

shell介绍

2019年8月3日 莫宇剑 著 10:47

Shell是系统的用户界面，提供了用户与内核进行交互操作的一种接口。它接收用户输入的命令并把它送入内核去执行。Shell是一个命令解释器，它解释由用户输入的命令并且把它们送到内核。Shell允许用户编写由shell命令组成的程序（shell脚本）。

Shell虽然没有 C++、Java、Python 等强大，但也支持了基本的编程元素，例如：

- > if...else 选择结构，case...in 开关语句，for、while、until 循环；
- > 变量、数组、字符串、注释、加减乘除、逻辑运算等概念；
- > 函数，包括用户自定义的函数和内置函数（例如 printf、export、eval 等）。

Shell的分类：

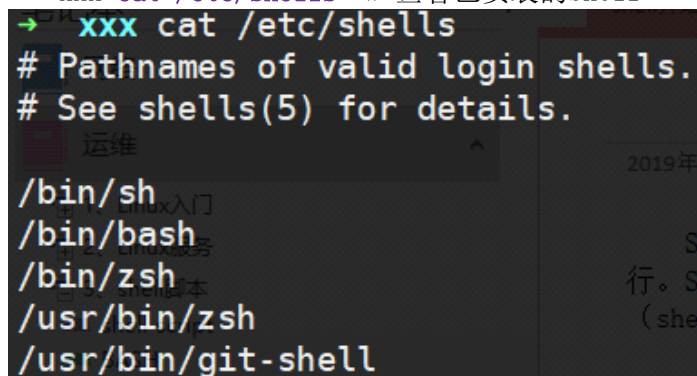
- Bourne Shell：从1979年开始Unix就使用Bourne Shell，Bourne Shell的主文件名为sh。它的功能较为简单。Bourne Shell家族包括sh, ksh, bash, psh, zsh。
- C Shell：C Shell主要在BSD (Berkeley Software Distribution, 伯克利软件套件)版的Unix系统中使用，它的语法与C语言类似。C Shell 家族主要包括 csh, tcsh。

1、Shell

→ xxx echo \$SHELL # 查看当前Shell
/usr/bin/zsh

→ xxx echo \$0 # echo \$0查看当前shell环境，\$0定义了当前的shell环境
-zsh

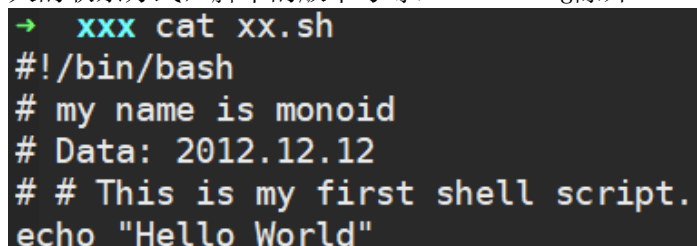
→ xxx cat /etc/shells # 查看已安装的Shell



```
→ xxx cat /etc/shells
# Pathnames of valid login shells.
# See shells(5) for details.
/bin/sh
/bin/bash
/bin/zsh
/usr/bin/zsh
/usr/bin/git-shell
```

2、脚本执行方式

shell script编写说明：最好用后缀sh定义。脚本的第一行必须是shebang开头，指明脚本的环境。
#!/bash/sh 在脚本中凡是带有#的都是注释信息，没有实质意义（对脚本的说明，比如日期，脚本编写人的联系方式，脚本的版本号等）。shebang除外



```
→ xxx cat xx.sh
#!/bin/bash
# my name is monoid
# Data: 2012.12.12
# # This is my first shell script.
echo "Hello World"
```

脚本的三种执行方式：

- a. source xx.sh
- b. bash xx.sh
- c. ./xx.sh (chmod u+x xx.sh)

循环:

1、for循环for 的写法有很多，常见的有：（以打印1-10内的数为例）

```
for i in `seq 1 10`
do
    echo $i
done

#!/bin/bash
for i in {1..10}
do
    echo $i
done

for i in 1 2 3 4 5 6 7 8 9 10
do
    echo $i
done

for((i=1;i<=10;i++))
do
    echo $i
done

for i in a b c d e f g
do
    for s in 1 2 3 4 5 6 7
    do
        echo $s + $i
        sleep 1
    done
done
```

2、while循环

```
count=0
while [ ${count} -lt 10 ]
do
    let count++
    sleep 1
    echo ${count}
done
```

Shell变量

1、局部变量:

变量名=值（等号两边不能有空格）

例：xx=24 xx是一个本地变量

查看变量的命令

echo \$xx(绝大多数情况下使用echo查看变量)，变量名前加\$

unset xx 这个是取消变量

```

→ xxx echo $my_var
→ xxx my_var=ghostwu
→ xxx echo $my_var
ghostwu

```

set （查看本地变量）

env 查看所有全局变量

```
→ xxx env
```

注：每一次登录系统，用户都是打开一个shell的进程（echo\$\$表示查看当前进程）

2、全局变量：

```

→ xxx echo $USER          # 当前用户
→ xxx echo $HISTSIZE      # 保存shell内存的历史命令条目数
→ xxx history
→ xxx !2842
→ xxx echo $PATH          # 决定了shell将到哪些目录中寻找命令或程序
→ xxx echo $HOME          # 当前用户主目录
→ xxx echo $LOGNAME       # 当前用户的登录名
→ xxx echo $SHELL         # 当前用户Shell类型
→ xxx echo $MAIL          # 当前用户的邮件存放目录

