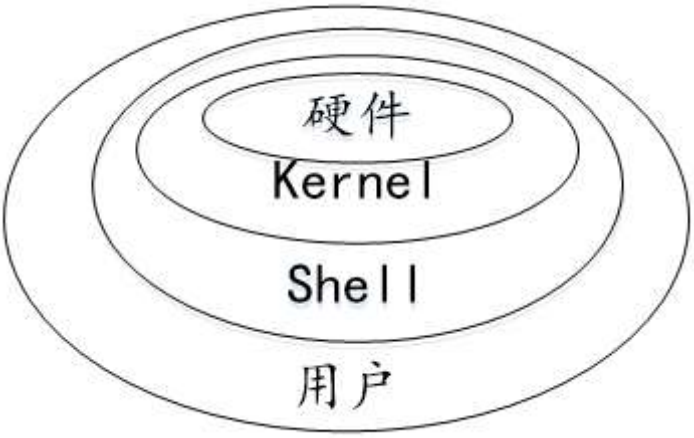
C4 Shell 编程

Shell概述

Shell既是一种命令语言，又是一种程序设计语言（即Shell脚本）。作为一种基于命令的语言，Shell交互式地解释和执行用户输入的命令；作为程序设计语言，Shell中可以定义变量、传递参数，并提供了许多高级语言所有的流程控制结构。



shell名称	说明
BSh	Bash Shell是Bourne Shell的一个免费版本，是最早的Unix Shell，包括许多附加的特点，是一个交换式的命令解释器和命令编程语言。
CSh	C Shell中使用"类C"语法，借鉴了Bourne Shell的许多特点，新增了命令历史、别名、文件名替换等功能。
KSh	Korn Shell的语法与Bourne Shell相同，同时具备了C Shell的交互特性，因此广受用户青睐。
bash	Bourne Again Shell，即bash，是GNU计划的一部分，用于GNU/Linux系统，大多数Linux都以bash作为缺省的shell。

Linux系统中的可执行文件可以分为五类：

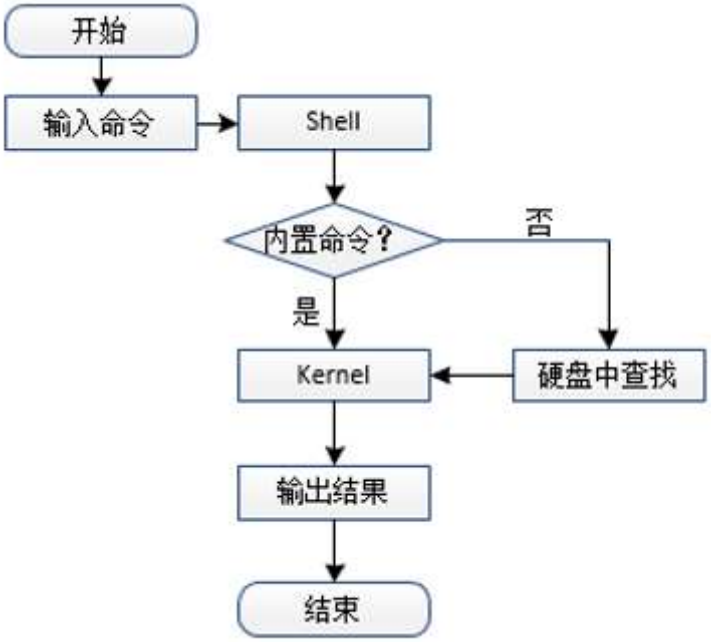
Linux命令：用来使系统执行某种操作的指令，存放在/bin和/sbin目录下；

内置命令：存放于Shell内部的命令的解释程序，是一些常用的命令。可以使用“type 命令名”的方式来查看某个命令是否为内置命令；

实用程序：存放于/usr/bin、/usr/sbin、/usr/local/bin等目录下的程序，如ls、which等；

用户程序：由用户编写的，经过编译后可执行的文件；

Shell脚本：使用Shell语言编写的批处理文件。



Shell应用技巧

输入/输出重定向

Linux系统中将从终端输入数据称为标准输入，将打印数据到终端称为标准输出，并设置了3个标准文件，分别关联标准输入、标准输出以及标准错误输出信息，标准输入文件的编号为0，默认设备是键盘；标准输出文件的编号为1，默认设备是显示器；标准错误文件的编号为2，默认设备也是显示器。

