.Shell 编程入门

　。走进 Shell 编程的大门

　　　为什么要学Shell？

　　　什么是 Shell？

　　　Shell 编程的 Hello World

　。Shell 变量

　　　Shell 编程中的变量介绍

　　　Shell 字符串入门

　　　Shell 字符串常见操作

　　　Shell 数组

　。Shell 基本运算符

　　　算数运算符

关系运算符

逻辑运算符

布尔运算符

字符串运算符

文件相关运算符

　。shell流程控制

if 条件语句

for 循环语句

while 语句

　。shell 函数

　不带参数没有返回值的函数

　有返回值的函数

　带参数的函数

Shell 编程入门

走进 Shell 编程的大门

为什么要学Shell？

学一个东西，我们大部分情况都是往实用性方向着想。从工作角度来讲，学习 Shell 是为了提高我们自己工作效率，提高产出，让我们在更少的时间完成更多的事情。

很多人会说 Shell 编程属于运维方面的知识了，应该是运维人员来做，我们做后端开发的没必要学。我觉得这种说法大错特错，相比于专门做Linux运维的人员来说，我们对 Shell 编程掌握程度的要求要比他们低，但是shell编程也是我们必须要掌握的！

目前Linux系统下最流行的运维自动化语言就是Shell和Python了。

两者之间，Shell几乎是IT企业必须使用的运维自动化编程语言，特别是在运维工作中的服务监控、业务快速部署、服务启动停止、数据备份及处理、日制分析等环节里，shell是不可缺的。Python 更适合处理复杂的业务逻辑，以及开发复杂的运维软件工具，实现通过web访问等。Shell是一个命令解释器，解释执行用户所输入的命令和程序。一输入命令，就立即回应的交互的对话方式。

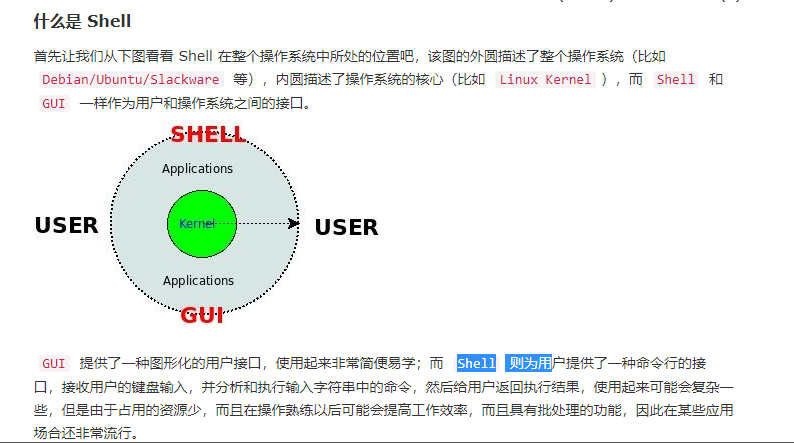
另外，了解 shell 编程也是大部分互联网公司招聘后端开发人员的要求。下图是我截取的一些知名互联网公司对于 Shell 编程的要求。



什么是 Shell？

简单来说“Shell编程就是对一堆Linux命令的逻辑化处理”。

W3Cschool 上的一篇文章是这样介绍 Shell的，如下图所示。



Shell 编程的 Hello World

学习任何一门编程语言第一件事就是输出HelloWord了！下面我会从新建文件到shell代码编写来说下Shell 编程如何输出Hello World。

(1)新建一个文件 helloworld.sh :touch helloworld.sh，扩展名为 sh（sh代表Shell）（扩展名并不影响脚本执行，见名知意就好，如果你用 php 写 shell 脚本，扩展名就用 php 好了）

(2) 使脚本具有执行权限：chmod +x helloworld.sh

(3) 使用 vim 命令修改helloworld.sh文件：vim helloworld.sh(vim 文件------>进入文件----->命令模式------>按i进入编辑模式----->编辑文件 ------->按Esc进入底行模式----->输入:wq/q! （输入wq代表写入内容并退出，即保存；输入q!代表强制退出不保存。）)

helloworld.sh 内容如下：

#!/bin/bash

#第一个shell小程序,echo 是linux中的输出命令。

echo "helloworld!"

shell中 # 符号表示注释。shell 的第一行比较特殊，一般都会以#!开始来指定使用的 shell 类型。在linux中，除了bash shell以外，还有很多版本的shell， 例如zsh、dash等等...不过bash shell还是我们使用最多的。

