



BBS.Seeyoon.Com

思远技术论坛 **AU3** 教程

思远技术论坛 虫子樱桃 收集&编写

前言

本人是虫子樱桃，思远的一个菜鸟，有幸能成为思远大家庭的一员。呵呵。在短短的时光里，我在思远认识了龙哥，认识了凌雨姐姐，认识了美丽七彩，认识了远航，认识了思远的兄弟姐妹们。于是，我觉得我应该为大家做点什么。

本人在开学的时候开始学习 AU3，遇到了不少的困难。AU3 是一种很好很强大的语言，我想会有和我一样遇到同样困难的人。所以写了这个很菜的教程。写的不好，还请大大们给点批评。谨以此电子书，献给我们的思远。

思远技术论坛 虫子樱桃

2010 年夏

上篇 初识 **AU3**

I 什么是 AU3

什么是 au3 呢！我一开始也有这种疑问，呵呵。后来经过一段时间的学习与摸索。觉得 AU3 就是一个介于编程语言与脚本之间的东东。首先，她可以用来编写小程序（这不是废话吗，呵呵），她可以编写出带图形界面的程序，并编译成独立运行的可执行文件。其次呢，她是脚本，常见的脚本诸如 vbs，P 处理（我一直把 p 处理认为是一种脚本），虽然也够强大，但是也没有弄出图形界面的美观吧。AU3 就可以，她可以通过 cmdline 调用和执行任何 dos 命令，同时界面上也给用户 DIY 提供了更大的空间，譬如说她可以添加 flash、图片、视频等。呵呵。够强大吧。

AU3 是一种带 basic 风格的脚本，也就是说如果你之前有学习 vb 或者 vbs 之类的语言，那么你就可以快速上手这种语言。下面援引官方说明的一段话，看看我们的 AU3 能干什么吧。

AutoIt 可以做的事:

- 简单易懂的类 BASIC 表达式
- 模拟键盘,鼠标动作事件
- 操作窗口与进程
- 直接与窗口的"标准控件"交互(设置/获取 文字,移动,关闭,等等)
- 脚本可以编译为标准可执行文件
- 创建用户图形界面接口(GUI)
- COM 支持
- 正则表达式
- 直接调用外部 DLL 和 Windows API 函数
- 程序运行功能(让程序运行于其它账户)
- 详细易懂的帮助文件于基于社区的支持论坛
- 完全兼容于 Windows 2000 / XP / 2003 / Vista / 2008
- Unicode 与 64位 运算支持
- 高精度,易使用的数学运算
- 可以运行于 Windows Vista Account Control (UAC)

AutoIt 被设计得尽可能小，并且不用依赖外部 DLL 文件或添加注册表项目即可独立运行。也可以安全的成为服务运行。脚本可以使用 **Aut2Exe** 编译为可独立运行的文件

此外我们还设计了 AutoIt 的 ActiveX 和 DLL 版本 —— **AutoItX** 这是个组件化的语言 (COM 同一 DLL 文件中的标准 DLL 函数). AutoItX 将使得您可以加入一些 AutoIt 独有的

特性到您最常用的脚本语言或程序设计语言中去!

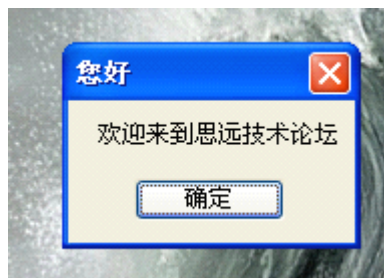
呵呵。我的简单介绍就到这里啦。没有看明白？以后会明白的。西西。

II 我的第一个脚本

和其他语言一样，咱们先编写我们的第一个脚本吧。（ps：当然前提是您已经安装了 AU3，搭建了 AU3 基本运行的环境。最新的汉化版可以去 ACN 下载或者百度下会有很多的，好，切入正题）打开 AU3 的编辑器 SciTE 输入以下内容。

```
Msgbox(0,"您好","欢迎来到思远技术论坛")
```

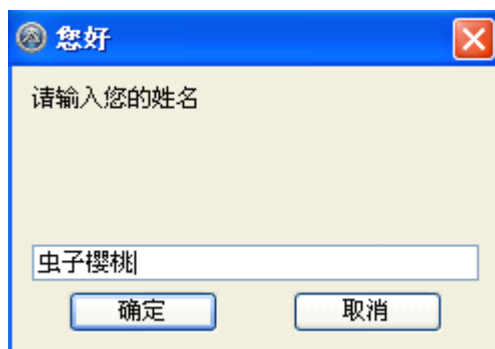
输入后，保存为脚本。文件名任意，只要不改 AU3 这个拓展名就 OK，例如本例中的是"我的第一个脚本.au3"。运行下，会是啥效果呢？看图吧。



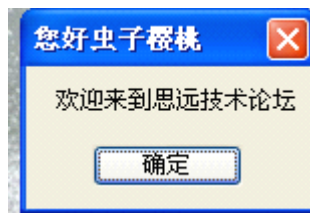
呵呵。很简单吧。接下来我们给它加点交互。同样，在 AU3 的编辑器（其实记事本也可以编写，只不过没有 AU3 自带的强大），输入以下内容

```
Dim $name  
$name=InputBox("您好","请输入您的姓名")  
Msgbox(0,"您好"&$name,"欢迎来到思远技术论坛")
```

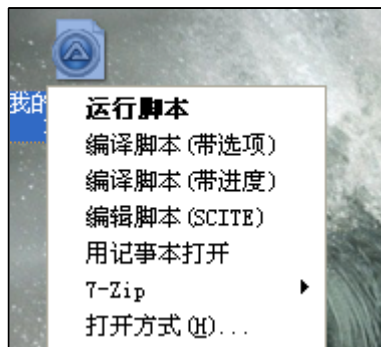
再运行看看。是啥效果呢？



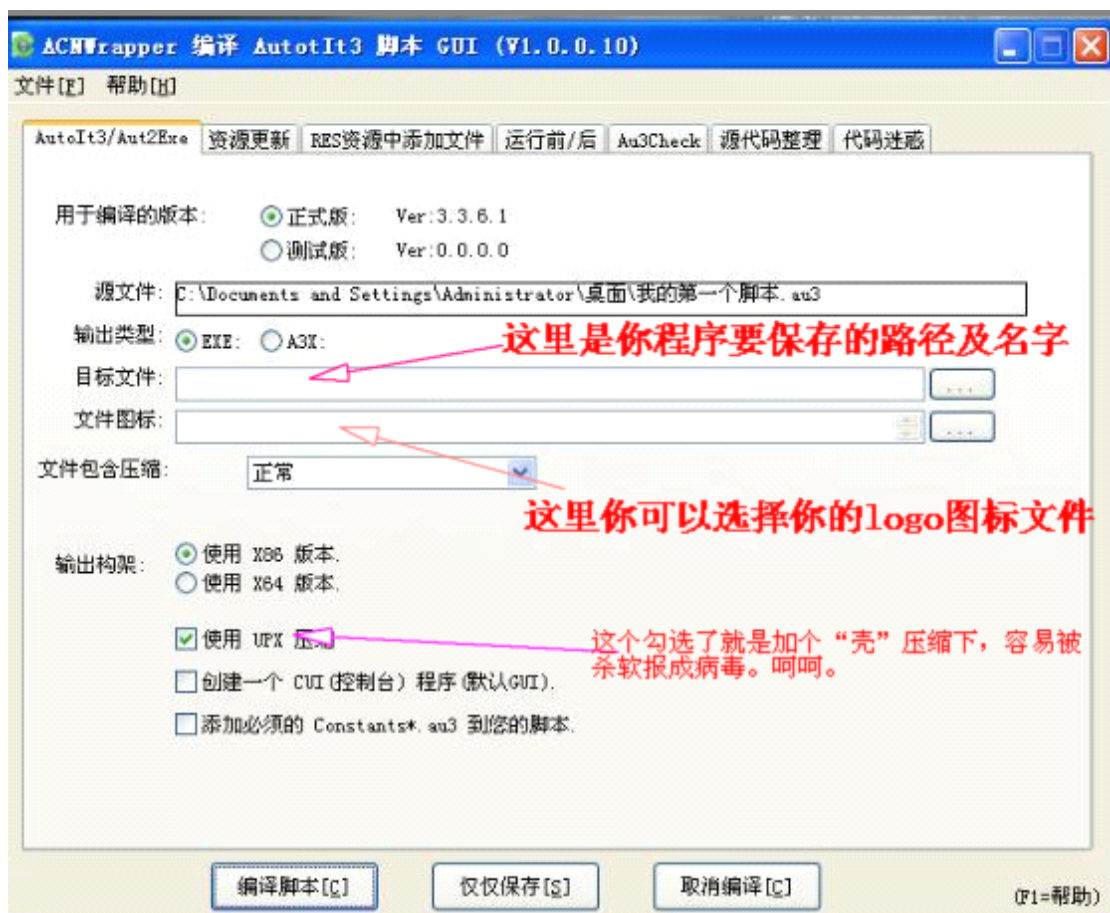
提示输入您的姓名，我输入我的名字。点下确定看看

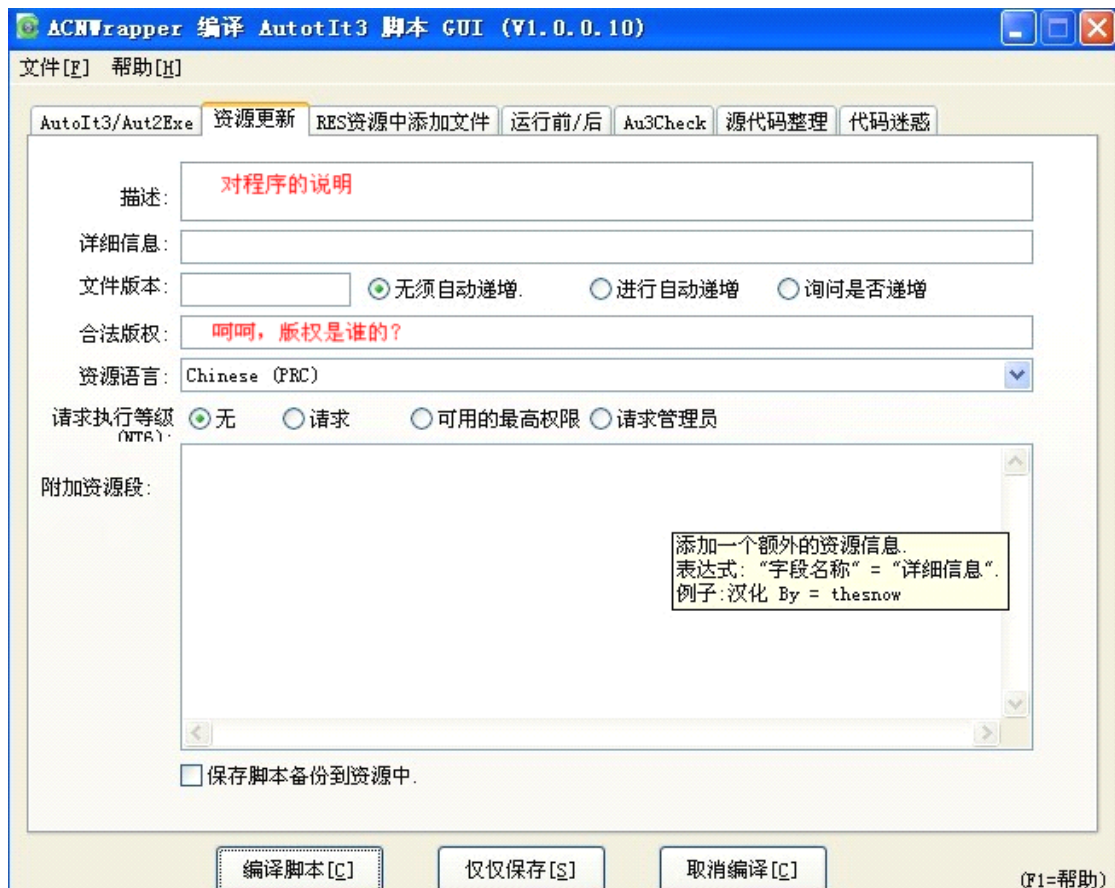


当然咱们编的这个小程序如果没有编译，在别的没有安装 AU3 的机子上是无法运行的，接下来，咱们把它编译成可执行的 EXE 文件。选择刚才我们写的那个脚本，右键看看。



