双星教育
Linux 系统管理员

Shell 语法篇

Terry Tsang
曾绍坤
signmem@netease.com

回顾

变量定义
grep find 命令
cut tr sort uniq
$? $$ $0 $1

变量

全局变量
局部变量
变量的使用
Shell 中变量不区分类型

脚本优势

一个脚本应该无错运行。
它应该完成他要完成的任务。
程序的逻辑结构定义清晰而且明显。
一个脚本不做不必要的工作。
脚本可以重用。

Shell 语法

#!/bin/bash （megic number)的作用

#!/bin/tcsh

如何进行赋值

如何进行输出

如何运行一个脚本

注意下面的三个例子

```
${char:-linux}
${char:+linux}
${char:=linux}
```

数学运算

Shell 同样支持数学运算

+ - * / %

var++ , var--

++var, --var

&&, ||, ;

~+ = pwd

~- = oldpwd

^ 平方

expr

expr length 可以获取字符长度
expr substr 可以截取字符
expr index 第一个出现的字符位置计算

转译字符

echo -e (调用转译字符)
\a alert (bell)
\b backspace
\e an escape character
\f form feed
\n new line
\r carriage return
\t horizontal tab
\v vertical tab
\\ backslash
\’ single quote
\nnn the eight-bit character whose value is the octal value nnn (one to three digits)
\xHH the eight-bit character whose value is the hexadecimal value HH (on or two hex digits)
\cx a control-x character
EOF

数组

数组定义

number=(1 2 3 4 7)

color=(black [4]=red [8]=white [5]=green)

数组调用

${number[*]}

${color[4]}

思考 ${number[0]} 代表什么？

变相数组录入 number=($number)

read

read 外部变量读入

判断

```
if   [ 条件 ]
     then
          执行语句
     else
          执行语句
fi
----------------------------------------
if  [ 条件 ]
     then
          执行语句
     elif [ 条件 ]
          then
                    执行语句
          else
                    执行语句
     fi
fi
```

条件判断语法

数字判断方法	字符判断方法
-eq	==
-ne	!=
-lt	<
-le	<=
-gt	>
-ge	>=

-z 变量为 null

文件判断

-e 文件存在
-s 文件字节不为0

-r 读权限
-w 写权限
-x 执行权限

-g setGID
-u setUID
-k sticty bit

-O owner
-G gorupID
-N 是否被修改

-f 文件
-d 目录
-b 块设备
-c 字符设备
-p 管道
-L 符号连接
-S socket
f1 -nt f2 (f1 文件新)
f1 -ot f2 (f1 文件老)
f1 -ef f2 (两个文件 inode 相同)

test 文件判断

-d 目录
-F 文件
-s 非空文件

-r 可读
-w 可写
-x 可执行

-u SUID
-g SGID
-G UID是否合法
-O UID 是否合法

-b 字符文件
-c 设备文件
-L 链接
-S socket 文件
-p 管道文件

多次判断

if [条件1] && [条件2]
if [条件1 -a 条件2]

if [条件1] || [条件2]
if [条件1 -o 条件2]

颜色输出

#echo -en "\\033[0;31m" <---> red
#echo -en "\\033[0;32m" <---> green
#echo -en "\\033[0;33m" <---> brown
#echo -en "\\033[0;34m" <---> navy blue
#echo -en "\\033[0;35m" <---> violet
#echo -en "\\033[0;36m" <---> powder blue
#echo -en "\\033[0;37m" <---> grey
#echo -en "\\033[0;38m" <---> black and underline
#echo -en "\\033[0;39m" <---> black
#echo -en "\\033[0;40m" <---> black background
#echo -en "\\033[0;41m" <---> red background
#echo -en "\\033[0;42m" <---> green background
#echo -en "\\033[0;43m" <---> brown background
#echo -en "\\033[0;44m" <---> navy blue background
#echo -en "\\033[0;45m" <---> violet background
#echo -en "\\033[0;46m" <---> powder blue background
#echo -en "\\033[0;47m" <---> grey background

通过脚本实现下面练习

1判断登陆用户，如果管理员登陆显示红色文字，普通用户现实蓝色文字

2备份 /etc(不包含子目录) 下所有配置文件(.conf)到 /backup/etc 目录下

3定制傻瓜闹钟程序，允许用户输入时间格式[hh:mm]，当到达时间后以输出 ‘time’s up!’

4计算世界模式距离现在有多少天?(2012-12-21)

循环

```
while  [  条件  ]
do
        执行语句
done
```

```
until [ 条件 ]
do
        执行语句
done
```

```
for  (( 条件1; 条件2;  条件3 ))
do
        执行语句
done
```

```
for  变量  in  条件
do
        执行语句
done
```

变量递增

let a++

let a+=1

a=`expr $a+1`

a=$(($a+1))

循环嵌套

```
for  变量  in 条件
do
        执行语句
                for (( 条件1;条件2;条件3; ))
                do
                        执行语句
                done
        执行语句
done
```

continue and break

如何使用 continue 及 break

case

```
case  条件  in
1)  执行语句
;;
2) 执行语句
;;
3) 执行语句
;;
*)  执行语句
;;
esac
```

函数

思考: 为什么要使用函数?
函数的应用

常用语法
sleep, let

介绍
脚本之间的嵌套

伪随机

RANDOM

如何控制随机数范围

/dev/random

/dev/urandom

循环练习

```
*
**
***
****
*****
```

```
1
12
123
1234
12345
```

```
1
23
345
4567
56789
```

```
1
23
456
78910
```

练习

1通过脚本添加20个用户，密码与用户名一样（用户名放在文件 user 中)
2对所有普通用户进行设定，要求每个用户下次登陆之后重新设定密码
3截取当前网络卡 eth0 的 IP
4如何检测 $JAVA_HOME 变量是否null 值
5要求输入用户名，并通过脚本自动搜索改用户是否存在系统中
6如果用户mary登陆，马上给管理员发送邮件

练习

网络检测脚本

要求判断网络当前是否正常 (是否掉线，是否能够连接网关，是否能够连接互联网, DNS 是否正常)

按照下面的方法进行输出

使用蓝色字体输出正常部分

网卡连接正常

IP 使用正常，当前 IP 是:

网关连接正常

互联网连接正常

DNS 正常

使用红色字体输出不正常部分

网卡连接不正常

IP 使用不正常，当前 IP 是:

网关连接不正常

互联网连接不正常

DNS 不正常

非常感谢