实现输入重定向的运算符为“<”，具体格式如下：

命令<文件名　命令 0<文件名

实现输出重定向的运算符为“>”，具体格式如下：

命令>文件名　命令 1>文件名

重定向标准错误信息使用运算符“>”和“>>”，其格式如下：

命令 2>文件名

可以使用运算符“&”通过文件编号引用文件，该运算符表示“等同于”，如“2>&1”则表示将标准错误重定向到标准输出中。

管道

在shell编程中，“|”被称为管道符号，用于连接两个命令，其格式如下：

命令1 | 命令2 | … | 命令n

管道可使前一个命令的输出作为后一个命令的输入，由此实现较为复杂的功能。

示例：ls –l /etc | grep init

命令连接符

Shell中提供了一些用于连接符号的符号，包括：“;”、“&&”以及“||”，使用这些符号，可以将多条shell指令进行连接，使这些指令顺序或根据命令执行结果，有选择地执行，下面将对这些符号的功能分别进行介绍。

①使用“;”运算符间隔的命令，会按照先后次序依次执行。

②使用“&&”连接符连接的命令，其前后命令的执行遵循逻辑与关系，只有该连接符之前的命令执行成功后，它后面的命令才被执行。

③使用“||”连接符链接的命令，其前后命令的执行遵循逻辑或关系，只有该连接符之前的命令执行失败时，才会执行后面的命令。

文本提取器

Shell中常用awk提取文档或标准输出中的文本信息，awk实际上是一个强大的文本分析工具，该工具是Shell中常用的文本提取命令为awk，该命令类似于常用命令中的grep，都能从指定文本中提取指定信息，不同的是，该命令可将文本按指定分隔符分割，并从分割后的文本中提取指定项。

awk的用法如下：

awk [-F分隔符1] ‘{print $1 [“分隔符2”] $2}’

说明：

①-F选项用于指定分隔符，若缺省则以空格分割；

②分隔符2用于设置打印提取项时使用的分隔符

示例：awk –F: ‘{print $1 “ ”$3}’ /etc/passwd

cat /etc/passwd | awk -F: ‘{print $1 “ ”$3}’

Shell编程

第一个Shell程序

Shell脚本的执行方法有两种（脚本名记为first）：

①为脚本文件添加可执行权限之后直接执行

示例：chmod +x first

./first

②将脚本文件名作为参数，通过shell对文件进行解析

示例：sh first

变量的定义

Shell变量定义规则：

Shell中的变量在使用之前无需定义，可以在使用的同时创建。

Shell中的变量没有明确的分类，一般情况下，一个变量保存一个串。

Shell中的变量若要进行计算，需使用工具程序进行转换。

Shell变量的变量名由字母、数字和下划线组成，开头只能是字母或下划线。

若shell变量中出现其它字符，则表示变量名到此前为止。

给变量赋值时，等号两端不能有空格，其格式为“变量名=值”，若要给变量赋空值，缺省格式中“值”的部分，跟上换行符即可。

若变量中含有空格，必须使用引号将变量括起。

正确示例：

Var=hello

_Var=hello$itheima

Var='hello itheima'

Var="hello itheima"

错误示例：

Var.h=hello

Var%k=hello$itheima

注意：

Shell中可以使用关键字“readonly”将变量设置为只读变量，如：

readonly var

此后若要重新为变量赋值，则会提示错误信息。

变量的引用

Shell中通过“$”符号来引用变量，用于区分变量与普通字符串，若要输出已定义的变量，其格式如下：

echo $var

其中echo类似于C语言中的printf()函数，用于实现变量或字符串的打印功能。

需要说明的是，在使用echo打印变量时，单引号与双引号的效果略有差异，使用单引号引起的$var会打印该字符串“$var”，而使用双引号引起的$var则打印变量var中存储的值。

在Shell脚本中，不仅可以引用整个变量，也可引用变量中的部分数据，或与变量相关的信息，具体如表所示。

变量的输入

Shell脚本中可以使用read命令从终端读取信息，该命令的功能类似于C语言中的scanf()函数，其格式如下：

read 变量名

当脚本执行到read语句时，终端会阻塞等待用户输入，并将用户输入的数据赋给read语句中的变量。

变量的分类

除普通变量外，Shell脚本中还会用到一些特别的变量，包括：环境变量、位置变量、标准变量和特殊变量。

环境变量

环境变量又称永久变量，此类变量不仅可以作用于单个脚本，还可用于创建该变量的Shell以及从该Shell派生的子Shell或进程中。

环境变量使用”export”关键字设定或创建，若要将一个本地变量更改为环境变量，可使用以下方法：

export 变量名

若要创建一个环境变量，则使用以下格式：

export 变量名=值

位置变量

位置变量即执行脚本时传入脚本中，对应位置的变量，类似函数的参数，引用方法为“$”加上参数的位置，如：$0、$1、$2，其中$0比较特殊，表示脚本的命令，其余依次表示传入脚本中第一个参数、第二个参数等。（脚本名sec.sh）

```
1 #1/bin/sh
2 echo "number of vars:"$#
3 echo "name of shell script:"$0
4 echo "fisrt var:"$1
5 echo "second var:"$2
6 echo "third var:"$3
7
```

```
number of vars:3
name of shell script:sec.sh
fisrt var:a
second var:b
third var:c
```