发现没。有两个编译脚本，一个是“带选项”的，一个是“带进度”的。带选项的就是你可以在你要编译的 EXE 文件加上自己的信息，而“带进度”就是直接编译（编译出来有点丑）。咱们来个带选项的吧





我选了一个思远的图标文件，编译出来是这个效果。



是不是很强大呢？下面让我们一起来学习 AU3 吧。

III 编程的基本知识

呵呵。在学习之前，我们先要搞明白自己要学的语言数据类型有哪几种？什么是常量，什么是变量。条件选择结构与循环有哪些。Skyfree 大大的教程《let's autoit》讲述的很详细。我这里只做简单的介绍吧。

A 数据类型

数据类型主要有数字型、字符型和布尔型。

数字型：就是我们常见的 1234567890 这几个。

字符型：在 AU3 中通常用英文的双引号 "" 来包含字符型数据。例如 "你好啊，思远"，就是个典型的字符型数据。同样的，"hello" 也是个字符型数据。发现什么了吗？呵呵。字符型数据既可以是个单词，也可以是个句子。上面学习的数字型数据加了 " " 它也就是字符型数据了。"123" 和 123 是不一样的。

布尔型数据:就是咱们常说的“真”和“假”，只不过它是英文的“true”和“false”。

常用的数据类型就是这三个了（AU3中还有二进制值，基本上不常用，就不介绍了。感兴趣的去看下帮助吧。）。呵呵。下面咱们来讲下什么是常量什么是变量。

B 常量与变量

常量是具有一定含义的名称，用于代替数字或字符串，其值从不改变。呵呵。看的有点头大的感觉吧，简单来说常量就是个具有不变值的东东。最经典的常量就是我们以前学的 π ，呵呵。在 AU3 中常量一般用 `const` 关键字来进行申明。如帮助中的例子：

```
Const $const1 = 1, $const2=12
```

ps:常量不能声明为一个已经存在的变量.

该说变量了。变量是什么呢？以前看 sxd 的 AU3 教程，觉得他讲的不错，呵呵。以下是摘录他的教程：

变量，就是程序里存放数据的容器。可以把变量想象为用来放东西的箱子，比如苹果和苹果箱。变量名，就是这个箱子的名称，因为我们会需要很多的箱子来放苹果，大苹果，小苹果，红苹果....au3 里变量名以美元符号（\$）开头，\$abc, \$123 这些都是变量名我们早上采了苹果，准备晚上吃，这时候我们就需要把苹果放进箱子里，等到晚上再拿出来用。
\$apple \$apple \$apple \$apple = = = 10101010

这时候 \$apple 这个变量里就装了一个数字类型的数据 10，到我们想用这个数据时候只要把 \$apple 喊出来就可以了。

在 AU3 中变量使用关键字 `Dim`, `Local` 和 `Global` 来声明并创建变量。一般变量是以美元符号 \$ 开头的。如下面例子：

之前我们说过 AU3 具有 vb 和 VBS 相似的语法风格。所以在声明多个变量时，还可以像下面这样来写：

```
Dim $a,$a1,$a2
```

相当于

```
Dim $a
```

```
Dim $a1
```

```
Dim $a2
```

C 运算符

AutoIt 支持以下这些赋值符号,数学运算符,比较和逻辑运算符。（转自帮助文件）

运算符	详细信息
	赋值运算
=	赋值,如 \$var = 5 (赋值数字 5 到 \$var)
+=	自增赋值,如 \$var += 1 (添加 1 到 \$var)
-=	自减赋值.
*=	自乘赋值.
/=	自除赋值.

&=	连续赋值. 如 \$var = "one" , 然后 \$var &= 10 (\$var 的结果为 "one10")
数学运算	
+	使两个数相加. 如 10 + 20 (等于 30)
-	使两个数相减. 如 20 - 10 (等于 10)
*	使两个数相乘. 如 20 * 10 (等于 200)
/	使两个数相除. 如 20 / 10 (等于 2)
&	使两个字符串连接起来. 比如 "one" & 10 (等于 "one10")
^	提高某个数的幂. 比如 2 ^ 4 (2 的 4 次方, 等于 16)
比较运算 (大小写敏感的字符串需要使用 == 来比较)	
=	判断两个值是否相等. 比如 If \$var= 5 Then (如果变量 \$var 的值为 5 则条件成立). 用于字符串时不区分大小写
==	判断两个字符串是否相等. 左方和右方的值将会转化成字符串, 并区分大小写, 这个运算只能用于区分字符串大小写的比较.
<>	判断两个值是否不相等. 比较会对字符串大小写敏感. 要比较一个大小写敏感的不等于操作使用 Not ("string1" = "string2")
>	判断第一个值(左边)是否大于第二个值(右边). Strings are compared lexicographically even if the contents of the string happen to be numeric.
>=	判断第一个值(左边)是否大于或等于第二个值(右边). Strings are compared lexicographically even if the contents of the string happen to be numeric.
<	判断第一个值(左边)是否小于第二个值(右边). Strings are compared lexicographically even if the contents of the string happen to be numeric.
<=	判断第一个值(左边)是否小于或等于第二个值(右边). Strings are compared lexicographically even if the contents of the string happen to be numeric.
逻辑运算	
AND	逻辑与运算. 如 If \$var = 5 AND \$var2 > 6 Then (如果变量 \$var 的值为 5 而且变量 \$var2 的值大于 6 则条件成立)
OR	逻辑或运算. 如 If \$var = 5 OR \$var2 > 6 Then (如果变量 \$var 的值为 5 或者变量 \$var2 的值大于 6 则条件成立)
NOT	逻辑非运算. 如 NOT 1 (结果为 False)

当一个表达式内含有多个运算符时, 其结合的先后顺序由 **运算符的优先级别**来控制. Autolt 中运算符的优先级如下所示. 处于同一优先级的两种运算符将按 **从左到右**的顺序结合. 越上面的运算符则优先级越高:

NOT

^
 * /
 + -
 &
 <> <= >= <> ==
AND OR

例如表达式 **2 + 4 * 10** 的值将是 **42**,结合顺序如下:

4 * 10 (结果为40)

2 + 40 (结果为42)

乘号 * 拥有比加号 + 更高的优先级. 会进行**先乘后加**

您还可以使用括号来使表达式内的某些部分优先被计算.

如 **(2 + 4) * 10** 结果等于 **60**.

D 条件选择结构

- [If...Then...Else](#)

(字面意思:如果(某个条件成立)...那么(执行某些操作)...否则(执行另外一些不同的操作)
 这里有点像 VBS 里面的判断。呵呵。几乎是一样的。举个简单的小例子:

```

dim $age
$age=inputbox("您好","请输入您的年龄")
if $age="" then
msgbox(0,"额","想忽悠在下")
else
msgbox(0,"您好","您的年龄是"&$age&", 谢谢您的合作!")
endif

```

呵呵，这个很简单的。不过虫子还是要八婆一句，就是写了 IF 要记得写 ENDIF 哦，虫子以前老爱那么干。结果脚本运行出错，可是又不是像 VBS 那样提示我语句未结束。看看我下面这个例子，IF 是没有结束的哦！

```

include <GUIConstants.au3>
#include <ButtonConstants.au3>
#include <GUIConstantsEx.au3>
#include <WindowsConstants.au3>
#Region ### START Koda GUI section ### Form=
$Form1 = GUICreate("测试", 403, 135, 192, 124)
$Button1 = GUICtrlCreateButton("你好", 112, 32, 107, 41)
GUISetState(@SW_SHOW)
#EndRegion ### END Koda GUI section ###
While 1
    $nMsg = GUIGetMsg()

```

```

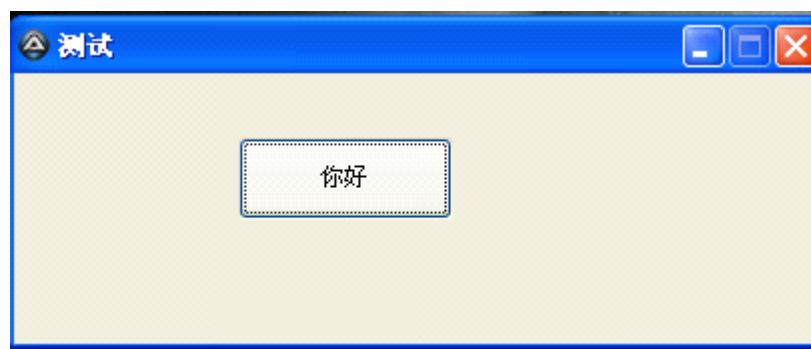
Switch $nMsg
Case $GUI_EVENT_CLOSE
Exit
Case $Button1
$sc=InputBox("你好","嘿嘿，我是思远的虫子，请在下面输入字符")
If $sc="" Then
MsgBox(0,"额","你貌似没有输入哦")
Else
MsgBox(0,"嘿","你输入的是"&$sc)
EndSwitch
WEnd

```

看看运行了是什么情况。记得哦，我们是 IF 有错误。



我们把 ENDIF 加上，运行看看。



运行正常吧。呵呵。所以说好的习惯最重要。虫子没有好的习惯，所以走了不少弯路，希望大家不要学我。O(╯╰╯)o 唉。也许上面的代码暂时看不懂，先不要管它啥意思，这只是个测试，以后大家会慢慢明白其中含义的。

- [Select...Case](#)

(这是开关语句,根据某个表达式的多种不同的值来选择执行不同的语句)

if...then...else 用来处理简单的判断还可以,但是处理多判断就有点力不从心了(我觉得 IF 语句用于那种判断条件"是"与"不是"的程序比较合适。而 select 语句比较适合条件的筛选。),呵呵,并不是它不能写出来,但代码写出来相对较冗长。下面我们就写个同样的内容。

```

dim $age
$age=inputbox("您好","请输入您的年龄")
select
case $age=""

```

```

    msgbox(0,"额","想忽悠在下")
case $age>10 and $age<18
    msgbox(0,"您好","小朋友")
case $age>=18 and $age<26
    msgbox(0,"您好","小伙子")
case $age>=26
    msgbox(0,"您好","朋友")
endselect

```

[Switch...Case](#)

(同上)

E 循环结构

什么叫循环？循环就不断得执行同一个东西。永远不会停止的循环，叫死循环。呵呵。感叹下：爱情是我们人生的一个死循环。在 AU3 中，循环有以下几种。

- [For...Next](#)

这个循环结构通常用来执行知道要执行次数的循环。基本结构如下：

For < 变量 > = < 开始 > **To** < 停止 > [**Step**]

语句 ...

Next

【说明】说明下那个 step 参数是什么。Step 表示的是步进值。步进值表示的是在初值上增加的大小。默认是1.比如说下面这个例子我们把步进值

```

for $i=1 to 10 step 1
msgbox(0,"您好啊","欢迎来看虫子樱桃的 AU3 教程")
next

```

上面这个会运行十次对话框而下面这个只会运行五次。

```

for $i=1 to 10 step 2
msgbox(0,"您好啊","欢迎来看虫子樱桃的 AU3 教程")
next

```

[While...Wend](#)

这个循环的字面意思就是”当...的时候...”学过英语的同志应该知道一个连词 while 吧。呵呵。这里就是这个连词的意思。这个循环用来表示在某种条件下执行。也就是说，当条件满足就执行相应的动作。它的基本格式如下：[参考的帮助文档]

While <表达式>

其它语句

...

