**L脚本语言语法手册**

0.02版

赵亮

**简 介**

L脚本语言是一个轻量级的，旨在接近自然语言的编程语言，目前支持在中文、英文基础上的编程。并可扩展为任意语种。L脚本语言的语法结构简单，程序结构相对松散，易学易用。

目前L脚本语言仍处于开发初期，功能尚不完善。目前尚未提供源码编辑器，建议使用notepad++或者ultraedit进行源码编辑。

目录

[一、 注释 2](#_Toc416797126)

[二、 对象定义和引用 2](#_Toc416797127)

[三、 整数和浮点数 3](#_Toc416797128)

[四、 数值计算、逻辑运算、位运算 3](#_Toc416797129)

[五、 分支和循环 5](#_Toc416797130)

[六、 字符串和正则表达式 6](#_Toc416797131)

[七、 函数 8](#_Toc416797132)

[八、 文件和目录 9](#_Toc416797133)

[九、 注册表 9](#_Toc416797134)

[十、 时间日期 10](#_Toc416797135)

[十一、 表 11](#_Toc416797136)

[十二、 网络 12](#_Toc416797137)

[十三、 随机数 12](#_Toc416797138)

[十四、 库 13](#_Toc416797139)

[十五、 与C/C++的交互 13](#_Toc416797140)

[十六、 用户交互接口 14](#_Toc416797141)

[十七、 类 14](#_Toc416797142)

[十八、 进程 14](#_Toc416797143)

[十九、 线程 15](#_Toc416797144)

[二十、 调试器使用 15](#_Toc416797145)

[二十一、 其它 15](#_Toc416797146)

1. 注释

L脚本语言的注释是以#打头的行，注释和空行会被解释引擎自动忽略

当解释引擎发现某行代码中包含错误时会尽可能地提供错误信息并简单地将这行代码忽略掉

1. 对象定义和引用

任何对象必须先进行定义之后才能引用，对象定义的格式为

*定义：对象类型，对象名称，初始值*

例如

定义：整数，number1,0

定义：整数，number2,1

定义：浮点数,double,1.0

定义：字符串,string1,”helloworld”

如果想要将对象的内容展示出来，可以这样

显示：number1

显示：string1

:是分隔符用来分隔动作和对象

,是分隔符用来分隔参数列表

在这里，对象的名称可以是除:,以外的任何字符串，中文、英文、韩文都是可以的

1. 整数和浮点数

L脚本语言内置了对整数和浮点数的支持，允许进行四则运算、逻辑运算以及取余数等

整数定义的格式为

*定义：整数，整数的名称，整数的初始值*

浮点数定义的格式为

*定义：浮点数，浮点数的名称，浮点数的初始值*

整数和浮点数定义时必须进行初始化

1. 数值计算、逻辑运算、位运算

数值计算和逻辑运算必须遵从固定的格式

*计算：表达式的内容*

如

计算：number1+number2

计算：number1\*number2

计算：number1>number2

每一行代码的运行结果会保存到一个“结果”对象中，这个对象是全局的

所以如果我们想要把数值计算的结果显示出来，就可以这样写

计算：number1+number2\*100

显示：结果

加减乘除四则运算

加法

计算：number+number2

显示：结果

减法

计算：number1-number2

显示：结果

乘法

计算：number1\*number2

显示：结果

除法

计算：number1/number2

显示：结果

取余数

计算：number1%number2

显示：结果

赋值

计算：number1=number2+1

显示：结果

计算：number1+=number2

计算：number1-=number2

计算：number1\*=number2

计算：number1/=number2

计算：number1%=number2

括号的使用

乘法 除法 取余数 操作的优先级高于 加法和减法 同级的运算保持左结合性，因此如果希望改变运算的顺序，需要通过使用括号，如

计算：(number1+number2)\*100

逻辑运算

L脚本语言中逻辑值"真"等价于数值1, 逻辑值"假"等价于数值0

因此逻辑运算的值仅有1和0两种状态

比较大小

计算：number1>number2

显示：结果

计算：number1>=number2

计算：number1<number2

计算：number1<=number2

计算：number1==number2

逻辑与、逻辑或、逻辑异或、逻辑非

位运算

包括按位取反，按位左移，按位右移，按位与，按位或，按位异或

1. 分支和循环

L脚本语言中支持条件分支，叫做“如果”语句，格式为

*如果：条件表达式或逻辑运算表达式 ，条件为真时执行的语句*

*否则：条件为假时执行的语句*

例如

如果：number1>number2,显示："大于"