(4) 运行脚本:./helloworld.sh 。（注意，一定要写成 ./helloworld.sh ，而不是 helloworld.sh ，运行其它二进制的程序也一样，直接写 helloworld.sh ，linux 系统会去 PATH 里寻找有没有叫 test.sh 的，而只有 /bin, /sbin, /usr/bin，/usr/sbin 等在 PATH 里，你的当前目录通常不在 PATH 里，所以写成 helloworld.sh 是会找不到命令的，要用./helloworld.sh 告诉系统说，就在当前目录找。）

shell ç¼ç¨Hello World

Shell 变量

Shell 编程中的变量介绍

Shell编程中一般分为三种变量：

1.我们自己定义的变量（自定义变量）: 仅在当前 Shell 实例中有效，其他 Shell 启动的程序不能访问局部变量。

2.Linux已定义的环境变量（环境变量， 例如：$PATH, $HOME 等..., 这类变量我们可以直接使用），使用 env 命令可以查看所有的环境变量，而set命令既可以查看环境变量也可以查看自定义变量。

3.Shell变量 ：Shell变量是由 Shell 程序设置的特殊变量。Shell 变量中有一部分是环境变量，有一部分是局部变量，这些变量保证了 Shell 的正常运行

常用的环境变量:

PATH 决定了shell将到哪些目录中寻找命令或程序 HOME 当前用户主目录 HISTSIZE　历史记录数 LOGNAME 当前用户的登录名 HOSTNAME　指主机的名称 SHELL 当前用户Shell类型 LANGUGE 　语言相关的环境变量，多语言可以修改此环境变量 MAIL　当前用户的邮件存放目录 PS1　基本提示符，对于root用户是#，对于普通用户是$

使用 Linux 已定义的环境变量：

比如我们要看当前用户目录可以使用：echo $HOME命令；如果我们要看当前用户Shell类型 可以使用echo $SHELL命令。可以看出，使用方法非常简单。

使用自己定义的变量：

#!/bin/bash

#自定义变量hello

hello="hello world"

echo $hello

echo "helloworld!"

ä½¿ç¨èªå·±å®ä¹çåé

Shell 编程中的变量名的命名的注意事项：

.命名只能使用英文字母，数字和下划线，首个字符不能以数字开头，但是可以使用下划线（\_）开头。

.中间不能有空格，可以使用下划线（\_）。

.不能使用标点符号。

.不能使用bash里的关键字（可用help命令查看保留关键字）。

Shell 字符串入门

字符串是shell编程中最常用最有用的数据类型（除了数字和字符串，也没啥其它类型好用了），字符串可以用单引号，也可以用双引号。这点和Java中有所不同。

单引号字符串：

#!/bin/bash

name='SnailClimb'

hello='Hello, I am '$name'!'

echo $hello

输出内容：

Hello, I am SnailClimb!

双引号字符串：

#!/bin/bash

name='SnailClimb'

hello="Hello, I am "$name"!"

echo $hello

输出内容：

Hello, I am SnailClimb!

Shell 字符串常见操作

拼接字符串：

#!/bin/bash

name="SnailClimb"

# 使用双引号拼接

greeting="hello, "$name" !"

greeting\_1="hello, ${name} !"

echo $greeting $greeting\_1

# 使用单引号拼接

greeting\_2='hello, '$name' !'

greeting\_3='hello, ${name} !'

echo $greeting\_2 $greeting\_3

输出结果：

è¾åºç»æ

获取字符串长度：

#!/bin/bash

#获取字符串长度

name="SnailClimb"

# 第一种方式

echo ${#name} #输出 10

# 第二种方式

expr length "$name";

输出结果:

10

10

使用 expr 命令时，表达式中的运算符左右必须包含空格，如果不包含空格，将会输出表达式本身:

expr 5+6 // 直接输出 5+6

expr 5 + 6 // 输出 11

对于某些运算符，还需要我们使用符号""进行转义，否则就会提示语法错误。

expr 5 \* 6 // 输出错误

expr 5 \\* 6 // 输出30

截取子字符串:

简单的字符串截取：

#从字符串第 1 个字符开始往后截取 10 个字符

str="SnailClimb is a great man"

echo ${str:0:10} #输出:SnailClimb

根据表达式截取：

#!bin/bash

#author:amau

var="http://www.runoob.com/linux/linux-shell-variable.html"

s1=${var%%t\*}#h

s2=${var%t\*}#http://www.runoob.com/linux/linux-shell-variable.h

s3=${var%%.\*}#http://www

s4=${var#\*/}#/www.runoob.com/linux/linux-shell-variable.html

s5=${var##\*/}#linux-shell-variable.html

Shell 数组

bash支持一维数组（不支持多维数组），并且没有限定数组的大小。我下面给了大家一个关于数组操作的 Shell 代码示例，通过该示例大家可以知道如何创建数组、获取数组长度、获取/删除特定位置的数组元素、删除整个数组以及遍历数组。