Wend

参数

表达式	若该表达式的值为真则重复执行循环体语句（以 WEnd 为结束标志），否则循
-----	---------------------------------------

环结束。

注意

While...WEnd 中的语句必须嵌套使用.

由于程序在每次执行循环体语句前需计算循环控制表达式的值, 只有当其为真时才执行循环语句, 因此循环体语句可能一次都不被执行.

要创建一个无限循环, 您只需设置 表达式 为非 0 数字.

看看下面的例子:

```
Dim $ok
$ok=InputBox("您好","请输入一个大于 10 的数, 小于或者等于会没有反应哦")
While $ok>10
MsgBox(0,"谢谢合作","您输入的是正确的")
sleep(1900)
exit
WEnd
```

- **Do...Until**

这个循环结构是怎么解释的呢? 呵呵。看见 Until 没。学过外语的应该知道, not....until 这个句型, 意思是"直到...才...". 这个结构用外语解释就是 do not stop until (直到...的时候才停止)。这个结构用来控制程序的运行, 当某种条件满足时, 就退出。同样的, 我们来个例子。

```
Dim $num
$num=Random(1,6,1)
Do
MsgBox(0,"虫子提示您","随机数当前值是"&$num)
Until $num<5
```

- **For...In...Next**

(暂时不讲解, 呵呵)

下篇 常见 **AU3**函数

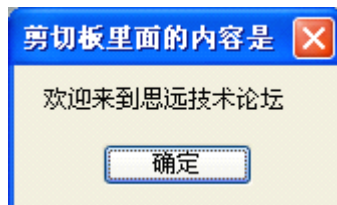
I 环境变量管理

[1]ClipGet () 获取剪切板里的文本.

这个函数是用来获取剪切板里面的内容的。呵呵。咱们做个实验。打开一个记事本，在里面输入"欢迎来到思远技术论坛"，然后复制。编写如下代码并运行之：

```
Dim $nr  
$nr=ClipGet()  
MsgBox(0,"剪切板里面的内容是",$nr)
```

出现的是这个画面



当然这个函数不是万能的，当我们复制的内容是非文本文件时，它的返回值就是空。比如我们复制一个文件，执行相同的代码。提示会是下面这样的。



呵呵，是文件的绝对路径。

[2]ClipPut ("要写入剪切板的文本.") 在剪切板中写入文本.

这个函数相对来说就功能一般啦。不过使用下面这个代码可以清空剪切板，这倒是很实用。

```
ClipPut("")
```

[3]EnvGet ("环境变量") 返回指定的系统环境变量的值.。

[4]EnvSet ("环境变量" [, "值"]) 写入或修改一个环境变量.

【注意哦】目标环境变量的值.使用时若没有指定此参数则该环境变量将被删除.此函数设置的环境变量只能被那些由 AutoIt 启动的程序(比如使用 Run 或 RunWait)访问.一旦 AutoIt 被关闭则该环境变量将不复存在.

环境变量一般是指在操作系统中用来指定操作系统运行环境的一些参数，比如临时文件夹位置和系统文件夹位置等。这点有点类似于 DOS 时期的默认路径，当你运行某些程序时除了在当前文件夹中寻找外，还会到设置的默认路径中去查找。简单地说这里的“Path”就是一个变量，里面存储了一些常用命令所存放的目录路径。

环境变量相当于给系统或用户应用程序设置的一些参数，具体起什么作用这当然和具体的环境变量相关。比如 path，是告诉系统，当要求系统运行一个程序而没有告诉它程序所在的完整路径时，系统除了在当前目录下面寻找此程序外，还应到哪些目录下去寻找；再如

tc 或 vc++ 中, `set include=path1;path2;` 是告诉编译程序到哪里去找 .h 类型的文件; 当然不仅仅是指定什么路径, 还有其它的作用的, 如 `set dircmd=/4` 设置一个环境变量的作用是在使用 `dir` 命令时会把 /4 作为缺省的参数添加到你的 `dir` 命令之后, 就像你的每个命令都加了 /4 参数, 它实际上是给命令解释程序 `command` 设置的一个环境变量, 并且是给 `dir` 这个内部命令设置的。 `DWORD GetEnvironmentVariable(LPCSTR lpName, LPSTR lpBuffer, DWORD dSize)`, 参数 `lpName` 是你要求查询的环境变量的名, `lpBuffer` 是返回你所指定的环境变量的值的, `dSize` 是告诉这个函数 `lpBuffer` 可以存放多少个字节。 分析本地故障时原因很可能就是因为环境变量中的默认路径被删除的结果, 默认路径一经设置, 当前系统如有程序运行时需要某些 DLL 或 EXE 文件, 以及 Active 控件时就会到所有默认路径中去查找, 如果在这些目录中查找到相应的程序则自动加载, 查找不到则报告缺少某某文件的错误信息。

环境变量的作用

解决双系统的软件共用问题 很多朋友会在自己的计算机上安装双系统, 例如 C 盘安装 Windows 98, D 盘安装 Windows XP。可是某些软件往往只在 Windows 98 系统中安装, Windows XP 系统中是无法正常使用的, 比较麻烦却有效的方法是再安装一遍。当我们了解了环境变量中的用途后就可以很好解决双系统的软件共用问题。

解决系统运行问题 为什么在 Windows 98 中安装了的软件在 Windows XP 下无法运行呢(绿色软件除外)? 原因是安装软件时往往须要向系统目录中复制某些文件, 而使用另外一个系统时会由于缺少这些文件而无法运行。因此, 我们可以通过设置环境变量的方法来解决这个问题。

设置环境变量有两种方式:

第一种是在命令提示符运行窗口中设置; 第二种是通过单击“我的电脑→属性→高级”标签的“环境变量”按钮设置。需要注意的是, 第一种设置环境变量的方式只对当前运行窗口有效, 关闭运行窗口后, 设置就不起作用了, 而第二种设置环境变量的方式则是永久有效。

2. 如何在命令提示符窗口中设置环境变量? 在“开始→运行”框中输入“`cmd`”后按“确定”按钮, 出现命令运行窗口。在命令提示符下输入“`set`”即可查看环境变量设置。要查看具体某个环境变量的设置, 比如要查看 `path` 环境变量的设置, 可以输入“`set path`”。要创建一个环境变量, 比如要创建一个名为 `aa` 的, 值为“`c:`”的环境变量, 可以输入“`set aa=c:`”命令。而要删除一个环境变量, 比如要删除 `aa` 环境变量, 则可输入“`set aa=`”命令(注意=后面不能有空格)。如何更改一个环境变量的设置呢? 更改环境变量有两种情况: 一是追加方式, 即在不改变环境变量现有设置的情况下, 增加变量的值, 比如要给环境变量 `aa` 增加一个值为“`D:`”的设置, 可以输入“`set aa=%path%;D:`”。另一种是完全修改方式, 对于这种方式, 我们可以采用直接创建一个环境变量的方法来实现。

3. 用户变量和系统变量的关系是什么? 点击“我的电脑→属性→高级”标签的“环境变量”按钮, 出现“环境变量”对话框, 如果当前是以 Administrator 登录系统的用户, 对话框的上面为 Administrator 的用户变量, 对话框的下面为系统变量(即相当于系统中所有用户的用户变量)。有的时候我们会看到在用户变量和系统变量中都存在某一个环境变量, 比如 `path`, 那么 `path` 的值到底是用户变量中的值还是系统变量中的值, 或者两者都不是呢? 答案是两者都不是。`path` 变量的值是用户变量中的值与系统变量中的值的叠加。环境变量和环境变量的值不要含有空格, 也不要使用中文, 切记!

常见环境变量

`%ALLUSERSPROFILE%` 局部 返回所有“[用户配置文件](#)”的位置。

`%APPDATA%` 局部 返回默认情况下应用程序存储数据的位置。

`%CD%` 局部 返回当前目录字符串。

%CMDCMDLINE% 局部 返回用来启动当前的 **Cmd.exe** 的准确命令行。

%CMDEXTVERSION% 系统 返回当前的“命令处理程序扩展”的版本号。

%COMPUTERNAME% 系统 返回计算机的名称。

%COMSPEC% 系统 返回命令行解释器可执行程序的路径。

%DATE% 系统 返回当前日期。使用与 **date /t** 命令相同的格式。由 **Cmd.exe** 生成。有关 **date** 命令的详细信息，请参阅 **Date**。

%ERRORLEVEL% 系统 返回最近使用过的命令的错误代码。通常用非零值表示错误。

%HOMEDRIVE% 系统 返回连接到用户主目录的本地工作站驱动器号。基于主目录值的设置。用户主目录是在“本地用户和组”中指定的。

%HOMEPATH% 系统 返回用户主目录的完整路径。基于主目录值的设置。用户主目录是在“本地用户和组”中指定的。

%HOMESHARE% 系统 返回用户的共享主目录的网络路径。基于主目录值的设置。用户主目录是在“本地用户和组”中指定的。

%LOGONSERVER% 局部 返回验证当前登录会话的域控制器的名称。

%NUMBER_OF_PROCESSORS% 系统 指定安装在计算机上的处理器的数目。

%OS% 系统 返回操作系统的名称。**Windows 2000** 将操作系统显示为 **Windows_NT**。

%PATH% 系统 指定可执行文件的搜索路径。

%PATHEXT% 系统 返回操作系统认为可执行的文件扩展名的列表。

%PROCESSOR_ARCHITECTURE% 系统 返回处理器的芯片体系结构。值: **x86**, **IA64**。

%PROCESSOR_IDENTIFIER% 系统 返回处理器说明。

%PROCESSOR_LEVEL% 系统 返回计算机上安装的处理器的型号。

%PROCESSOR_REVISION% 系统 返回处理器修订号的系统变量。

%PROMPT% 局部 返回当前解释程序的[命令提示符](#)设置。由 **Cmd.exe** 生成。

%RANDOM% 系统 返回 0 到 32767 之间的任意十进制数字。由 **Cmd.exe** 生成。

%SYSTEMDRIVE% 系统 返回包含 **Windows XP** 根目录（即系统根目录）的驱动器。

%SYSTEMROOT% 系统 返回 **Windows XP** 根目录的位置。

%TEMP% and %TMP% 系统和用户 返回对当前登录用户可用的应用程序所使用的默认临时目录。有些应用程序需要 **TEMP**，而其它应用程序则需要 **TMP**。

%TIME% 系统 返回当前时间。使用与 **time /t** 命令相同的格式。由 **Cmd.exe** 生成。有关 **time** 命令的详细信息，请参阅 **Time**。

%USERDOMAIN% 局部 返回包含用户帐户的域的名称。

%USERNAME% 局部 返回当前登录的用户的名称。

%UserPrefix% 局部 返回当前用户的配置文件的位置。

%WINDIR% 系统 返回操作系统目录的位置。

II 文件,目录和驱动器管理

[5]DirCopy ("源目录", "目标目录" [, 标志])

复制指定目录及其所有子目录和文件(类似于 **xcopy** 命令)。

参数

源目录	要复制的文件夹的路径(结尾不需反斜线符号).例如: "C:\Path1"
目标目录	要复制到的目标路径(结尾不需反斜线符号).例如: "C:\Path_Copy"
标志	[可选参数] 此标志参数用以决定是否覆盖已存在的文件: 0 = (默认)不覆盖已存在的文件 1 = 覆盖已存在的文件

[6]DirCreate ("文件夹路径") 新建一个目录/文件夹.