标准变量

标准变量也是环境变量，在bash环境建立时生成，该变量自动解析，通过查看etc目录下的profile文件可以查看系统中的标准环境变量；使用env命令可以查看系统中的环境变量，包括环境变量和标准变量。

特殊变量

Shell中定义了一些特殊的变量，这些变量及含义分别如下：

#。传递到脚本或函数的参数数量。

?。上个命令执行情况，0表示成功，其它值表示失败。

$。运行脚本的进程id。

*。传递给脚本或函数的所有参数。

变量的运算

Shell中变量没有明确的类型，变量值都以字符串的形式存储，但Shell中也可能进行一些算术运算，因此需要使用命令将变量中的字符串转换为数值。Shell中的运算一般通过两个命令实现：let和expr。

let

let命令可以进行算术运算和数值表达式测试

expr

Expr命令可以对整型变量进行算术运算，使用expr命令时可以使两个数值直接进行运算

运算的结果会直接在命令行输出。

若要通过变量的引用进行运算，添加“$”即可。

①若表达式中是遍历，使用“$”引用即可。

②运算符与变量或数据之间需要有一个空格。

②若要在脚本中使用expr命令，需要使用符号“`”（该按键一般位于Tab键之上）将其内嵌到等式当中。

Shell中的条件语句

条件判断

条件判断是条件语句的核心，Shell中通常使用test命令或[命令对条件进行判断，其判断的条件可以是命令或脚本。

字符串比较

条件	说明
str1=str2	若字符串str1等于str2，则结果为真
str1!=str2	若字符串str1不等于str2，则结果为真
–n str	若字符串str不为空，则结果为真
–z str	若字符串str为空，则结果为真

算术比较

条件	说明
expr1 –eq expr2	若表达式expr1与expr2返回值相同，则结果为真
expr1 –ne expr2	若表达式expr1与expr2返回值不同，则结果为真
expr1 –gt expr2	若表达式expr1返回值大于expr2返回值，则结果为真
expr1 –ge expr2	若表达式expr1返回值大于等于expr2返回值，则结果为真
expr1 –lt expr2	若表达式expr1返回值小于expr2返回值，则结果为真
expr1 –le expr2	若表达式expr1返回值小于等于expr2返回值，则结果为真
!expr	若表达式结果为假，则结果为真

针对文件的条件测试

条件	说明
-d file	若文件file是目录，则结果为真
-f file	若文件file是普通文件，则结果为真
-r file	若文件file可读，则结果为真
-w file	若文件file可写，则结果为真
-x file	若文件file可执行，则结果为真
-s file	若文件file大小不为0，则结果为真
-a file	若文件file存在，则结果为真

if条件语句

select条件语句

case语句

Shell中的循环语句

for循环

while循环

until循环

Shell脚本测试

Shell编程中可以通过Shell提供的一些选项，对脚本文件进行调试，其中常用的调试选项有：-n、-v、-x，这些选项的功能分别如下：

-n。不执行脚本，仅检查脚本中的语法问题；

-v。在执行脚本的过程中，将执行过的脚本命令打印到屏幕；

-x。将用到的脚本内容打印到屏幕上。

Shell中的函数

函数的调用

shell中函数的格式如下：

[function] 函数名[()]{

    代码段

    [return int]

}

①function关键字可以省略

②函数名后的()可以省略，若省略，函数名与{之间需有一个空格

③return语句可以省略，若无return语句，函数返回代码段中最后一条命令的执行结果

函数的参数

Shell编程中可通过位置变量向脚本中传递参数，Shell脚本的函数没有参数列表，但亦可通过环境变量向其中传递参数。函数中的位置变量不与脚本中的位置变量冲突，函数中的位置变量在函数调用处传入，脚本中的位置变量在脚本执行时传入。

函数的变量

当然，有些情况下我们并不期望在函数中的变量可以在全局使用，而只是希望定义一个在函数内部使用的变量，但若函数中的变量与脚本中的变量冲突，脚本执行结果显然会与预期不符。Shell脚本中可以通过“local”关键字来定义一个仅作用于函数的局部变量。