#!/bin/bash

array=(1 2 3 4 5);

# 获取数组长度

length=${#array[@]}

# 或者

length2=${#array[\*]}

#输出数组长度

echo $length #输出：5

echo $length2 #输出：5

# 输出数组第三个元素

echo ${array[2]} #输出：3

unset array[1]# 删除下表为1的元素也就是删除第二个元素

for i in ${array[@]};do echo $i ;done # 遍历数组，输出： 1 3 4 5

unset arr\_number; # 删除数组中的所有元素

for i in ${array[@]};do echo $i ;done # 遍历数组，数组元素为空，没有任何输出内容

Shell 基本运算符

说明：图片来自《菜鸟教程》

Shell 编程支持下面几种运算符

算数运算符

关系运算符

布尔运算符

字符串运算符

文件测试运算符

算数运算符

算数运算符

我以加法运算符做一个简单的示例：

#!/bin/bash

a=3;b=3;

val=`expr $a + $b`

#输出：Total value : 6

echo "Total value : $val

关系运算符

关系运算符只支持数字，不支持字符串，除非字符串的值是数字。

通过一个简单的示例演示关系运算符的使用，下面shell程序的作用是当score=100的时候输出A否则输出B。

#!/bin/bash

score=90;

maxscore=100;

if [ $score -eq $maxscore ]

then

echo "A"

else

echo "B"

fi

输出结果：

B

逻辑运算符

示例：

#!/bin/bash

a=$(( 1 && 0))

# 输出：0；逻辑与运算只有相与的两边都是1，与的结果才是1；否则与的结果是0

echo $a;

布尔运算符

这里就不做演示了，应该挺简单的。

字符串运算符

简单示例：

#!/bin/bash

a="abc";

b="efg";

if [ $a = $b ]

then

echo "a 等于 b"

else

echo "a 不等于 b"

fi

输出：

a 不等于 b

文件相关运算符

使用方式很简单，比如我们定义好了一个文件路径file="/usr/learnshell/test.sh" 如果我们想判断这个文件是否可读，可以这样if [ -r $file ] 如果想判断这个文件是否可写，可以这样-w $file，是不是很简单。

shell流程控制

if 条件语句

简单的 if else-if else 的条件语句示例

#!/bin/bash

a=3;

b=9;

if [ $a = $b ]

then

echo "a 等于 b"

elif [ $a > $b ]

then

echo "a 大于 b"

else

echo "a 小于 b"

fi

输出结果：

a 大于 b

相信大家通过上面的示例就已经掌握了 shell 编程中的 if 条件语句。不过，还要提到的一点是，不同于我们常见的 Java 以及 PHP 中的 if 条件语句，shell if 条件语句中不能包含空语句也就是什么都不做的语句。

for 循环语句

通过下面三个简单的示例认识 for 循环语句最基本的使用，实际上 for 循环语句的功能比下面你看到的示例展现的要大得多。

输出当前列表中的数据：

for loop in 1 2 3 4 5

do

echo "The value is: $loop"

done

产生 10 个随机数：

#!/bin/bash

for i in {0..9};

do

echo $RANDOM;

done

输出1到5:

通常情况下 shell 变量调用需要加 $,但是 for 的 (()) 中不需要,下面来看一个例子：

#!/bin/bash

for((i=1;i<=5;i++));do

echo $i;

done;

while 语句

基本的 while 循环语句：

#!/bin/bash

int=1

while(( $int<=5 ))

do

echo $int

let "int++"

done

while循环可用于读取键盘信息：

echo '按下 <CTRL-D> 退出'

echo -n '输入你最喜欢的电影: '

while read FILM

do

echo "是的！$FILM 是一个好电影"

done

输出内容:

按下 <CTRL-D> 退出

输入你最喜欢的电影: 变形金刚

是的！变形金刚 是一个好电影

无限循环：

while true

do

command

done

shell 函数

不带参数没有返回值的函数

#!/bin/bash

function(){

echo "这是我的第一个 shell 函数!"

}

Function

输出结果：

这是我的第一个 shell 函数!

有返回值的函数

输入两个数字之后相加并返回结果：

#!/bin/bash

funWithReturn(){

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 $?"

输出结果：

输入第一个数字:

1

输入第二个数字:

2

两个数字分别为 1 和 2 !

输入的两个数字之和为 3

带参数的函数

#!/bin/bash

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

输出结果：

第一个参数为 1 !

第二个参数为 2 !

第十个参数为 10 !

第十个参数为 34 !

第十一个参数为 73 !

参数总数有 11 个!

作为一个字符串输出所有参数 1 2 3 4 5 6 7 8 9 34 73 !