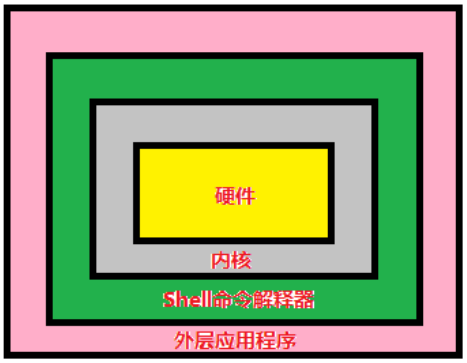
第 9 章 Shell 编程

9.1 概述

Shell 是一个命令行解释器，它为用户提供了一个向 Linux 内核发送请求以便运行程序的界面系统级程序，用户可以用 Shell 来启动、挂起、停止甚至是编写一些程序。



Shell 还是一个功能相当强大的编程语言，易编写、易调试、灵活性强。Shell 是解释执行的脚本语言，在 Shell 中可以调用 Linux 系统命令。

9.2 Shell 脚本的执行方式

2）脚本格式

（1）脚本以#!/bin/bash 开头

（2）脚本必须有可执行权限

3）第一个 Shell 脚本

（1）需求：创建一个 Shell 脚本，输出 helloworld

（2）实操：

```
[atguigu@hadoop101 datas]$ touch helloworld.sh
[atguigu@hadoop101 datas]$ vi helloworld.sh

在 helloworld.sh 中输入如下内容
    #!/bin/bash
    echo "helloworld"
```

4）脚本的常用执行方式

第一种：输入脚本的绝对路径或相对路径

（1）首先要赋予helloworld.sh 脚本的+x权限

```
[atguigu@hadoop101 datas]$ chmod 777 helloworld.sh
```

（2）执行脚本

/root/helloWorld.sh

./helloWorld.sh

第二种：bash或sh+脚本（不用赋予脚本+x权限）

sh /root/helloWorld.sh

sh helloWorld.sh

9.3 Shell 中的变量

1）Linux Shell 中的变量分为，系统变量和用户自定义变量。

2）系统变量：$HOME、$PWD、$SHELL、$USER 等等

3）显示当前 shell 中所有变量：set

9.3.1 定义变量

1）基本语法：

（1）定义变量：变量=值

（2）撤销变量：unset 变量

（3）声明静态变量：readonly 变量，注意：不能 unset

2）变量定义规则 0

（1）变量名称可以由字母、数字和下划线组成，但是不能以数字开头。

（2）等号两侧不能有空格

（3）变量名称一般习惯为大写

3）案例实操

（1）定义变量 A

A=8

（2）撤销变量A

unset A

（3）声明静态的变量B=2，不能unset

readonly B=2

（4）可把变量提升为全局环境变量，可供其他shell程序使用

export 变量名

9.3.2 将命令的返回值赋给变量

1）A=`ls -la` 反引号，运行里面的命令，并把结果返回给变量 A

2）A=$(ls -la) 等价于反引号

9.3.3 设置环境变量

1）基本语法：

（1）export 变量名=变量值　　（功能描述：将 shell 变量输出为环境变量）

（2）source 配置文件　　（功能描述：让修改后的配置信息立即生效）

（3）echo $变量名　　（功能描述：查询环境变量的值）

2）案例实操：

（1）在/etc/profile 文件中定义 JAVA_HOME 环境变量

export JAVA_HOME=/opt/module/jdk1.8.0_144

export PATH=$PATH:$JAVA_HOME/bin

（2）查看环境变量 JAVA_HOME 的值

[atguigu@hadoop101 datas]$ echo $JAVA_HOME

/opt/module/jdk1.8.0_144

9.3.4 位置参数变量

1）基本语法

$n　（功能描述：n 为数字，$0 代表命令本身，$1-$9 代表第一到第九个参数，十以上的参数，十以上的参数需要用大括号包含，如${10}）

$*　（功能描述：这个变量代表命令行中所有的参数，$*把所有的参数看成一个整体）

$@　（功能描述：这个变量也代表命令行中所有的参数，不过$@把每个参数区分对待）

$#　（功能描述：这个变量代表命令行中所有参数的个数）

2）案例实操

（1）输出输入的的参数 1，参数 2，所有参数，参数个数

```
#!/bin/bash
echo "$0    $1     $2"
echo "$*"
echo "$@"
echo "$#"
```

（3）$*与$@的区别

```
#!/bin/bash
for i in "$*"
#$*中的所有参数看成是一个整体，所以这个 for 循环只会循环一次
        do
                echo "The parameters is: $i"
        done
x=1
for y in "$@"
#$@中的每个参数都看成是独立的，所以“$@”中有几个参数，就会循环几次
        do
                echo "The parameter$x is: $y"
                x=$(( $x +1 ))
        done
```

a）$*和$@都表示传递给函数或脚本的所有参数，不被双引号“”包含时，都以$1 $2 …$n的形式输出所有参数

b）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”的形式输出所有参数；“$@”会将各个参数分开，以“$1” “$2”…”$n”的形式输出所有参数