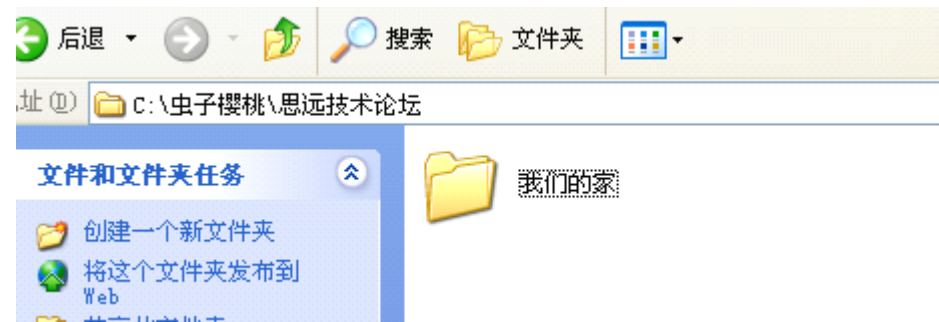
如我们运行下面的脚本

DirCreate ("C:\虫子樱桃")

就在 C 盘的根目录下建立了一个以我名字命名的文件夹。在帮助文件中有这么一句话“在要创建的目录的父目录(上一级目录)不存在时,这些父目录也会被创建.”,这句话的意思是创建文件夹的上级目录若不存在,那么他的上级目录也会被自动创建。运行下面的代码:

DirCreate ("C:\虫子樱桃\思远技术论坛\我们的家")

无图无真相,结果如下。



[7]DirGetSize ("路径" [, 标志]) 返回给点目录所占用的空间(单位字节).

参数

路径	要得到文件大小占用的目录, 例如. "C:\Windows".
标志	[可选参数] 此标志决定了本函数的行为及结果,它的值可以是下列数值的组合(数值相加): 0 = (默认) 1 = 启用扩展模式 -> 返回一个包含扩展信息的数组(请查看下面的 注意部分). 2 = 子目录下的文件大小将不计算入内(递归模式被取消)

返回值

成功: 返回值 >= 0,占用空间的大小.

失败: 返回 -1,并把 @error 设为 1, 说明目标路径并不存在.

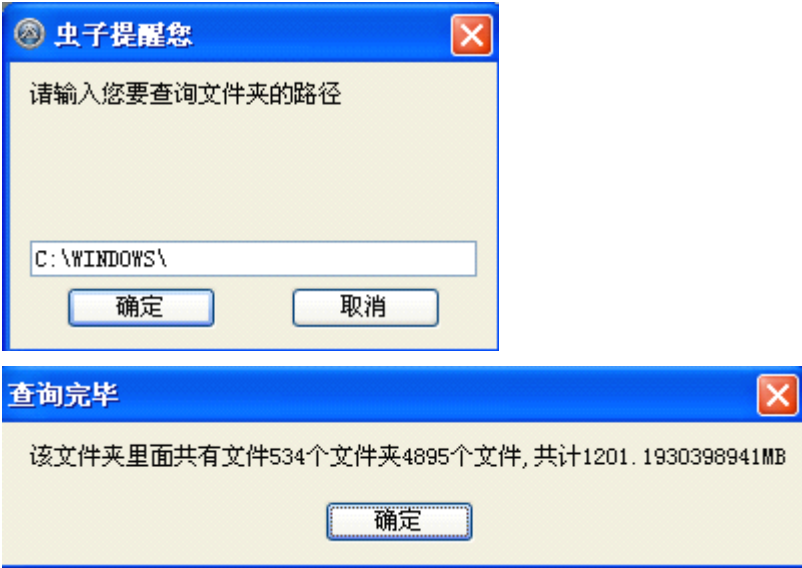
注意/说明

如果脚本程序被暂停则此函数也随之暂停,直到脚本恢复执行为止!

在使用扩展模式时,本函数所返回的数组是个含有下列元素的一维数组:

\$array[0] = 空间大小

\$array[1] = 文件数
\$array[2] = 文件夹数
我们来写一个脚本，测试下这个函数：
Dim \$folder
\$folder=DirGetSize("C:\WINDOWS\",1)
MsgBox(0,"查询完毕","该文件夹里面共有文件"&\$folder[2]&"个文件夹"&\$folder[1]&"个文件,共计"&\$folder[0]/(1024*1024)&"MB")
再加点交互，就是我们的一个程序了-----判断任意文件夹大小。代码如下
Dim \$folder,\$way
\$way=InputBox("虫子提醒您","请输入您要查询文件夹的路径")
\$folder=DirGetSize(\$way,1)
MsgBox(0,"查询完毕","该文件夹里面共有文件"&\$folder[2]&"个文件夹"&\$folder[1]&"个文件,共计"&\$folder[0]/(1024*1024)&"MB")
运行界面（呵呵，在以后的例子我就尽量不上图了，为什么？呵呵。文件体积过大。）：



【注意哦】1MB=1024KB=1024*1024Byte
[8]DirMove ("源目录", "目标目录" [, 标志]) 移动指定目录及其所有子目录和文件。
参数

源目录	要移动的文件夹的路径(结尾不需反斜线符号).例如: "C:\Path1"
目标目录	要移动到的目标路径(结尾不需反斜线符号).例如: "C:\Path_Copy"
标志	[可选参数] 此标志参数用以决定是否覆盖已存在的文件: 0 = (默认)不覆盖已存在的文件 1 = 覆盖已存在的文件

注意/说明

若源目录和目标目录位于不同的卷标或 UNC 路径已被使用,则本函数将执行复制/删除操作而不是直接移动.

若目标目录已存在并指定了覆盖标志(参数),则源目录将被移动到目标目录 里面.

由于 AutoIt,并没有"DirRename" (目录重命名)函数,请使用 DirMove 函数来重命名文件夹!

[9]DirRemove ("路径" [, 递归遍历]) 删除一个目录/文件夹.

参数

路径	要删除的文件夹的路径.
递归遍历	[可选参数] 此标志参数用以决定是否删除子目录. 0 = (默认)不删除文件及子目录 1 = 删除文件及子目录(类似于 DOS 下的 DelTree 命令)

[10]DriveGetDrive ("类型") 返回一个含有指定驱动器盘符的数组.

参数

类型	要获得的驱动器类型: "ALL(全部)", "CDROM(光驱)", "REMOVABLE(可移动驱动器)", "FIXED(固定驱动器)", "NETWORK(网络驱动器)", "RAMDISK(内存盘)", or "UNKNOWN(未知类型)"
----	---

返回值

成功: 以字符串数组的形式返回所找到的驱动器的盘符(驱动器盘符后接一个冒号).数组中的0号元素储存驱动器的数量.

失败: 返回 "" (空字符串),并把 @error 设为 1.

[11]DriveGetFileSystem ("路径") 返回指定驱动器的文件系统类型.

返回值

成功: 以字符串的形式返回指定驱动器的文件系统类型,请查看下面的列表.

失败: 设置 @error 为 1.

返回值	详细信息
1 (数字)	指定驱动器没有媒体(CD, 软驱, Zip 等)或者媒体格式未知(RAW).
"FAT"	通常是那些容量低于500MB 的驱动器所使用的文件系统,比如软驱,内存盘,USB"笔式"驱动器等等.
"FAT32"	Windows 9x/Me 下的硬盘分区所使用的文件系统.
"NTFS"	Windows NT/2000/XP 下的硬盘分区所使用的文件系统.
"NWFS"	Novell 网络文件服务器所使用的文件系统.
"CDFS"	通常是 CD(也可能是虚拟光驱软件挂载的 ISO 镜像).

"UDF"	通常是 DVD
-------	---------

[12]DriveGetLabel ("路径") 返回指定的驱动器分区的卷标(如果存在).

[13]DriveGetType ("路径") 返回指定驱动器的类型.

返回值

成功: 返回指定驱动器的类型: "Unknown"(未知类型), "Removable"(可移动), "Fixed"(固定的), "Network"(网络), "CDROM"(光驱), "RAMDisk"(内存盘)

失败: 返回 ""(空字符串), 并将 @error 设为 1.

[14]DriveSetLabel ("路径", "卷标") 修改指定驱动器的卷标.

参数

路径	目标驱动器的路径.
卷标	指定驱动器的新卷标(最长11个字符)

注意/说明

大多数的硬盘驱动器(分区)的卷标最长可达11个字符,因此如果指定的卷标长度超出限制则 DriveSetLabel 函数会出错.此外,如果目标分区的文件系统是 FAT32 则其卷标的所有字符会自动转换为大写字符.

[15]DriveSpaceFree ("路径") 以 MB(兆字节)为单位返回指定路径所在分区的剩余空间.

参数

路径	要获得相应信息的驱动器路径.
----	----------------

返回值

成功: 以 MB(兆字节)为单位返回指定路径所在分区的剩余空间(浮点数).

失败: 返回值为 0,并把 @error 设为 1.

注意/说明

即使给定的路径是某驱动器子目录的完整路径(必须是确实存在的)DriveSpaceFree 也能计算出其所在分区的可用空间;但如果指派的是一个文件的路径则不行.

如果对精确度的要求不高则可使用 Round 函数(四舍五入).

[16]FileCopy ("源文件", "目标路径" [, 标志]) 复制一个或多个文件.

参数

源文件	要复制的文件的路径.可使用通配符.
目标路径	要复制到的目标路径.

标志	<p>[可选参数] 此标志参数用以决定是否覆盖已存在的文件。可以是下面的这些值:</p> <p>0 = (默认)不覆盖已存在的文件</p> <p>1 = 覆盖已存在的文件</p> <p>8 = 当目标文件夹不存在,就自动创建一个 (参考 注意 部分)。</p>
----	--

返回值

成功: 返回值为1.

失败: 返回值为0.

注意/说明

目标文件夹必须确实存在,除非把标志设置为8.

例如组合标志'9'(1+8)覆盖存在的目标文件,并且如果目标目录结构不存在,就自动创建一个.

关于通配符请查看 [FileFindFirstFile](#) 上的说明.

一些文件属性不能替换(如只读等).

[17]FileCreateNTFSLink ("源路径", "硬链接" [, 标志]) 创建一个 NTFS 硬连接到一个文件或者文件夹.

参数

源路径	创建硬链接的源路径.(需要连接的位置).
硬链接	要映射到的路径.
标志	<p>[可选参数] 这个标志决定是否覆盖已经存在的连接。可以是下面的值:</p> <p>0 = (默认) 不覆盖已经存在的连接.</p> <p>1 = 覆盖存在的连接.</p>

注意/说明

目标目录必须已经存在.

只能工作于 NTFS 文件系统的卷.

如果源路径是一个文件,硬盘连接必须在同一分区.

如果源路径是一个目录,硬盘连接可以不在同一分区.

FileDelete 或者 FileMove 可以用于硬盘连接.

要管理连接可以使用 [NTFSLink](#) 程序.

