# TCL简介

Tcl("Tool Command Language"，即工具命令语言；Tcl读作“踢叩” "tickle" )是一种易学易用易于扩展的脚本语言,实际上包含了两个部分：一个语言和一个库。

首先，Tcl是一种简单的脚本语言，主要使用于发布命令给一些交互程序如文本编辑器、调试器和shell。它有一个简单的语法和很强的可扩充性，Tcl可以创建新的过程以增强其内建命令的能力。

其次，Tcl是一个库，可以被嵌入应用程序，Tcl的库包含了一个分析器、用于执行内建命令的例程和可以使你扩充（定义新的过程）的库函数。应用程序可以产生Tcl命令并执行，命令可以由用户产生，也可以从用户接口的一个输入中读取（按钮或菜单等）。

Tcl和其他编程语言例如C不同，它是一种解释语言而非编译语言。Tcl程序由一系列Tcl命令组成，在运行时由Tcl解释器解释运行。

Tcl有大量的用C/C++编写的扩展用于提供Tcl本身不具有的功能。其中使用最广泛的扩展是TK,TK提供了各种OS平台下的图形用户界面GUI(连强大的Python语言都不单独提供自己的GUI，而是提供接口适配到TK上)。另一个流行的扩展包是Exp-ect，它提供了通过终端自动执行命令的能力，例如passwd, ftp, telnet等命令驱动的外壳。

