.3 函数返回值

|  |
| --- |
| #!/bin/bash  # fun2.sh  funWithReturn(){  echo "这个函数会对输入的两个数字进行相加运算..."  echo "输入第一个数字: "  read aNum  echo "输入第二个数字: "  read anotherNum  echo "两个数字分别为 $aNum 和 $anotherNum !"  return $(($aNum+$anotherNum))  }  funWithReturn  echo "输入的两个数字之和为 $? !" |

#### 2.6.4 跨脚本调用函数

假如上述的脚本文件fun2.sh保存在此路径： /root/fun2.sh

则可在脚本fun\_other.sh中调用脚本fun2.sh中的函数

|  |
| --- |
| #!/bin/bash  # fun\_other.sh  . /root/fun2.sh ## 注： **.** 和 **/** 之间有空格  # 或者 source /root/fun2.sh  funWithParam 11 22 33 44 55 66 77 88 99 100 101 |

## 03/ shell编程综合练习

自动化软件部署脚本

### 3.1 需求

#### 1、需求描述

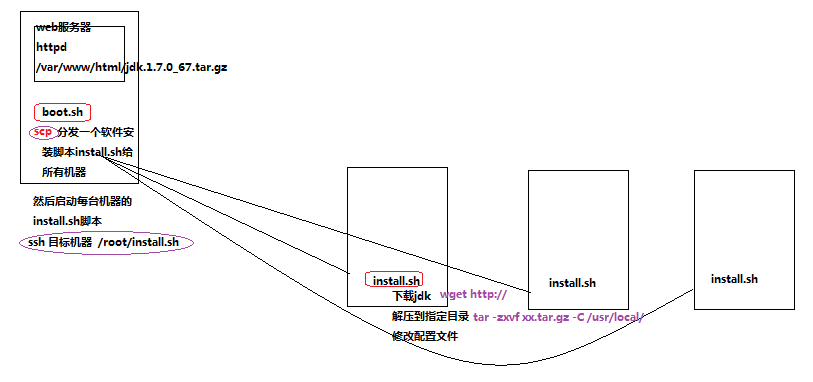
公司内有一个N个节点的集群，需要统一安装一些软件（jdk）

需要开发一个脚本，实现对集群中的N台节点批量自动下载、安装jdk

#### 2、思路

1/ 编写一个**启动脚本**，用来发送一个软件**安装脚本**到每一台机器

2/ 然后启动每台机器上的软件安装脚本来执行软件下载和安装



#### 3、expect的使用

痛点：使用scp命令远程拷贝文件时，会有人机交互的过程，如何让脚本完成人机交互？

妙药： expect

用法示例：

先观察 ssh localhost 的过程

再看expect的功能

|  |
| --- |
| #!/bin/bash/expect  ## exp\_test.sh  set timeout -1;  spawn ssh localhost;  expect {  "(yes/no)" {send "yes\r";exp\_continue;}  "password:" {send "hadoop\r";exp\_continue;}  eof {exit 0;}  } |

*执行： expect -f exp\_test.sh*

### 3.2 准备内网软件下载服务器

选择一台服务器（比如mini）作为软件源服务器

1、安装httpd*（如果已有，可跳过）*

yum install -y httpd

service httpd start

chkconfig --level 35 httpd on

#### 2、制作局域网yum源

1/ 挂载centos安装光盘到/mnt/cdrom *（如果已有，可跳过）*

mkdir /mnt/cdrom

mount -t iso9660 -o loop /dev/cdrom /mnt/cdrom

2/ 将本地yum库放入httpd服务器

ln -s /mnt/cdrom /var/www/html/centos

*检查点：用浏览器访问 http://mini/centos 看能否看到光盘内容*

#### 3、编写repo配置

vi /etc/yum.repos.d/centos.repo

|  |
| --- |
| [c6-httpd]  name=CentOS-httpd  baseurl=http://192.168.33.3/centos  gpgcheck=0  enabled=1 |