[注意哦]硬连接是给文件一个副本,同时建立两者之间的连接关系。修改其中一个,与其连接的文件同时被修改。如果删除其中任意一个其余的文件将不受影响。

硬连接”让一个文件在多个目录下重复出现,但只占用一份文件的空间。例如,一个大小为 10MB 的“E:\a.doc”的文件,创建一个硬连接到“E:\共享文件夹\a.doc”中,则两个“a.doc”文件只占用一个文件的空间 (10MB),两个文件的内容当然是完全一样的。如果编辑该文件的编辑器在修改保存文件时,依然保持源文件在 NTFS 文件夹中的位置,那么在“E:\a.doc”上所作的改动会自动反映到“E:\共享文件夹\a.doc”中。同理,修改了“E:\共享文件夹\a.doc”文件后,所作的修改也会立即反映到硬连接对应的“E:\a.doc”文件。

创建硬连接之后，删除硬连接的副本（即连接点）只会将文件系统对文件数据的引用减一，只有删除所有的引用时，才真正删除文件。

“硬连接”还可以用于目录，这称为“目录连接点”（Junction）。例如，可以为“C:\Applications”目录创建一个硬连接到“E:\共享文件夹\Applications”，则两个目录共享同样的存储空间，在“C:\Applications”目录中对任何文件所作的改动都会立即反映到“E:\共享文件夹\Applications”目录中对应的文件中去。同理，修改“E:\共享文件夹\Applications”目录中的文件，改动也会立即反映到“C:\Applications”目录中去。

对于目录连接点需要注意：目录连接点连接到原来的目录，如果将源目录删除，则其连接点将不再可用，这与文件硬连接是不同的。

软连接也叫符号连接，他只是对源文件在新的位置建立一个“快捷（借用一下 windows 常用词）”，所以，当源文件删除时，符号连接的文件将成为无源之水->仅仅剩下个文件名了，当然删除这个连接，也不会影响到源文件，但对连接文件的使用、引用都是直接调用源文件的。

[18]FileCreateShortcut ("目标文件", "lnk 文件" [, "工作目录" [, "参数" [, "描述" [, "图标文件" [, "快捷键" [, 图标编号 [, 状态]]]]) 创建指定文件的快捷方式(lnk 文件).

参数

目标文件	要创建快捷方式的文件的完整路径.
lnk 文件	快捷方式文件(*.lnk)的完整路径.
工作目录	[可选参数] 工作目录(起始位置).
参数	[可选参数] 额外的文件参数.
描述	[可选参数] 文件描述(备注).
图标文件	[可选参数] 要使用的图标文件的完整路径.
快捷键	[可选参数] 快捷键,格式和 Send() 函数能使用的一样.
图标编号	[可选参数] 要使用的图标编号(通常是0).
状态	[可选 参 数] 快 捷 方 式 运 行 时 的 起 始 状 态 (运 行 方 式). 可 使 用 @SW_SHOWNORMAL,@SW_SHOWMINNOACTIVE 或 @SW_SHOWMAXIMIZED

注意/说明

Windows 下 快捷方式的快捷键可以是这些形式:Ctrl+Alt+X,Ctrl+Shift+X,Shift+Alt+X,Ctrl+NumPadKey 或 Alt+NumPadKey(NumPadKey 指数字键盘上的按键),其中 X 代表各种字母,标点符号或功能键.如果指定的快捷键不合法则 Windows 将视为默认的 Ctrl+Alt.

Windows 把数字键盘上的按键和主键盘区上的(同符号的按键,包括数字键和标点符号键)视为不同的两种按键.此外,FileCreateShortcut 函数允许用户创建快捷键为 Ctrl+X 或 Alt+X 等格式的快捷方式(而正常情况下 Windows 仅允许 X 处是数字键盘上的按键);但是,您应

该尽可能避免指派这种快捷键,因为它们很容易跟标准的应用程序快捷键冲突.

Windows 不允许使用 ESC,ENTER,TAB,SPACEBAR (空格),PRINT SCREEN,SHIFT 或 BACKSPACE (退格)等按键作为快捷键.

FileCreateShortcut 函数并不严格要求目标文件,工作目录,图标或快捷键等参数的合法性,一般都可"成功"创建 LNK 文件;但 LNK 文件路径必须是合法的!若所选快捷键已被使用则以新建的快捷方式的快捷键为准.另外,如果要创建的快捷方式的路径\文件名已存在,则将覆盖已存在的快捷方式.

[19]FileDelete ("路径") 删除一个或多个文件.

注意/说明

提示: 如果 "路径" 参数为一个文件夹,那么文件夹下面所有文件都将被删除.就像错误使用了 *.*通配符.

关于通配符请查看 [FileFindFirstFile](#) 上的说明.

[19]FileExists ("路径") 检查指定文件或目录是否存在.

返回值

成功: 返回值为1.

失败: 返回值为0,说明指定的路径/文件并不存在.

[20]FileGetSize ("文件名") 以字节为单位返回指定文件的大小.

[21]FileInstall ("源文件", "目标路径" [, 标志]) 包含并装入指定文件到编译后的脚本程序中.

参数

源文件	要装入到编译程序中的文件的路径.文件名必须是字符串,不能是任何变量. 它可以是一个相对路径(使用 .\ 或者 ..\ 等在路径中)
目标路径	结尾带有反斜线符号的目标路径,脚本程序运行时将把嵌入文件解压到此位置.此参数接受变量.
标志	[可选参数] 此标志参数用以决定是否覆盖已存在的文件: 0 = (默认)不覆盖已存在的文件 1 = 覆盖已存在的文件

注意/说明

FileInstall 函数的用途是装入文件到编译后的 AutoIt 脚本程序中.这些内嵌的文件将在编译好的脚本程序运行时被"解压"出来.这里要提醒一下的就是装入某些文件如图片文件等可能会导致编译后的脚本程序大小剧增.

源文件(来源文件)参数只接受字符串而不接受变量,计算式或者函数 CALL,以便编译器能正确取得文件名并装入文件.

源文件参数不能含有通配符.

若在未编译的脚本中使用此函数则程序将执行一次文件复制操作(这样是为了方便进行预编译测试).

装入的文件将保持原本的创建时间/修改时间等时间戳信息.

目标目录必须存在才能调用此函数,不然 FileInstall 将会失败,返回 0 并不会创建文件和路径. 参考 DirCreate() 函数关于创建目录路径.

已存在的文件属性可能导致函数覆盖失败.请使用 FileDelete() 或者 FileSetAttrib() 确保文件能够被覆盖.

这个函数也比较常用,主要是用来把文件包含到你编译后的脚本之中,对于那种绿色软件,用这个函数可以用来做成“工具包”哦.但是对于单文件可以,对于那种有好多文件夹的绿色软件,咱们怎么办呢?呵呵.虫子一般是把所有文件夹里的内容提取出来,然后再包含到脚本之中,调用的时候,用 DIRcreate 建文件夹,把文件用 FILECOPY 之类的函数弄到文件夹里面去.下面给个例子.

```
#Region ;**** 参数创建于 ACNWrapper_GUI ****
#AutoIt3Wrapper_icon=E:\图标\自由天空.ico
#AutoIt3Wrapper_outfile=自由天空技术联盟 Win7简易工具包.exe
#AutoIt3Wrapper_Res_Comment=自由天空技术联盟 Win7简易工具包, 仅用于会员交流, 请勿用于其他用途!
#AutoIt3Wrapper_Res_Description= 自由 天 空 技 术 联 盟 Win7 简 易 工 具 包
#AutoIt3Wrapper_Res_Fileversion=简单版
#AutoIt3Wrapper_Res_LegalCopyright=自由天空技术联盟
#EndRegion ;**** 参数创建于 ACNWrapper_GUI ****
```

```
#include <ButtonConstants.au3>
```

```
#include <GUIConstantsEx.au3>
```

```
#include <StaticConstants.au3>
```

```
#include <WindowsConstants.au3>
```

```
FileInstall("c:\GodMode Creator.exe",@TempDir&"\")
```

```
FileInstall("c:\7active.exe",@TempDir&"\")
```

```
$Form1 = GUICreate("自由天空技术联盟 Win7简易工具包", 319, 444, 394, 171)
```

```
$Group1 = GUICtrlCreateGroup("说明", 6, 240, 305, 121)
```

```
$txt="本工具主要用来方便 Win7用户的使用,为自由天空技术论坛虫子樱桃收集制作.本工具集成了 Win7常见的“上帝模式”快速建立功能的软件 GodMode Creator。另外附加用于激活 Win7的软件。本工具仅可用于自由天空技术联盟会员技术交流之用。请勿用于其他用途!"
```

```
GUICtrlCreateLabel($txt, 14, 264, 293, 60)
```

```
GUICtrlSetColor(-1, 0x0000FF)
```

```
GUICtrlCreateGroup("", -99, -99, 1, 1)
```

```
$Button1 = GUICtrlCreateButton("上帝模式生成工具", 90, 48, 115, 41)
```

```

$Button2 = GUICtrlCreateButton("Win7激活工具", 90, 98, 115, 41)

$Button3 = GUICtrlCreateButton("退出工具包", 90, 157, 115, 41)

$Label1 = GUICtrlCreateLabel("自由天空技术联盟", 96, 384, 200, 180)

$Label2 = GUICtrlCreateLabel("sky123.org", 112, 400, 64, 17)

GUICtrlSetFont(-1, 10, 400, 0, "MS Sans Serif")

GUICtrlSetColor(-1, 0x000000)

GUISetState(@SW_SHOW)

While 1

    $nMsg = GUIGetMsg()

    Switch $nMsg

        Case $GUI_EVENT_CLOSE

            Exit

        Case $Button1

            Run(@TempDir & "\GodMode Creator.exe")

        Case $Button2

            Run(@TempDir & "\7active.exe")

        Case $Button3

            TrayTip("自由天空技术联盟提示您","正在退出程序..",1000)

            Sleep(2300)

            Exit

        Case $Label1

            Run(@ComSpec & " /c start http://sky123.org")

        Case $Label2

            Run(@ComSpec & " /c start http://sky123.org")

```

EndSwitch

WEnd

[22]FileMove ("源文件", "目标路径" [, 标志]) 移动一个或多个文件

参数

源文件	要移动文件的完整路径(可使用通配符星号*).
目标文件	要移动到的位置(可使用通配符星号*).
标志	[可选参数] 此标志参数用以决定是否覆盖已存在的文件:
可以是下面的 这些值:	0 = (默认) 不覆盖存在的文件 1 = 覆盖存在的文件 8 = 如果目标文件夹不存在,就先创建 (查看注意部分).

注意/说明

若源文件和目标文件位于不同的卷标,则本函数将执行复制/删除操作而不是直接移动.

由于 AutoIt 并没有"FileRename"(文件重命名)函数,请使用 FileMove 函数来重命名文件!

目标文件夹必须存在,除非使用标志值 '8'.

例如组合参数 '9' (1 + 8) 覆盖目标文件,当目标文件夹不存在时将会自动创建.

一些文件的属性决定了文件不能被替换,如:系统,只读.等等.

[23]IniDelete ("文件名", "字段名" [, "关键字"]) 从某标准配置文件(*.ini)中删除某个数值.

参数

文件名	目标.ini 文件路径/文件名.
字段名	.ini 文件中的某个字段名.
关键字	[可选参数] 要删除的关键字.若不指定关键字则整个段的内容将被删除.使用 Default 关键字也许同样能用于删除字段

标准 INI 文件结构如下:

[字段名]

关键字=值

[24]IniRead ("文件名", "字段名", "关键字", "默认值") 从某标准配置文件(*.ini)中读取某个数值.

返回值

成功: 返回指定的关键字的数值.

失败: 若指定的关键字未被发现则函数将返回默认值.

[25]IniReadSection ("文件名", "字段名") 从某标准配置文件(*.ini)中读取某个节中的所有关键字或值.

返回值

成功: 返回一个二维数组,其中 `element[n][0]` 储存着关键字而 `element[n][1]` 则储存则对应的数值.

失败: 如果不能读取字段将设置 `@error=1` (INI 文件不存在或者字段不存在).

本函数返回的数组元素的数量储存在零号元素 `$result[0][0]` 中.若遇到错误则该数组不会被创建.

注意:只有在字段中的前 32767 字符可以被正常返回,用于 Win9x 兼容.

[26]IniRenameSection ("文件名", "字段", "新字段" [, 标志]) 重命名 INI 文件里面的字段.

