Shell脚本

Shell Script，Shell[脚本](http://baike.baidu.com/view/54.htm)与Windows/Dos下的[批处理](http://baike.baidu.com/view/80110.htm)相似，也就是用各类命令预先放入到一个文件中，方便一次性执行的一个[程序文件](http://baike.baidu.com/view/600314.htm)，主要是方便[管理员](http://baike.baidu.com/view/315045.htm)进行设置或者管理用的。但是它比Windows下的批处理更强大，比用其他编程[程序编辑](http://baike.baidu.com/view/5250042.htm)的程序效率更高，毕竟它使用了Linux/Unix下的命令。

换一种说法也就是，shell script是利用shell的功能所写的一个程序，这个程序是使用[纯文本文件](http://baike.baidu.com/view/8752736.htm)，将一些shell的语法与指令写在里面，然后用正规表示法，管线命令以及数据流重导向等功能，以达到我们所想要的处理目的

**shell和shell脚本的区别：**

shell是什么呢？确切一点说，Shell就是一个[命令行解释器](http://baike.baidu.com/view/441296.htm)，它的作用就是遵循一定的语法将输入的命令加以解释并传给系统。它为用户提供了一个向Linux发送请求以便运行程序的接口系统级程序，用户可以用Shell来启动、挂起、停止甚至是编写一些程序。 Shell本身是一个用C语言编写的程序，它是用户使用Linux的桥梁。Shell既是一种命令语言，又是一种[程序设计语言](http://baike.baidu.com/view/128511.htm)(就是你所说的shell脚本)。作为命令语言，它互动式地解释和执行用户输入的命令；作为程序设计语言，它定义了各种[变量](http://baike.baidu.com/view/296689.htm)和参数，并提供了许多在高阶语言中才具有的控制结构，包括循环和分支。它虽然不是 Linux系统[内核](http://baike.baidu.com/view/1366.htm)的一部分，但它调用了系统内核的大部分功能来执行程序、创建文档并以并行的方式协调各个程序的运行。

**交互式shell和非交互式shell**

　　交互式模式就是shell等待你的输入，并且执行你提交的命令。这种模式被称作交互式是因为shell与用户进行交互。这种模式也是大多数用户非常熟悉的：登录、执行一些命令、签退。当你签退后，shell也终止了。

　　shell也可以运行在另外一种模式：非交互式模式。在这种模式下，shell不与你进行交互，而是读取存放在文件中的命令，并且执行它们。当它读到文件的结尾，shell也就终止了。

如下:

简单的实现系统巡检的命令:

date

free –m

df –Th

写成一个简单的脚本test.sh

#!/bin/bash

date

free –m

df –Th

chmod +x test.sh

./test.sh

一.基础知识

1.shell脚本的格式注意事项

2.shell脚本文件的扩展名

3.shell脚本执行顺序以及产生后果(如出现错误)

a /tmp/test

rm –rf $a/\*

4.用户身份的不同执行脚本的区别

5.shell种类的介绍(nologin和锁定区别)及用户切换切换shell方式(bash chsh -l)

6.历史命令的介绍history

!! !100 !ser !$

7.Shell退出时执行的命令.bash\_logout

8.别名的介绍alias(以及命令的回顾)

二.变量

1.环境变量

环境[变量](http://baike.baidu.com/view/296689.htm)是一个具有特定名字的对象，它包含了一个或者多个应用[程序](http://baike.baidu.com/view/17674.htm)所将使用到的信息。例如path，当要求系统运行一个程序而没有告诉它程序所在的完整路径时，系统除了在[当前目录](http://baike.baidu.com/view/396351.htm)下面寻找此程序外，还应到path中指定的路径去找。用户通过设置环境变量，来更好的运行进程

环境变量:系统在启动过程中预先指定好的一系列的变量.比如当前用户是谁 当前shell是什么 当前用户的家目录在什么位置等等

2.预定义变量

预定义变量:系统预定义好的 和进程名称 进程编号 进程参数 进程返回值相关

3.位置变量:和命令行参数相关

4.自定义变量(用户自己定义的变量)

第一类 环境变量

环境变量 echo

env 查看所有环境变量

echo $变量名 输出变量

PATH

USER

HOME

HOSTNAME

PWD

UID

PS1

LANG=zh\_CN.UTF8 (setup,yum groupinstall.系统时间)

set 查看所有变量(包括环境变量和非环境变量)

非环境变量:为用户定义的变量

第二类 预定义变量$0 $$ $# $? $\*

$0 进程名(如:/etc/init.d/network)

$$ 进程号(/var/run 模拟系统结束进程)