否则：显示：“小于”

在L脚本语言中循环语句叫做“当”语句，格式为

*当：条件表达式或逻辑运算表达式，条件为真时重复执行的语句*

例如

当：1，调用：函数，function1

1. 字符串和正则表达式

字符串定义时可以进行初始化，若没有初始化参数，则默认为空串

字符串定义的格式为

*定义：字符串，字符串的名称，字符串的初始值*

例如

定义：字符串,string1,”hello”

定义：字符串，string2," world"

支持字符串的连接操作

格式为

*连接：字符串，第一个字符串，第二个字符串*

连接：字符串,string1,string2

显示：string1

支持字符串取子串的操作

格式为

*取子串：子串名称，原字符串名称，取子串时的其实位置，取子串的长度*

如

定义：字符串,string3

取子串:string3,string1,1,3

允许将字符串的内容清空

格式为

*清空：字符串，字符串的名称*

支持字符串查找子串的操作

格式为

*查找：字符串,原字符串,子串*

如果找到子串，“结果”对象置为一个状态对象，其内容是“找到”否则其内容是“没找到”

如

查找：字符串,string1,string2

支持字符串替换子串的操作

格式为

*替换:字符串,原字符串,字符串中的子串,替换的子串*

和查找一样，如果找到子串，“结果”对象置为一个状态对象，其内容是“找到”否则其内容是“没找到”

如

替换:字符串,string1,string2,"xyz"

正则表达式本身是特殊格式的字符串，定义格式为

*定义:正则表达式,正则表达式的名称,正则表达式的内容*

如

定义:正则表达式,regexp1,"\d\d\d"

如果要使用正则表达式匹配某个字符串，格式如下

定义：字符串,str2,"abc"

定义：字符串,str1,123

匹配：正则表达式,regexp1,str1

匹配：正则表达式,regexp1,str2

1. 函数

和其它编程语言一样，函数必须先定义之后才能进行调用

函数定义的格式为

*定义：函数，函数的名称，形式参数列表*

*开始：函数，函数的名称*

*#函数体*

*结束：函数，函数的名称*

函数的参数可以是0个或任意多个，函数的参数必须是对象或静态数值

函数体可以是任意合法的代码行，但是不允许函数的嵌套定义

函数体必须是由对称的“开始“，“结束”包起来

函数体内部允许定义局部对象，其生存期随函数调用时产生，函数调用结束时销毁

函数调用的格式为

*调用：函数，函数的名称，实际参数列表*

L脚本语言的函数没有返回值，因此，如果我们需要保存函数运行的结果，可以将所需结果对象作为参数传递给函数

1. 文件和目录

L脚本语言中，文件和目录都是内置对象

打开文件或目录的语句会返回一个文件对象，不需要定义

格式为

*打开:文件，文件对象名,文件的完整路径名*

*打开:目录，目录对象名,目录的完整路径*

如：

打开:文件,file1,c:\1.txt

打开:目录,dir1,c:\

对文件对象支持的操作有 复制、移动、删除、取大小、读取、写入

对目录对象支持的操作有 枚举目录中的文件 、枚举目录中的子目录 、复制、 移动、 删除

1. 注册表

L脚本语言内置了对访问Windows注册表的支持

打开一个注册表项会返回一个注册表对象，在这个对象上支持读取、写入、删除操作

语法格式为

*打开：注册表,注册表对象名,注册表路径,值的名称*

*读取：注册表,注册表对象名,字符串对象名*

*写入：注册表，注册表对象名,字符串对象名*

*删除：注册表,注册表对象名*

如

打开：注册表,reg1,HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run,HttpServer

读取：注册表,reg1,string1

写入：注册表，reg1,string1

删除：注册表,reg1

1. 时间日期

L脚本语言中，时间对象的定义方式为

*定义：时间，时间对象的名称*

可以将当前时间读取到时间对象中

*读取：当前时间, 时间对象的名称*

时间对象可以通过比较运算符进行比较

如果想要让脚本在运行过程中等待一段时间，可以调用睡眠语句

*睡眠：需要睡眠的毫秒数*

例如：

定义：时间,time1

读取：当前时间,time1

显示：time1

睡眠：1000

定义：时间,time2

读取：当前时间，time2

显示：time2

如果:((time1<time2)||(time1==time2)),显示："time1早于或等于time2"