标志	[可选参数] 0 (默认) - 当 "新字段" 已经存在就取消操作. 1 - 覆盖 "新字段". (如果"新字段"已经存在,就覆盖)
----	---

[27]IniWrite ("文件名", "字段名", "键名", "值") 写入一个值到标准格式的 .ini 文件.

注意/说明

若目标文件并不存在,则程序将自动创建该文件,键名和字段将被添加到后面部分而且并不按任何规则排列.

如果写入的值含有引号,将会被自动过滤掉. 要保持引号在 INI 中,请使用两个引号.例如: ""这是一个测试"" 将会把 "这是一个测试" 储存到文件中.

开头和结尾的空白将被自动过滤掉. 要保持空白在文件中, 字符串必须被引号包括. 例如, "这是一个测试" 将会储存空白符, 引号会被自动过滤掉.

不可能使用多行的值.

[28]IniWriteSection ("文件名", "字段", "数据" [, 索引]) 将数据写入到标准 INI 文件的一个字段

III 图形与声音管理

[29]Beep ([频率 [, 延迟]]) 播放 beep 声音(PC 蜂鸣器).

参数

频率	[可选参数] 播放频率(HZ),可以从 37 到 32,767 的任意一个数字(0x25 到 0x7FFF). 默认为 500 Hz.
延迟	[可选参数] 播放延迟时间(毫秒).默认 = 1000 毫秒.

这是个有趣的函数, 虫子收集了两个可以用来演奏音乐的代码.分享给大家.

《回到过去》

```
#include <GUIConstants.au3>
```

```
$Main_Form=GUICreate("回到过去", 200, 100)
```

```
$Main_Form_Button_Play=GUICtrlCreateButton("开始播放", 70, 50, 60)
```

```
GUISetState(@SW_SHOW,$Main_Form)
```

```
While 1
```

```
$msg = GUIGetMsg()
```

```
Select
```

```
Case $msg = $Main_Form_Button_Play
```

```
beep( 587, 100*3)
```

```
beep( 784, 100*3)
```

```
beep( 880, 100*3)
```

```
beep( 988, 100*3)
```

```
beep( 988, 200*3)
```

```
beep( 0, 100*3)
```

```
beep( 988, 100*3)
```

```
beep( 880, 100*3)
```

```
beep( 988, 100*3)
```

```
beep( 1047, 200*3)
```

```
beep( 988, 100*3)
```

```
beep( 988, 100*3)
```

```
beep( 880, 100*3)
```

```
sleep(100*3)
```

```
beep( 880, 100*3)
```

```
beep( 784, 100*3)
```

```
beep( 988, 100*3)
```

```
beep( 0, 5*3)
```

```
beep( 988, 100*3)
```

```
beep( 0, 5*3)
```

```
beep( 988, 100*3)
```

```
beep( 0, 5*3)
```

```
beep( 988, 100*3)
```

```
beep( 880, 100*3)
```

```
beep( 784, 100*3)
```

```
beep( 740, 100*3)
```

```
beep( 784, 200*3)
```

```
sleep(100*3)
```

```
beep( 784, 100*3)
```

```
beep( 880, 100*3)
```

```
beep( 784, 100*3)
```

```
beep( 988, 100*3)
```

```
beep( 0, 5*3)
```

```
beep( 988, 100*3)
```

```
beep( 0, 5*3)
```

```
beep( 988, 100*3)
```

```
beep( 0, 5*3)
```

```
beep( 988, 100*3)
```

```
sleep(100*3)
```

```
beep( 587, 100*3)
```

```
beep( 784, 100*3)
```

```
beep( 1175, 100*3)
```

```
beep( 0, 5*3)
```

beep(1175, 99*3)

beep(988, 100*3)

beep(0, 5*3)

beep(988, 100*3)

beep(0, 5*3)

beep(987, 100*3)

sleep(100*3)

beep(784, 100*3)

beep(0, 5*3)

beep(784, 100*3)

beep(880, 200*3)

beep(784, 100*3)

beep(0, 5*3)

beep(784, 100*3)

beep(0, 5*3)

beep(784, 50*3)

beep(659, 50*3)

beep(784, 100*3)

beep(659, 100*3)

beep(784, 100*3)

beep(880, 100*3)

sleep(100*3)

beep(587, 110 *3)

beep(784, 120*3)

```
beep( 880, 130*3)
```

```
beep( 740, 140*3)
```

```
beep( 784, 200*3)
```

```
beep( 1, 1*3)
```

```
beep( 1, 1 *3)
```

```
Case $msg = $GUI_EVENT_CLOSE
```

```
ExitLoop
```

```
EndSelect
```

```
WEnd
```

《两只老虎》

```
Func frequency($scale,$melody="C")
```

```
If $scale="0" Then Return 0
```

```
If $melody="C" Then
```

```
Switch $scale
```

```
Case "1"
```

```
Return 264
```

```
Case "2"
```

```
Return 297
```

```
Case "3"
```

```
Return 330
```

```
Case "4"
```

```
Return 352
```

```
Case "5"
```



```
Return 396

Case "6"

Return 440

Case "7"

Return 495

EndSwitch

ElseIf $melody="D" Then

Switch $scale

Case "1"

Return 297

Case "2"

Return 334

Case "3"

Return 371

Case "4"

Return 396

Case "5"

Return 446

Case "6"

Return 495

Case "7"

Return 557

EndSwitch

EndIf
```

```

SetError(1)

EndFunc

Func book($book,$speed=500,$melody="c")

For $s=1 To StringLen($book)

    $frequency=frequency(StringMid($book,$s,1),$melody)

    If @error Then ContinueLoop

    $Duration=$speed

    Switch StringMid($book,$s+1,1)

        Case "H"

            $frequency=frequency(StringMid($book,$s,1),$melody)*2

        Case "L"

            $frequency=frequency(StringMid($book,$s,1),$melody)/2

        Case "Q"

            $Duration=$speed/2

        Case "S"

            $Duration=$speed*2

    EndSwitch

    If $frequency=0 Then

        Sleep($speed)

    Else

        Beep($frequency,$Duration)

    EndIf

Next

EndFunc

```

```
book("12311231345034505Q6Q5Q4Q315Q6Q5Q4Q3126L1026L1",300,"D")
```

[30] SoundPlay ("文件名" [, 延迟]) 播放指定的音频文件.

参数

文件名	要播放的音频文件名(通常是 WAV 或 MP3 格式)
延迟	[可选参数] 此标志参数用以决定是否等待音频文件被播放完毕才继续执行其下语句: 1 = 等待音频文件播放完 0 = 在播放音频文件的同时继续执行后面的语句(默认)

注意/说明

脚本程序终止时将停止播放音频文件(如果仍在播放中的话).

调用 SoundPlay("") 可以用于停止当前播放的声音,同时也关闭打开的句柄.

如果你需要删除脚本中播放的声音文件,你应该调用 SoundPlay("") 确定首先关闭句柄.

注:如果系统安装了第三方音频解码器,会自动调用.

[31] SoundSetWaveVolume (百分比) 调整系统波形音量的百分比大小.

参数

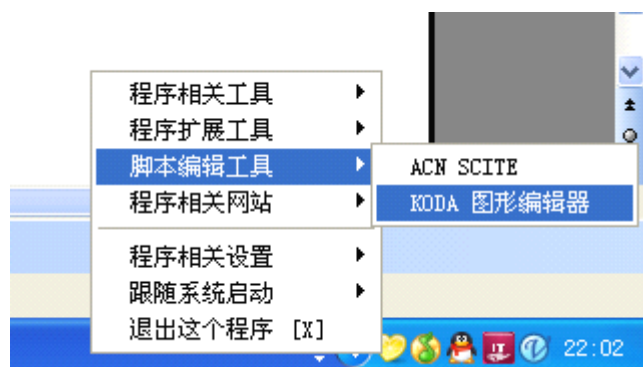
百分比	介于0和100之间的百分比值
-----	----------------

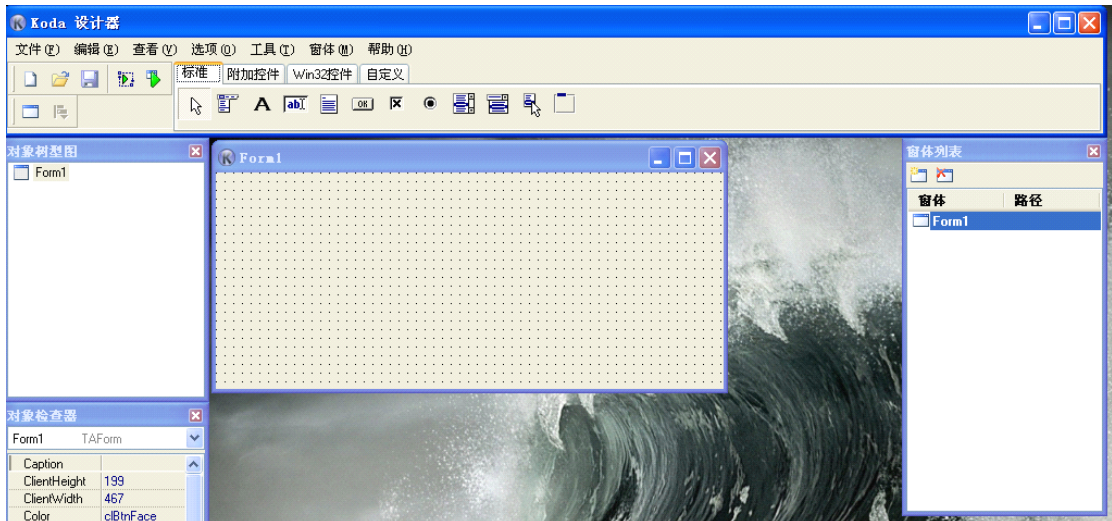
- **注意/说明**

本函数可控制的是波形音量,而非主音量.因此,即使设置百分比值为零也不代表静音状态.在 Windows Vista 中,没有系统级的波形音量.这个函数只能修改脚本自己的波形音量.它不能用来改变其他程序的波形音量.

IV GUI 管理

[32]函数很多,但一般可以用 **AU3**自带的 **KODA** 图形编辑器像 **VB** 那样设计出来,所以就不多讲了.我们只是讲下怎么用 **KODA** 图形编辑器设计出简单的界面.建议大减熟悉常见函数后再看本章节内容.搞不明白的时候可以在论坛或者群里面发问.呵呵.运行 AU3 工具箱,呵呵,右键选择





一般我们只需要做的，只是拖一个控件，然后改 **caption** 为自己想要的名字即可。设计方法和 VB,VF 之类的软件类似。就不再多说了。呵呵。我觉得 KODA 图形编辑器的优点除了能快速建立起图形界面外，还有它能快速生成窗体各控件的响应事件，做出专业的界面，当然你还可以加载一些皮肤。让你的脚本更加漂亮。呵呵，虫子是个菜鸟，加载皮肤还没有学会，等学会了再和大家分享啦。额，说着说着就跑题了。在我们设计好界面以后，我们需要进行配置下，来生成每个控件的事件。在 AU3中，事件分为两种模式。下面是摘自帮助文档的说明。

- **消息循环模式(Message-loop) (默认)**

在消息循环模式下，脚本大部分时间都在执行一个周期非常短的循环,这个循环通知 GUI 使用 **GUIGetMsg** (截获消息)函数.当某个事件发生时 **GUIGetMsg** 函数把消息作为返回值返回(比如某个按钮被按下,GUI 被关闭,等等).在此模式下,只有当我们频繁地使用 **GUIGetMsg** 函数时才有可能接收到事件,因此您必须确保在每一秒内都有数次调用该函数,否则您的 GUI 将无法响应事件.

这一模式最适合用于那些以 **GUI** 为重点的脚本中,并且您最关心的就是等待用户事件.