```

shell初始化：用户登录shell后，首先读取/etc/profile。该文件声明

PATH、USER、LOGNAME、MAIL、HOSTNAME、HISTSIZE、INPUTRC等shell变量。然后读取用户独有的配置文件。

/etc/bashrc定义用于shell函数和别名的系统全局变量

在/etc/profile文件中添加变量【对所有用户生效(永久的)】

用vim在文件/etc/profile文件中增加变量，该变量将会对Linux下所有用户有效，并且是“永久的”。

例如：编辑/etc/profile文件，添加xx变量

```
→ xxx sudo vim /etc/profile
```

```
export xx=aa
```

```
→ xxx source /etc/profile
```

注：修改文件后要想马上生效还要运行

~/.bash_profile：用户个人的环境设置

~/.bashrc：用户个人的别名

~/.bash_logout：用户退出执行的操作

[root@test ~]# cat .bash_logout # profile、bashrc等等文件其实质还是脚本

```
[root@test ~]# cat .bash_logout
```

```
# ~/.bash_logout
```

reboot

shell中运算符：

	含义	数字	字符
1	=	-eq	==
2	>	-gt	>

3	\geq	-ge	>=
4	<	-lt	<
5	\leq	-le	<=

Demo:

→ ~ aa=10

→ ~ bb=20

可以用test判断，还可以用[]判断，在脚本中我们会看到大量的[]的判断语句
用[]判断时首尾留有空格

→ ~ test \$aa -eq \$bb

→ ~ echo \$?

例:

→ ~ [\$aa -gt \$bb] ; echo \$?

→ ~ [\$aa -gt \$bb] && echo OK 与：前面成功才执行后面的命令

→ ~ [\$aa -gt \$bb] || echo OK 或：前面的命令成功即终止，前面的命令失败则执行后面的命令

[]除了判断数字大小还可以判断文件属性

→ ~ [-f /etc/passwd] ;echo \$?

→ ~ [-f /dev/sda1] ;echo \$?

→ ~ [-b /dev/sda1] ;echo \$?

→ ~ [-r /etc/shadow] ;echo \$?

3、until循环

until 判断（不满足条件执行）

```
1#!/bin/bash
2count=0
3until [ $count -eq 5 ]
4do
5    let count++
6    echo $count
7done
```

4、条件控制

```
yum install tmux -y
if [ $? -eq 0 ]; then
    echo "xxxxxx"
    echo "xxxxxx"
else
    echo "xxxxxx"
fi
```

5、case

```
read -p "请选择序号或者名称" number
case $number in
    1)
        echo "this is superman";;
    2)
```

```

        echo "this is betterman";;
    *)
        echo "this is noman"
esac

```

6、位置变量

```

#!/bin/bash
grep $1 /etc/passwd
if [ $? -eq 0 ]; then
    echo "这个用户$1是存在的"
else
    echo "对不起，您输入的用户不存在"
fi

```

7、定义函数：

```

function hf() {
    echo `date`
    echo `cal`
}
echo "Today is `hf`"

```

8、Shell中的数值计算

原生bash不支持简单的数学运算，但是可以通过其他命令来实现，例如 `awk` 和 `expr`，`expr` 最常用。常见的算数运算符有 `+` `-` `*` `/` `%` `==` `!=`。表达式和运算符之间要有空格。表达式要被反引号`包裹。*之前要加上\才能用作乘法运算。

```

#!/bin/bash
# 算数运算符
a=10
b=5
echo "a + b = `expr $a+$b`"
echo "a + b = `expr $a + $b`"
echo "a - b = `expr $a - $b`"
echo "a * b = `expr $a * $b`"
echo "a * b = `expr $a \* $b`"
echo "a / b = `expr $a / $b`"
echo "a == b is `expr $a == $b`"
echo "a != b is `expr $a != $b`"

```

9、单引号与双引号：

字符串是shell编程中最常见有用的数据类型，字符串可以用双引号，可以用单引号，也可以不用引号

```

#!/bin/bash
url="http://c.biancheng.net"
website1='C语言中文网: ${url}'
website2="C语言中文网: ${url}"
echo $website1
echo $website2
运行结果：
C语言中文网: ${url}

```

以单引号' '包围变量的值时,单引号里面是什么就输出什么,即使内容中有变量也会原样输出。**这种方式比较适合定义显示纯字符串的情况,即不希望解析变量、命令等的场景。**

以双引号" "包围变量的值时,输出时会先解析里面的变量和命令,而不是把双引号中的变量名和命令原样输出。这种方式比较适合字符串中附带有变量和命令并且想将其解析后再输出的变量定义。

10、综合实验:

```
#!/bin/bash
```

```
# 猜数字
```

```
echo "我会随机生成一个1-100之内的数,你来猜,我会告诉你你猜的数是大了还是小了。"
```

```
answer=`expr $RANDOM % 100 + 1`
```

```
read guess
```

```
while [ $answer -ne $guess ]
```

```
do
```

```
    if [ $guess -gt $answer ]
```

```
    then
```

```
        echo "大了,重猜。"
```

```
    elif [ $guess -lt $answer ]
```

```
    then
```

```
        echo "小了,重猜。"
```

```
    else
```

```
        break
```

```
    fi
```

```
    read guess
```

```
done
```

```
echo "恭喜你,猜对了。"
```

循环控制:continue和break。continue表示退出本次循环,进入下一次循环。break表示循环结束,不在进行下一次循环。

shell使用read读取用户的输入,格式为:read 变量名。用户输入的内容被记录在指定的变量中。

11、执行remote脚本

➔ xxx **bash** <(curl -sSL <http://127.0.0.1/xx.sh>)

<http://c.biancheng.net/shell/>

<http://c.biancheng.net/view/706.html>

<https://www.cnblogs.com/ghostwu/p/9074465.html> # 全局变量