$# 位置参数的数量

$\* 所有位置参数的内容

$? 命令执行后的返回状态.0为执行正确，非0为执行错误

第三类 位置变量

位置变量: 和命令行参数相关 (命令后跟的参数$1-$9)

第四类自定义变量

自定义变量:当用户变量不够用时,自定义的变量

如下测试脚本

#!/bin/bash

cd /tmp

touch a.txt

ls –ld /tmp

du –sh /tmp

改为

#!/bin/bash

$DIR=/tmp

cd $DIR

touch a.txt

ls –ld $DIR

du –sh $DIR

再进一步完善脚本read 命令的使用

算式运算符:

+、-、\*、/、()

$((5+3))

expr操作符：

+、-、\\*、/、%取余（取模）

expr 1 + 2

a=1;b=2

c=$a\*$b

echo $c

export作用范围

父子shell的说明,及变量的定义

如例子:

a.sh

#!/bin/bash

echo IN a.sh’

aa=123

./b.sh

b.sh

#!/bin/bash

echo ‘IN b.sh’

echo $aa

运行脚本方式的介绍

./a.sh

bash a.sh

source a.sh

. a.sh

source a.sh 和. a.sh

利用当前的shell执行后边的脚本 如果没有外部命令 则不产生子进程

如测试脚本如下

cat aa.sh

#!/bin/bash

cd /

pwd

./ aa.sh

. aa.sh

函数库定义

函数库:将常用的变量定义到一个文件里 直接加载这个文件 就不用重复定义变量了

如系统中的确定与失败

子进程定义的变量能否被父进程集成?

nologin shell 和 login shell

/etc/bashrc

/etc/bashrc

~/.bashrc

~/.bash\_profile

read命令的使用

read命令:将脚本后边跟着的变量的值读取到脚本中

-p –t 参数的说明

如下边这个有趣的脚本:

#!/bin/bash

read -p "请输入银行卡账号: " num

read -p "请输入银行卡密码: " -t 5 pass

echo 账号$num 密码$pass >> /tmp/haha.txt

算式置换

a=10+20

a=$((10+20))

命令置换(将命令执行结果赋给变量)

a=`date +%m%d`

a=$(date +%m%d) 推荐

原因

a=`ls `date +%m%d`` 该赋值失败

a=$(ls $(date +%m%d ))

通配符

通配符是shell解释的 正则表达式是命令解释的

\*匹配任所有字符

?匹配一个字符

[]匹配一个范围

{}如touch abc{a,b,c}{1..3}.txt

变量的引用

echo 命令介绍

echo -n -e参数说明 “”’’说明 \n \t

echo \$

echo ’$’

test命令的使用

语法:test EXPRESSION 或者 [ EXPRESSION ]

字符串判断(用于看用户有没有输入指定的变量 指定用户输入目录 如没填)

-n 字符段长度是否非零的 如果结果为真值 返回值为0 如果结果为假值返回值非0

例:判断两个文件名字是否一致

mkdir /a /b

touch /a/x.txt /b/x.txt

[ "/a/x.txt"="/b/x.txt"] 可定错误不同 目录名称不同

应为

[ "$(basename /a/x.txt)"="$(basename /b/x.txt)" ]

test 整数

eq 等于

ge 大于等于

gt 大于

le 小于等于

lt 小于

ne 不等于

test 文件

ef 两个文件有相同的设备编号和inode编号 (判断硬链接)

touch aa

ln aa bb

ls -i

456733 aa 456733 bb

根据文件类型判断

-d 文件存在必须是个目录

-e 有文件名存在就行不管类型

-f 文件存在而且是个标准普通文件

-h 文件存在并且是否为符号链接文件

-r 判断文件权限是否有r权限

-w 写权限

-x 执行权限

条件判断语句

if cmd;如为真值

then

fi 执行

如为假值则不执行

例

if [ -f /etc/passwd ]

then

echo ok

fi

若文件不存 则 不执行

if useradd uu3

then

id uu3

fi

添加成功则显示用户信息

if [ -f /etc/ssh/sshd\_config ]

then

service sshd start

else

echo ssh is not install

fi

练习题:

复习白天讲的脚本

1.判断当前用户是否为root 如果为root用户启动ssh服务 如果非root

用户提示用户启动服务

2.每隔3秒调用自己一次

3.分析下边脚本

a.sh

#!/bin/bash

echo $$

./a.sh | ./a.sh &

4.判断自己是否为重复运行脚本,如果为重复运行的脚本则自动退出(同一

时间该脚本只有一个实例运行)