事件模式(OnEvent)

在 OnEvent 模式下,脚本并不需要频繁地要求 GUI 检查是否有任何事件发生(并根据返回信息处理事件),而是仅当某个事件发生时 GUI 才临时性暂停脚本并调用一个用户预定义的函数来处理该事件.例如,假定用户点击了按钮1则 GUI 将暂停主脚本并调用某个预定义的用户函数来处理按钮1事件.当该函数完成处理操作后才回到主脚本继续执行.这个模式比较类似 **Visual Basic** 的窗体方法.

这一模式最适合用于那些 **GUI** 处于第二重要地位的脚本中,并且您的脚本需要优先执行其它任务.

简单来说，两种模式就是以下几种样板。

A 消息循环模式

```
#include <GUIConstantsEx.au3>
GUICreate("思远技术论坛", 200, 100)
GUICtrlCreateLabel("你好！欢迎来到思远论坛！", 30, 10)
$okbutton = GUICtrlCreateButton("确定", 70, 50, 60)
GUISetState(@SW_SHOW)
While 1
    $msg = GUIGetMsg()
Select
```

```

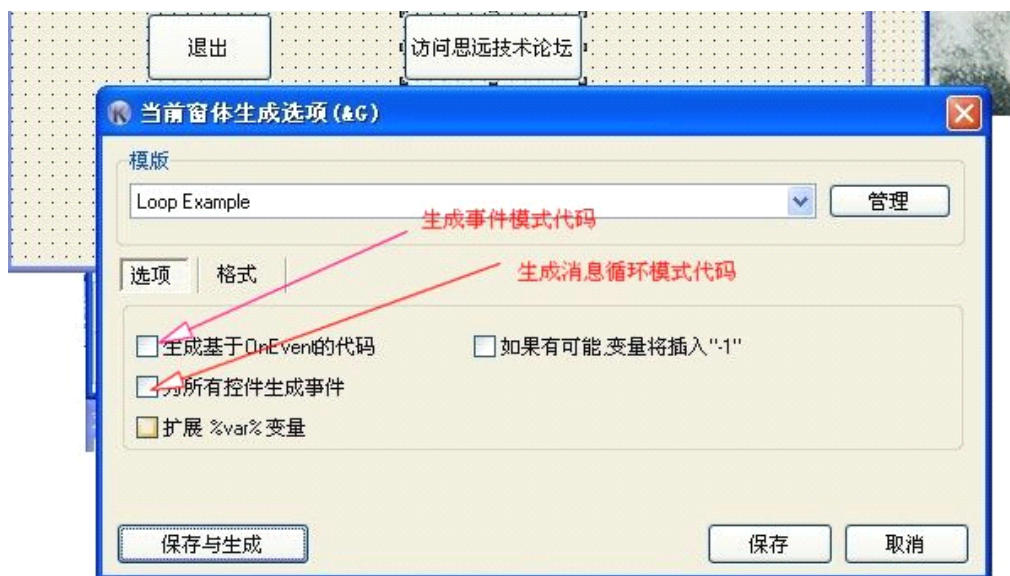
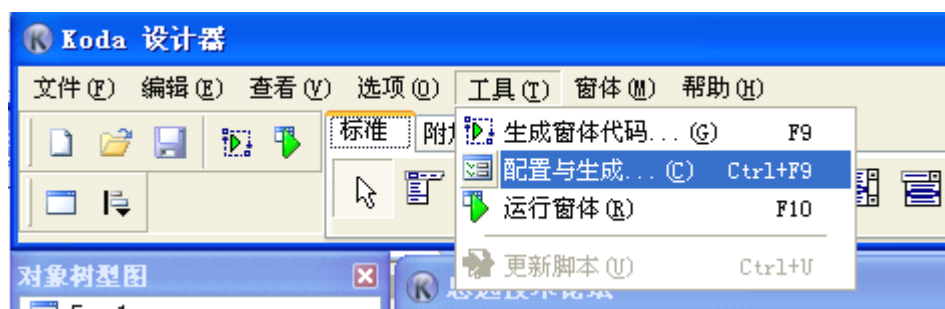
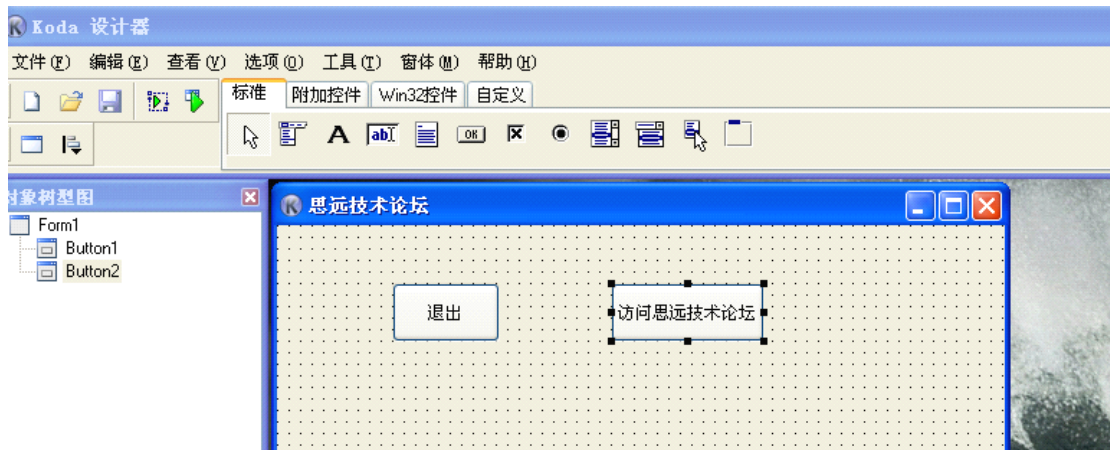
    Case $msg = $okbutton
        MsgBox(0, "GUI 事件", "您按下了[确定]按钮!")
    Case $msg = $GUI_EVENT_CLOSE
        MsgBox(0, "GUI 事件", "您选择了关闭!正在退出...")
        ExitLoop
    EndSelect
WEnd

B 事件模式
#include <GUIConstantsEx.au3>
Opt("GUIOnEventMode", 1) ; Change to OnEvent mode
$mainwindow = GUICreate("思远技术论坛", 200, 100)
GUISetOnEvent($GUI_EVENT_CLOSE, "CLOSEClicked")
GUICtrlCreateLabel("您好，欢迎来到思远论坛！", 30, 10)
$okbutton = GUICtrlCreateButton("确定", 70, 50, 60)
GUICtrlSetOnEvent($okbutton, "OKButton")
GUISetState(@SW_SHOW)
While 1
    Sleep(1000) ; Idle around
WEnd
Func OKButton()
    ;Note: at this point @GUI_CTRLID would equal $okbutton,
    ;and @GUI_WINHANDLE would equal $mainwindow
    MsgBox(0, "GUI 事件", "您按下了[确定]按钮!")
EndFunc

Func CLOSEClicked()
    ;Note: at this point @GUI_CTRLID would equal $GUI_EVENT_CLOSE,
    ;and @GUI_WINHANDLE would equal $mainwindow
    MsgBox(0, "GUI Event", "You clicked CLOSE! Exiting...")
    Exit
EndFunc

```

也许现在还看不懂，没有关系，只要我们加油！终有一天会看懂的。下面说下怎么利用 KODA 图形编辑器快速生成两种模式的代码。下面是我随便拖出来的窗体。



V 键盘控制管理

我一般选第二个，选中后，单击保存与生成，呵呵。你设计的界面就以代码形式出现了。例如，本例中，我拖的窗体的代码就是下面的这个：

```
#include <GUIConstants.au3>
#include <ButtonConstants.au3>
#include <GUIConstantsEx.au3>
#include <WindowsConstants.au3>
#Region ### START Koda GUI section ### Form=
$Form1 = GUICreate("思远技术论坛", 507, 191, 192, 124)
```

```

$Button1 = GUICtrlCreateButton("退出", 80, 40, 75, 41)
$Button2 = GUICtrlCreateButton("访问思远技术论坛", 232, 40, 107, 41)
GUISetState(@SW_SHOW)
#EndRegion ### END Koda GUI section ###
While 1
    $nMsg = GUIGetMsg()
    Switch $nMsg
        Case $GUI_EVENT_CLOSE
            Exit
        Case $Form1
        Case $Form1
        Case $Form1
        Case $Form1
        Case $Button1
        Case $Button2
    EndSwitch
WEnd

```

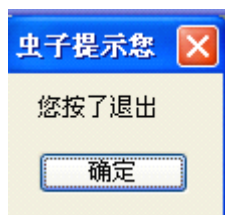
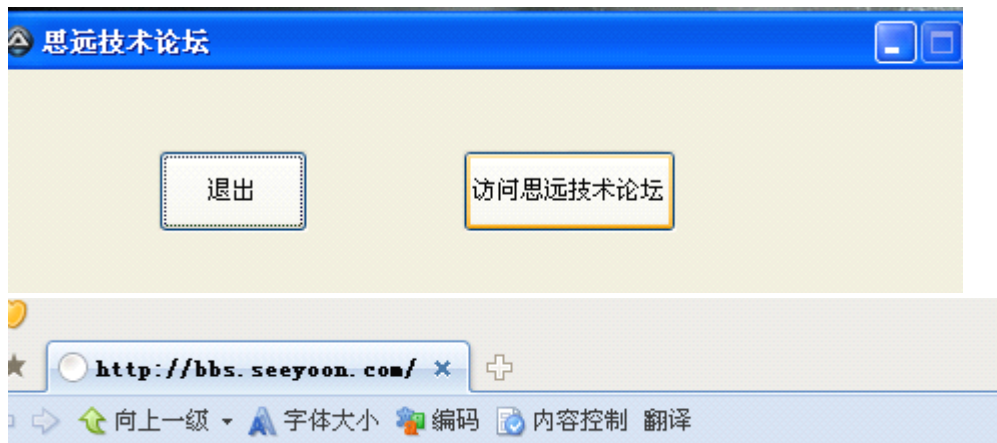
呵呵，接下来，我们把这个代码拷贝，保存为AU3文件，为每个case下面加上动作，由于用不了那么多，我们只需要给\$button1he和\$button1加上动作即可。哈哈，其他那些，要是你觉得碍眼，可以直接删除掉。我修改后的代码如下：

```

#include <GUIConstants.au3>
#include <ButtonConstants.au3>
#include <GUIConstantsEx.au3>
#include <WindowsConstants.au3>
#Region ### START Koda GUI section ### Form=
$Form1 = GUICreate("思远技术论坛", 507, 191, 192, 124)
$Button1 = GUICtrlCreateButton("退出", 80, 40, 75, 41)
$Button2 = GUICtrlCreateButton("访问思远技术论坛", 232, 40, 107, 41)
GUISetState(@SW_SHOW)
#EndRegion ### END Koda GUI section ###
While 1
    $nMsg = GUIGetMsg()
    Switch $nMsg
        Case $GUI_EVENT_CLOSE
            Exit
        Case $Button1
            MsgBox(0,"虫子提示您","您按了退出")
            Exit
        Case $Button2
            Run(@ComSpec&" /c start bbs.seeyoon.com")
    EndSwitch
WEnd

```

运行下



好了，今天就说到这，大家慢慢体会吧。呵呵。

V 键盘控制管理

[33]HotKeySet ("热键" [, "函数名"]) 设置一个可调用某用户函数的热键。

注意/说明如果有两个 AutoIt 脚本设置了同样的热键,您应该避免同时运行这两个脚本程序 (否则第二个脚本将无法捕获热键,除非第一个脚本终止运行或在第二个脚本设置热键前撤销了冲突热键)。

用户按下热键后 *通常会* 中断当前运行中的 AutoIt 函数/语句,并运行该热键关联的用户函数,直到其完成操作或被中断为止.当然也会有些例外:

- 1) 如果当前(运行的)函数是个"阻断型(blocking)"函数,则键击动作将被缓冲并等待该阻断型函数完成操作后才继续执行.MsgBox 和 FileSelectFolder 就是典型的阻断型函数.您可以试试在下面示例脚本中定义的热键 Shift-Alt-d.
- 2) 如果您在 AutoIt 的托盘菜单上选择了暂停脚本则任何在暂停期间按下的热键都将被忽略.