9.3.5 预定义变量

1）基本语法：

$?　　（功能描述：最后一次执行的命令的返回状态。如果这个变量的值为0，证明上一个命令正确执行；如果这个变量的值为非0（具体是哪个数，由命令自己来决定），则证明上一个命令执行不正确了。）

$$　　（功能描述：当前进程的进程号（PID））

$!　　（功能描述：后台运行的最后一个进程的进程号（PID））

2）案例实操

```
#!/bin/bash

echo "$$"

./helloworld.sh &

echo "$!"

echo "$?"
```

9.4 运算符

1）基本语法：

（1）“$((运算式))”或“$[运算式]”

（2）expr m + n

注意 expr 运算符间要有空格

（3）expr m - n

（4）expr *, /, % 乘，除，取余

2）案例实操：计算（2+3）X4 的值

（1）采用$[运算式]方式

[root@hadoop101 datas]# S=$[(2+3)*4]

[root@hadoop101 datas]# echo $S

（2）expr 分布计算

S=`expr 2 + 3`

expr $S * 4

（3）expr 一步完成计算

expr `expr 2 + 3` * 4

9.5 条件判断

9.5.1 判断语句

1）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）

2）案例实操：

[atguigu] 返回 true

[] 返回 false

[condition] && echo OK || echo notok 条件满足，执行后面的语句

9.5.2 常用判断条件

1）两个整数之间比较

= 字符串比较

-lt 小于

-le 小于等于

-eq 等于

-gt 大于

-ge 大于等于

-ne 不等于

2）按照文件权限进行判断

-r 有读的权限

-w 有写的权限

-x 有执行的权限

3）按照文件类型进行判断

-f 文件存在并且是一个常规的文件

-e 文件存在

-d 文件存在并是一个目录

4）案例实操

（1）23 是否大于等于 22

[23 -ge 22]

（2）student.txt 是否具有写权限

[-w student.txt]

（3）/root/install.log 目录中的文件是否存在

[-e /root/install.log]

9.6 流程控制

9.6.1 if 判断

1）基本语法：

```
if [ 条件判断式 ];then
  程序
fi
```

或者

```
if [ 条件判断式 ]
  then
    程序
fi
```

注意事项：（1）[条件判断式]，中括号和条件判断式之间必须有空格

2）案例实操

```
#!/bin/bash

if [ $1 -eq "123" ]
then
        echo "123"
elif [ $1 -eq "456" ]
then
        echo "456"
fi
```

9.6.2 case 语句

1）基本语法：

```
case $变量名 in
  "值 1"）
    如果变量的值等于值 1，则执行程序 1
    ;;
  "值 2"）
    如果变量的值等于值 2，则执行程序 2
    ;;
  …省略其他分支…
  *）
    如果变量的值都不是以上的值，则执行此程序
    ;;
esac
```

2）案例实操

```
!/bin/bash

case $1 in
"1")
        echo "1"
;;

"2")
        echo "2"
```

```
;;
*)
        echo "other"
;;
esac
```

9.6.3 for 循环

1）基本语法 1：

for 变量 in 值 1 值 2 值 3…

do

程序

done

2）案例实操：

（1）打印输入参数

```
#!/bin/bash
#打印数字

for i in "$*"
    do
      echo "The num is $i "
    done

for j in "$@"
        do
        echo "The num is $j"
        done
```

3）基本语法 2：

for ((初始值;循环控制条件;变量变化))

do

程序

done

4）案例实操

（1）从 1 加到 100

```
#!/bin/bash

s=0
for((i=0;i<=100;i++))
```

```
do
        s=$[$s+$i]
done
echo "$s"
```

9.6.4 while 循环

1）基本语法：

while [条件判断式]

do

程序

done

2）案例实操

（1）从 1 加到 100

```
#!/bin/bash
s=0
i=1
while [ $i -le 100 ]
do
        s=$[$s+$i]
        i=$[$i+1]
done

echo $s
```

9.7 read 读取控制台输入

1）基本语法：

read(选项)(参数)

选项：

-p：指定读取值时的提示符；

-t：指定读取值时等待的时间（秒）。

参数

变量：指定读取值的变量名

2）案例实操

读取控制台输入的名称

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

```
read -t 7 -p "please 7 miao input your name " NAME

echo $NAME
```

9.8 函数

9.8.1 系统函数

1）basename 基本语法

basename [pathname] [suffix]

basename [string] [suffix] （功能描述：basename 命令会删掉所有的前缀包括最后一个（‘/’）字符，然后将字符串显示出来。

选项：

suffix 为后缀，如果 suffix 被指定了，basename 会将 pathname 或 string 中的 suffix 去掉。

2）案例实操

[atguigu@hadoop101 opt]$ basename /opt/test.txt

test.txt

[atguigu@hadoop101 opt]$ basename /opt/test.txt .txt

test

3）dirname 基本语法

dirname 文件绝对路径 （功能描述：从给定的包含绝对路径的文件名中去除文件名（非目录的部分），然后返回剩下的路径（目录的部分））

4）案例实操

[atguigu@hadoop101 opt]$ dirname /opt/test.txt

/opt

9.8.2 自定义函数

1）基本语法：

```
[ function ] funname[()]
{
    Action;
    [return int;]
}

funname
```

2）经验技巧：

（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)

3）案例实操

（1）计算输入参数的和

```
#!/bin/bash
function sum()
{
    s=0
    s=$[ $1 + $2 ]
    echo "$s"
}

read -p "Please input the number1: " n1;
read -p "Please input the number2: " n2;
sum $n1 $n2;
```