官方网站为<http://www.tcl-lang.org/>

# TCL环境设置

要设置TCL环境，需要在计算机上提供以下两个工具：(一)文本编辑器，(二)Tcl解释器。

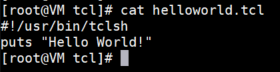
## 2.1文本编辑器

这将被用来写入程序。一些常用的编辑器包括Windows记事本，操作系统编辑命令，如VIM、vi。

编辑器创建的文件称为源文件，并包含程序的源代码。对Tcl程序的源文件来说，扩展名为“.tcl”。

## Tcl解释器

解释器会逐行执行Tcl命令，遇到错误时停止执行而不会完全编译TCL文件。看一个tcl的示例程序helloWorld.tcl

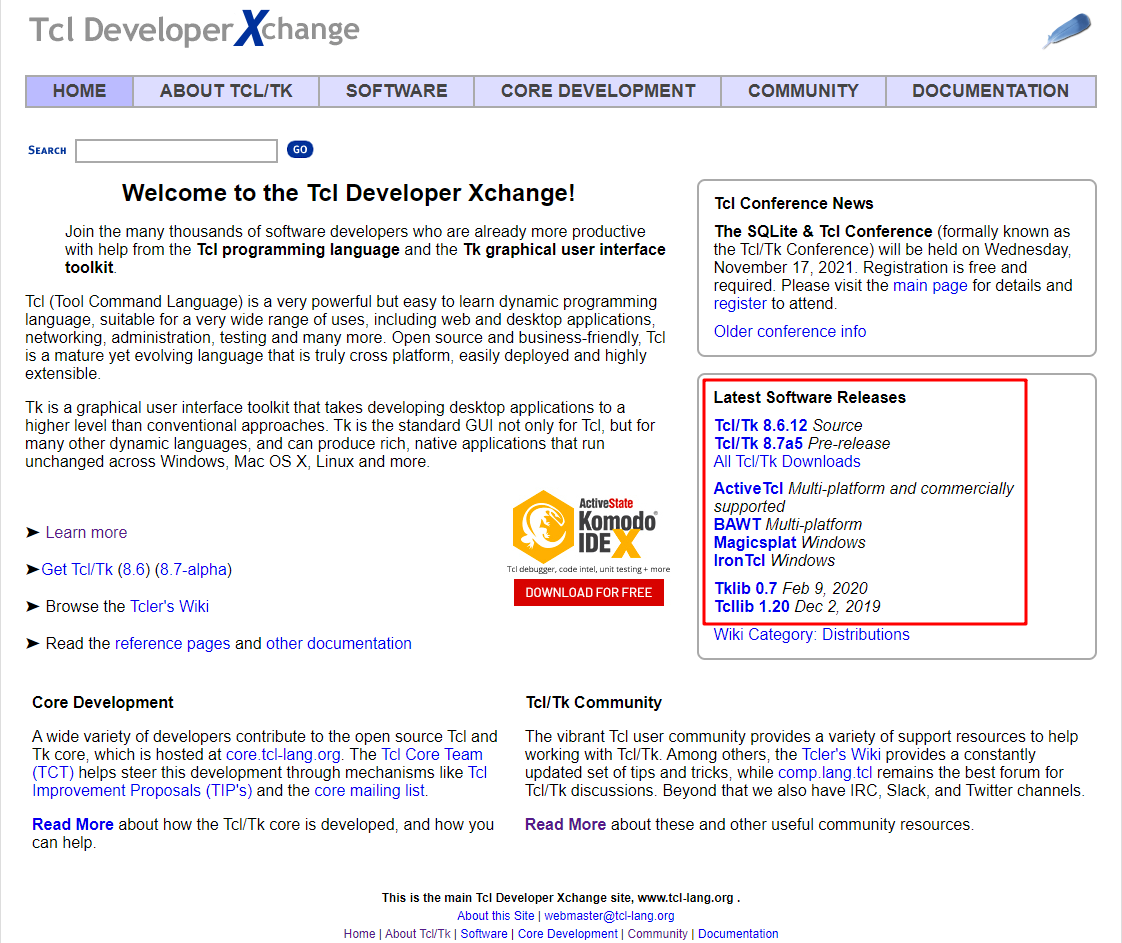


使用tcl解释器运行tcl程序：



## 2.3在windows上安装

从Tcl官网二进制文件列表下载最新版本的Windows安装程序。Tcl社区版免费供个人使用。运行下载的可执行文件安装Tcl。



运行Tcl的 helloworld.tcl 文件

C:\Tcl> tclsh helloworld.tcl

输出

C:\Tcl> helloWorld

## 2.4在Linux上安装

大多数Linux操作系统内置Tcl，可以在这些系统上直接使用。万一它不可用，可以使用下面的命令来下载和安装Tcl-TK。

$ yum install tcl tk

# 3.TCL内置特殊变量



Tcl版本：

#!/usr/bin/tclsh

puts $tcl\_version



Tcl环境路径

puts $env(PATH)



# 4.TCL基本语法

所有的TCL文件都以.tcl为扩展名。

在TCL中，使用新的行或分号终止代码前一行，但分号不是必要的

## 4.1注释

解释器会忽略注释。单行注释以#开始

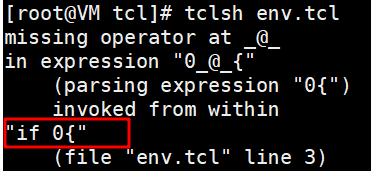
#!/usr/bin/tclsh

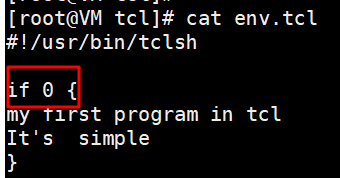
# my first program in Tcl

puts "Hello World!"

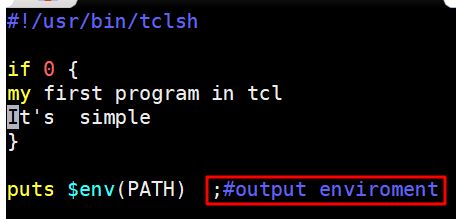
多行注释使用if 0代码块

但要注意，if 0和{之间要加空格，否则会报错





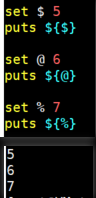
行内注释使用;＃



## 4.2标识符

Tcl标识符是用来标识变量，函数，或任何其它用户定义的项目的名称。

在TCL中一个标识符以字母A-Z或a-z或下划线开头，后跟零个或多个下划线，美元符号($)或数字（0〜9）。

注意，标识符中的$符号如果出现在中间，标识符需要加{};$在最后则不需要。甚至$、@、%符号也可以单独作为标识符(不建议这么用，没有实际意义)