1. 表

L脚本语言中的表的概念与Lua语言中的表的概念类似，实际上就是名字与对象之间的映射关系

表中的元素可以是任意合法的L脚本对象，表的大小是动态的，允许动态添加删除元素

表定义的语法格式为

*定义：表,表的名称*

*插入：表,表的名称,对象名*

*取大小，表，整数变量名*

*取元素：表，元素的别名，元素在表中的索引*

例如

定义：字符串,str1,"hello"

定义：字符串,str2,"world"

定义：整数,number1,123456

定义：浮点数,number2,1.75

定义：表,table1

插入：表,table1,str1

插入：表,table1,str2

插入：表,table1,number1

插入：表,table1,number2

定义：函数，function1,参数1，参数2

开始:函数,function1

显示：参数1

显示：参数2

结束:函数,function1

插入：表,table1,function1

显示:table1

1. 网络

目前L脚本支持TCP和UDP协议的网络访问，后台使用阻塞式socket实现

对网络的访问通过网络连接对象和地址对象完成

定义地址对象的格式为

*定义：地址,地址对象的名称,IP(如127.0.0.1),端口号(如27015)*

定义网络连接对象的格式为

*定义：网络连接,网络连接的名称,TCP或UDP*

针对网络连接对象的动作有监听，连接，发送，接收 格式为

*连接: 网络连接的名称, 地址对象的名称*

*发送: 网络连接的名称,字符串对象名*

*接收: 网络连接的名称, 字符串对象名*

详细请参考例子代码

1. 随机数

在程序中我们经常会用到随机信息

在L脚本语言中，随机数是一类对象，实际上是特殊的整数类型

使用方法为

*生成：随机数,随机数对象的名称*

*刷新：随机数, 随机数对象的名称*

例如

生成：随机数,random1

显示：random1

定义：整数,count,1000

定义：函数,genrandom

开始：函数,genrandom

刷新：随机数,random1

显示：random1

计算：count=count-1

结束：函数,genrandom

当：count>0,调用：函数,genrandom

1. 库

L脚本语言支持自定义库以及库的导入机制，以最大程度的提供灵活性，并允许将自定义的函数注册为全局动作

库文件是只包含对象定义和函数定义的脚本文件

库文件必须以#scplib 为起始行

在库中定义的对象为全局用户对象和函数为全局用户函数

使用库时需要首先导入，语法格式为

*导入：库，库文件的完整路径名*

普通用户函数调用时的格式为

*调用：函数，函数的名称，函数的参数列表*

注册为全局动作的函数就可以直接通过函数名来调用，语法格式为

*注册：函数，函数的名称*

*函数：函数的参数列表*

请参考例子代码

1. 与C/C++的交互

由于操作系统提供的API都是C语言的接口，为了实现功能扩展，L脚本语言也允许直接调用操作系统的C语言API函数

目前，仅支持函数调用约定为\_\_stdcall 的C语言函数

1. 用户交互接口
2. 类

L脚本语言支持用户自定义类，以支持面向对象的类型封装

语法格式为

*定义：类，类的名字*

*开始：类，类的名字*

*定义：整数，成员变量，初始值*

*定义：字符串，成员变量字符串，初始值*

*定义：函数，成员函数，成员函数参数*

*开始：函数，成员函数*

*结束：函数，成员函数*

*结束：类，类的名字*

允许调用类的成员函数，语法格式为

*调用：类的名字：成员函数，成员函数参数*

1. 进程

如果我们想要在程序中启动外部程序，可以

*运行:程序,进程的名称,进程可执行文件的路径名,启动参数*

如果想要关闭已启动的外部程序，可以

*关闭:程序, 进程的名称*

例如:

运行:程序,proc1,notepad.exe,c:\hello.txt

关闭:程序,proc1

1. 线程

L脚本支持多线程，允许在脚本中派生新的线程，以支持并发操作

语法格式为

*定义：线程，线程的名称，线程函数*

*运行：线程,线程的名称，线程函数的参数*

在这里，线程函数就是线程执行体所执行的操作，这里和调用函数很相似，但是这个函数是在另一个线程中执行的

详细的使用方法可以参考例子代码

1. 调试器使用

L脚本语言提供了一个简单的调试器，用来进行脚本源代码的调试，在调试器中打开文件后，选中行点击右键即可对代码行设置断点，点击调试按钮即开始调试过程，当脚本执行到设置了断点的行后就会中断下来，并在消息输出窗口中显示提示信息。再次右键点击已设置断点的行即可清除断点。

调试器也支持简单的调试命令

bp 行号 ，用来设置断点

bc 行号 ,用来清除断点

g ，调试运行脚本

sd ，单步调试脚本

s ，继续运行脚本

1. 其它