#### 4、分发repo配置到局域网

从母鸡shizhan01上把centos.repo拷贝给所有需要自动安装软件的服务器（仔鸡mini1/mini2）

cd /etc/yum.repos.d/

scp /etc/yum.repos.d/innet.repo mini1:$PWD

scp /etc/yum.repos.d/innet.repo mini2:$PWD

#### 5、准备一个jdk安装包放在内网web服务器上

### 3.3 脚本开发

#### 1、启动脚本

vi boot.sh

|  |
| --- |
| #!/bin/bash  SERVERS="mini1 mini2"  PASSWORD=hadoop  BASE\_SERVER=192.168.33.11  ## 实现免密登陆配置的函数  auto\_ssh\_copy\_id() {  expect -c "set timeout -1;  spawn ssh-copy-id $1;  expect {  \*(yes/no)\* {send -- yes\r;exp\_continue;}  \*assword:\* {send -- $2\r;exp\_continue;}  eof {exit 0;}  }";  }  ssh\_copy\_id\_to\_all() {  for SERVER in $SERVERS  do  auto\_ssh\_copy\_id $SERVER $PASSWORD  done  }  ## 调用免密登陆配置函数，实现母鸡到各仔鸡的免密登陆配置  ssh\_copy\_id\_to\_all  ## 完成分发install.sh到各仔鸡的操作  ## 并让仔鸡启动install.sh  for SERVER in $SERVERS  do  scp install.sh root@$SERVER:/root  ssh root@$SERVER /root/install.sh  done |

#### 2、安装执行脚本

vi install.sh

|  |
| --- |
| #!/bin/bash  BASE\_SERVER=192.168.33.11  ## 为本机安装wget命令  yum install -y wget  ## 使用wget从母鸡的web服务器上下载jdk压缩包  wget $BASE\_SERVER/soft/jdk-7u67-linux-x64.gz  ## 将下载的压缩包解压  tar -zxvf jdk-7u67-linux-x64.gz -C /usr/local  ## 修改profile配置文件  cat >> /etc/profile << EOF  export JAVA\_HOME=/usr/local/jdk1.7.0\_67  export PATH=\$PATH:\$JAVA\_HOME/bin  EOF |

#### 3、启动脚本

只要在baseServer即mini上启动boot.sh即可

#### 注意事项:

我的机器是centos7在安装jdk时解压什么都正确但是运行的是openjdk,此时需要将所有的openjdk都删除掉才能运行自己的jdk http://blog.csdn.net/jzp630641095/article/details/72743000

查看java版本时出现openjdk

[root@hadoop01 bin]# java -version

openjdk version "1.8.0\_102"

OpenJDK Runtime Environment (build 1.8.0\_102-b14)

OpenJDK 64-Bit Server VM (build 25.102-b14, mixed mode)

查看所有的openjdk版本

[root@hadoop01 bin]# rpm -qa|grep java

java-1.8.0-openjdk-1.8.0.102-4.b14.el7.x86\_64

tzdata-java-2016g-2.el7.noarch

javapackages-tools-3.4.1-11.el7.noarch

java-1.8.0-openjdk-headless-1.8.0.102-4.b14.el7.x86\_64

java-1.7.0-openjdk-headless-1.7.0.111-2.6.7.8.el7.x86\_64

java-1.7.0-openjdk-1.7.0.111-2.6.7.8.el7.x86\_64

python-javapackages-3.4.1-11.el7.noarch

删除依赖

[root@hadoop01 bin]# rpm -e --nodeps java-1.8.0-openjdk-1.8.0.102-4.b14.el7.x86\_64

最终只剩下

[root@hadoop01 bin]# rpm -qa|grep java

tzdata-java-2016g-2.el7.noarch

此时运行自己的jdk

[root@hadoop01 bin]# java -version

java version "1.8.0\_161"

Java(TM) SE Runtime Environment (build 1.8.0\_161-b12)

Java HotSpot(TM) 64-Bit Server VM (build 25.161-b12, mixed mode)

### 作业：

批量部署脚本，存在一些不太完美的地方，比如，目标机器名需要写死在脚本中，请改造这一点：

可以将所有需要安装软件的机器名写在一个文件：比如slaves中

让脚本自动读取slaves文件中的机器名来批量安装

cat slaves | while read host

do

echo $host

expect -c "set timeout -f

spawn ssh-copy-id $host"

done