#合法标识符

set aa\_@$ 20

puts ${aa\_@$}

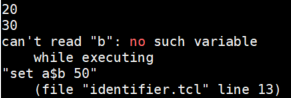
set {a$b} 30

puts ${a$b}

#非法标识符

set a$b 50 ;#此处的b被当做一个变量

puts ${a$b}



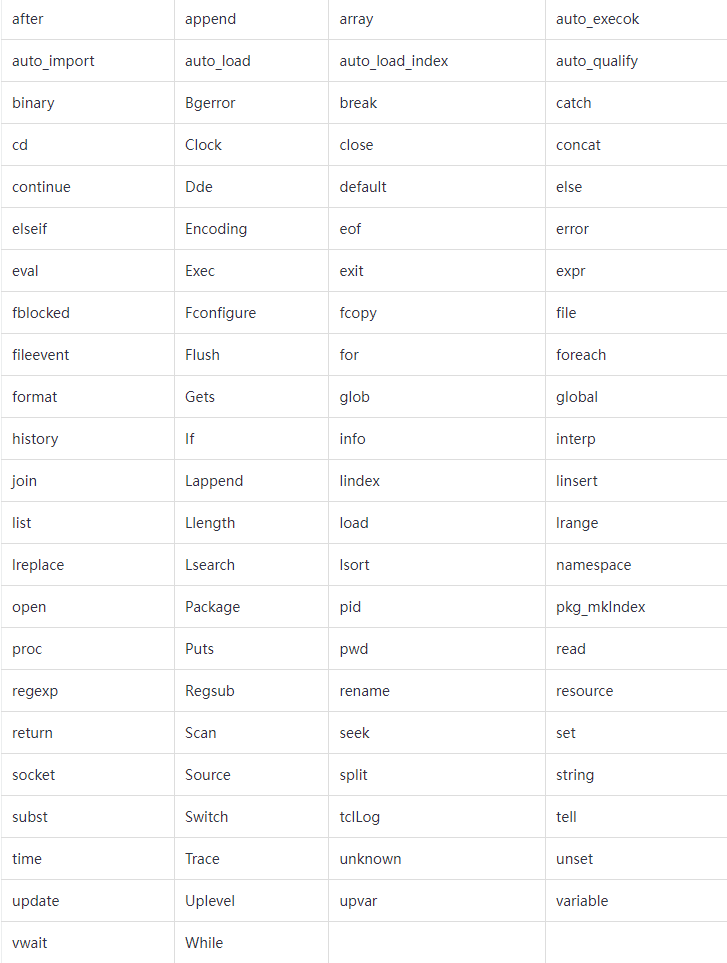
TCL是大小写敏感的语言。因此Manpower 和manpower 在Tcl是两个不同的标识符。

下面是一些合法标识符的例子：



## 4.3保留字

下表列出了TCL中的保留字，这些保留字不可被用作常量或变量，或其他任何标识符名称。



## 4.4TCL空格

一行只包含空格，即空行会被Tcl解释器忽略

Tcl代码中，关键字和后面的变量之间必须至少有一个空格，以便解释器能够区分。如：

puts "Hello World!"

有时不需要空格在字符之间，如3和+之间，或+和2也没有必要（可以不用），虽然是自由的，如果想便于阅读，建议添加一些空格。

## 4.5TCL命令

TCL是工具命令语言，命令是TCL中最重要的组成部分。这些命令构成该语言的保留字，不能用于其它变量命名。

Tcl命令的语法如下：

commandName argument1 argument2 ... argumentN

Tcl命令的简单例子：

#!/usr/bin/tclsh

puts "Hello, world!"

上面的语句中，puts是Tcl命令，"Hello, world!"是参数1

#!/usr/bin/tclsh

puts stdout "Hello, world!"

puts 是Tcl命令，stdout是参数1 ，“Hello World”参数2。在这里，在标准输出打印Hello World.

**命令替换**

方括号中的脚本会先被执行，然后方括号外的命令被执行

puts [expr 3+2+3]

**变量替换**

$使用在变量名之前，这将返回该变量的值。一个简单的例子为一个变量设置值并打印

set a 10

puts $a

执行输出10

**反斜杠替换**

通常被称为转义序列，每个反斜线后跟有自己的含义字母。一个简单的例子换行符替换如下所示。

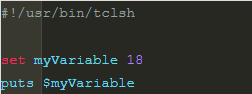
puts "Hello\nWorld"



## 4.6TCL数据类型

### 4.6.1简单的Tcl对象

在Tcl中，无论是整数，布尔值，浮点数或字符string。当你想使用一个变量时，你可以直接给它指定一个值，在Tcl中没有声明的步骤。这些不同类型对象可以有内部表示。当需要时，它可以将一种数据类型转换为另一种数据类型。将值分配给变量的语法如下：



上述语句将创建一个变量myVariable并将其存储为字符串string，即使我们还没有使用双引号。现在，如果我们尝试对变量进行算术运算，它将自动转换为一个整数。一个简单的例子如下所示：

set a 10

puts [expr $a + 5]

输出

需要注意的一点是，这些变量没有任何默认值，在使用之前必须赋值。

如果我们尝试使用puts打印，则该数字将转换为正确的字符string。

### 4.6.2字符串表示

与其他语言不同，在Tcl中单个单词不需要双引号。一个例子是

set str hello

puts $str



当要表示多个单词时，可以使用双引号或大括号。如下所示

set words "hello world"

puts $words

set words1 {hello world}

puts $words1



### 4.6.3列表

列表存储的是一组元素。可以用一组使用双引号或大括号的单词来表示一个简单的列表。

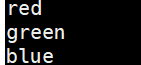
set mylist {red green blue}

puts [lindex $mylist 0]

puts [lindex $mylist 1]

puts [lindex $mylist 2]

输出：



### 4.6.4关联数组

关联数组具有不一定是整数的索引（键）。它通常是一个像键值对一样的字符串。

set marks(a) 80

puts $marks(a) ;#输出80

set marks(b) 90.5

puts $marks(b) ;#输出90.5

### 4.6.5句柄

TCL句柄通常用于表示文件和图形对象。这些可以包括句柄网络请求以及其它通道，如串口通讯，套接字或I/O设备。下面是创建一个文件句柄的例子

set myfile [open "filename" r]

## 4.7Tcl变量

**变量命名**

变量的名称可以包含任何字符和长度。甚至可以将变量包围在花括号中，但不建议这样做

set命令用于为变量赋值。set命令的语法是

**set variableName value**

set A 10

set {variable B} test ;#不建议这样写

puts $A ;#输出10

puts ${variable B} ;#输出test

**动态类型**

TCL是一种动态类型语言。变量的值可以在需要时被动态地转换为所需的类型。例如，一个数字10，其被存储为字符串，进行算术运算时被转换为数字。如下所示。

set variableA "10"

puts $variableA

set sum [expr $variableA +20] ;#字符串转为数字

puts $sum

**数学表达式**

expr是用于表示数学表达式。 Tcl默认精度为12位。为了得到浮点运算的结果，应该增加至少一个浮点数。

set variableA "10"

set result [expr $variableA / 9];

puts $result

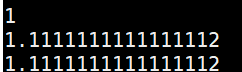
set result [expr $variableA / 9.0];

puts $result

set variableA "10.0"

set result [expr $variableA / 9];

puts $result



第一种情况，被除数和除数是整数，得到一个整数作为结果。

第二种情况，除数是小数；第三种情况，被除数是小数。在第二和第三情况下，得到的是浮点数。

可以使用tcl\_precision特殊变量改变精度。

set variableA "10"

set tcl\_precision 5 ;#设置精度为5

set result [expr $variableA / 9.0]

puts $result

输出1.1111

## 4.8Tcl运算符

运算符是一个符号，告诉编译器执行特定的数学或逻辑操作。 Tcl语言有丰富的内置运算符，包括以下几种类型：

* 算术运算符
* 关系运算符
* 逻辑运算符
* 位运算符
* 三元运算符

### 4.8.1算数运算符

Tcl语言支持的算术运算符如下：（变量A=10，变量B=20）



### 4.8.2关系运算符

Tcl语言支持的关系运算符如下：（变量A=10，变量B=20）



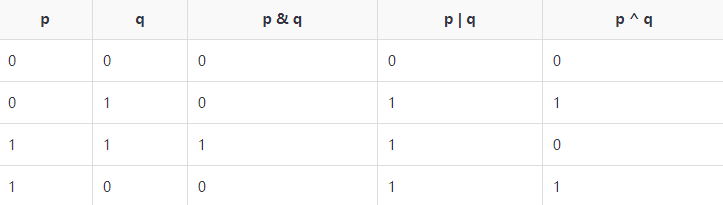
### 4.8.3逻辑运算符

Tcl语言支持的逻辑运算符如下：（A=1，B=0）



### 4.8.4位运算符

位运算符用于进行逐位操作。&, |, 和 ^ 的真值表如下：



Tcl语言支持的位运算符如下：（A=60，B=13）



### 4.8.5三元运算符



### Tcl运算符优先级

运算符优先级影响一个表达式如何进行计算。某些运算符的优先级高于其他运算符;例如，乘法运算符的优先级比加法运算高。

例如：x =7 + 3\* 2;这里，x被赋值13，而不是20，因为运算符 \* 优先级高于+，所以它首先被乘以3 \* 2，然后加上 7。

下表中具有高优先级的操作出现在表格上方，低优先级的在下方