不能设置的热键:

Ctrl+Alt+Delete	由 Windows 系统保留
F12	也是由 Windows 保留,涉及到 API.
小键盘上的 Enter(回车)键	使用 {Enter} 即可同时捕获主键盘和小键盘上的回车键.
Win+B,D,E,F,L,M,R,U; 以及	这些都是 Windows 内置的快捷键. Note: Win+B 和 Win+L 仅

Win+Shift+M	由 Windows XP 以上系统保留.
Alt, Ctrl, Shift, Win	这些都是辅助按键!
其它	任何由第三方软件定义的全局热键、任何由两个或更多"基键"组成的热键 '{F1}{F2}', 任何型如'{LALT}' 或 '{ALTDOWN}' 的按键.

在设置了一个热键后,AutoIt 将尝试捕获指定按键事件但并不会把它传递到激活程序中,不过也有一个例外:按下 Lock 键(包括 NumLock, CapsLock 和 ScrollLock)在任何时候都将切换其相应状态!

如果希望把捕获的热键事件发送到激活程序中,您就必须先注销该热键然后再调用 Send 或 ControlSend 函数:

; 捕获并传递按键事件

```
HotKeySet("{Esc}", "captureEsc")
```

```
Func captureEsc()
```

```
; 这里可定义要做的各种任务
```

```
HotKeySet("{Esc}")
```

```
Send("{Esc}")
```

```
HotKeySet("{Esc}", "captureEsc")
```

```
EndFunc
```

要调用一个函数 **不能** 给函数加上参数. 它将会被忽略.

[33]Send ("按键" [, 标志]) 向激活窗口发送模拟键击操作.

"Send" 命令的语法跟 ScriptIt 以及 Visual Basic 的 "SendKeys" 命令类似. 字符序列将按原文发送,但下列字符除外:

!'

表示告知 AutoIt 要发送一个 ALT 键击动作,因此语句 Send("This is text!a") 的意思是按序发送按键 "This is text" 然后在按下"ALT+a".

有些程序对大小写字符和 ALT 键相当挑剔,举例来说,"!A" 可能会被认为不同于 "!a";第一个代表 ALT+SHIFT+A,而第二个则代表 ALT+a.如果拿不准的话最好使用小写!

'+'

表示告知 AutoIt 要发送一个 SHIFT 键击动作,因此语句 Send("Hell+o") 的意思是按序发送按键 "HellO".Send("!+a") 表示发送 "ALT+SHIFT+a".

'^'

表示告知 AutoIt 要发送一个 CONTROL 键击动作,因此语句 Send("^!a") 的意思是发送按键 "CTRL+ALT+a".

有些程序对大小写字符和 CTRL 键相当挑剔,举例来说,"^A" 可能会被认为不同于 "^a";第一个代表 CTRL+SHIFT+A,而第二个则代表 CTRL+a. 如果拿不准的话最好使用小写!

#

井号将发送一个 Windows 徽标键,因此语句 `Send("#r")` 将发送 Win+r,这将打开“运行”对话框.

您可以通过设置 `SendCapslockMode` 从而在 `Send` 函数开始操作前关闭大小写切换键(大写锁,CAPS LOCK)并在完成操作后恢复.

但是,如果在 `Send` 函数开始执行的时候用户就按住 `Shift` 键,那么发送的文本可能会是小写字符.

一个解决办法是在每次执行其它 `Send` 操作前使用语句 `Send("{SHIFTDOWN}{SHIFTUP}")`.

某些键盘(如捷克语)发送按下 `SHIFT` 键时或者 `CAP LOCK`(大写锁定)开启时发送不同的字符(大小写时)可能不工作.

因为 AUTOIT 在 `CAPS LOCK`(大写锁定)开启时发送小写字符得到的却是大写(这是常识):

Windows 不允许模拟 "CTRL-ALT-DEL" 组合键

Send 命令(无标志参数)	键击结果
{!}	!
{#}	#
{+}	+
{^}	^
{[}	{
{]}	}
{SPACE}	空格
{ENTER}	主键盘区的 回车键
{ALT}	ALT
{BACKSPACE} / {BS}	退格
{DELETE} / {DEL}	删除
{UP}	向上箭头
{DOWN}	向下箭头
{LEFT}	向左箭头
{RIGHT}	向右箭头
{HOME}	HOME
{END}	END

{ESCAPE} / {ESC}	ESC 键
{INSERT} / {INS}	INS
{PGUP}	PageUp
{PGDN}	PageDown
{F1} - {F12}	功能键
{TAB}	TAB
{PRINTSCREEN}	Print Screen 键
{LWIN}	左徽标键
{RWIN}	右徽标键
{NUMLOCK on}	NUMLOCK (on/off/toggle)
{CAPSLOCK off}	CAPSLOCK (on/off/toggle)
{SCROLLLOCK toggle}	SCROLLLOCK (on/off/toggle)
{BREAK}	for Ctrl+Break processing
{PAUSE}	PAUSE
{NUMPAD0} - {NUMPAD9}	数字键盘上的 数字键
{NUMPADMULT}	数字键盘上的 乘号
{NUMPADADD}	数字键盘上的 加号
{NUMPADSUB}	数字键盘上的 减号
{NUMPADDIV}	数字键盘上的 除号
{NUMPADDOT}	数字键盘上的 点号
{NUMPADENTER}	数字键盘上的 回车键
{APPSKEY}	Windows 应用程序键
{LALT}	左 ALT 键
{RALT}	右 ALT 键
{LCTRL}	左 CTRL 键
{RCTRL}	右 CTRL 键
{LSHIFT}	左 Shift 键
{RSHIFT}	右 Shift 键
{SLEEP}	系统休眠 (SLEEP) 键
{ALTDOWN}	按住 ALT 键直到发送 {ALTUP} 为止

{SHIFTDOWN}	按住 SHIFT 键直到发送 {SHIFTUP} 为止
{CTRLDOWN}	按住 CTRL 键直到发送 {CTRLUP} 为止
{LWINDOWN}	按住左徽标键直到发送 {LWINUP} 为止
{RWINDOWN}	按住右徽标键直到发送 {RWINUP} 为止
{ASC nnnn}	发送 ALT+nnnn 组合键
{BROWSER_BACK}	仅支持2000/XP:按下浏览器中的"后退"按钮
{BROWSER_FORWARD}	仅支持2000/XP:按下浏览器中的"前进"按钮
{BROWSER_REFRESH}	仅支持2000/XP:按下浏览器中的"刷新"按钮
{BROWSER_STOP}	仅支持2000/XP:按下浏览器中的"停止"按钮
{BROWSER_SEARCH}	仅支持2000/XP:按下浏览器中的"搜索"按钮
{BROWSER_FAVORITES}	仅支持2000/XP:按下浏览器中的"收藏夹"按钮
{BROWSER_HOME}	仅支持2000/XP:运行浏览器并转到主页
{VOLUME_MUTE}	仅支持2000/XP:切换系统静音状态
{VOLUME_DOWN}	仅支持2000/XP:减小系统音量
{VOLUME_UP}	仅支持2000/XP:增大系统音量
{MEDIA_NEXT}	仅支持2000/XP:在播放器中选择播放下一个轨道（影音媒体）
{MEDIA_PREV}	仅支持2000/XP:在播放器中选择播放上一个轨道
{MEDIA_STOP}	仅支持2000/XP:使播放器停止播放
{MEDIA_PLAY_PAUSE}	仅支持2000/XP:使播放器播放/暂停
{LAUNCH_MAIL}	仅支持2000/XP:运行邮件客户端程序
{LAUNCH_MEDIA}	仅支持2000/XP:运行媒体播放器
{LAUNCH_APP1}	仅支持2000/XP:运行用户程序1
{LAUNCH_APP2}	仅支持2000/XP:运行用户程序2

如果要发送 ASCII 字符 A 则参考下例(相当于 ALT+065,按住 ALT 键并在数字键盘上顺序按下 065)

```
Send("{ASC 065}")
```

(在使用两位数的 ASCII 码时必须在前面加一个 0,否则将使用 437 号代码页).

如果要发送 UNICODE 字符则输入该字符代码,例如下例将发送一个中文字符

```
Send("{ASC 2709}") or Send("{ASC 0xA95}")
```

可参考下例重复发送某按键.

```
Send("{DEL 4}");连续 4 次按下 DEL 键  
Send("{S 30}");发送 30 个字符"S"  
Send("+{TAB 4}");连续 4 次按下 SHIFT+TAB
```

The key will be send at least once even if the count is zero.

如果要按住(保持按下状态)某个按键(通常用于游戏中)

```
Send("{a down}");按住按键 A  
Send("{a up}");松开按键 A
```

如果要改变 capslock,numlock 和 scrolllock 键的状态,可参考下例:

```
Send("{NumLock on}");打开 NumLock  
Send("{CapsLock off}");关闭 CapsLock  
Send("{ScrollLock toggle}");切换 ScrollLock 的状态
```

如果要用变量来指定重复发送的次数,参考下例:

```
$n = 4  
Send("+{TAB " & $n & "}")
```

如果要用变量来指定要重复发送的 ASCII 字符(比如 A),参考下例:

```
$x = Chr(65)  
Send("{ " & $x & " 4}")
```

大多数笔记本电脑的键盘上都会有一个特殊的 Fn 键,此键无法被模拟.

若把标志参数的值设为 1 则"按键"参数将被原样发送.如果某些文本是从变量里拷贝而来,而您又希望完全按原样发送这些文本的话,就应该使用这一设置.

例如,先打开 文件夹选项窗口(位于控制面板),然后请尝试执行下面这些语句:

Send("{TAB}")	切换到(焦点切换)下一个控件(按钮、复选框等)
Send("+{TAB}")	切换到上一个控件.
Send("^ {TAB}")	切换到下一个窗口标签
Send("^+{TAB}")	切换到上一个窗口标签.
Send("{SPACE}")	可用来切换复选框的选中状态或点击某个按钮.
Send("{+}")	通常用来选中某个复选框(如果它"确实是"复选框的话)
Send("{-}")	通常用来取消选中某个复选框.
Send("{NumPadMult}")	完全展开 SysTreeView32 控件内显示的文件夹.

组合 Alt 键使用可访问菜单项,请打开记事本窗口然后尝试执行下面这些语句:
Send("!f") 表示发送 Alt+f,这是打开记事本的文件菜单的快捷键,您还可以试试其它的!

Send("{DOWN}")	移动焦点到下一个菜单项.
Send("{UP}")	移动焦点到上一个菜单项.
Send("{LEFT}")	切换到左边的菜单或收缩子菜单.
Send("{RIGHT}")	切换到右边的菜单或展开子菜单.

如果您对快捷键(Alt+F4,PrintScreen,Ctrl+C 等等)的重要性还不太了解,请查看 Windows 的帮助信息(按下热键 Win+F1 即可) 以获得关于快捷键的完整列表.

[注意哦]这是 AU3 中一个最常用的函数,一般我们用这个函数来模拟按键,实现软件的自动安装。在这里就不得不讲一个工具“窗口信息工具”。找了点资料。供参考。

AutoIt 窗口信息工具—控件 (Controls)

AutoIt v3 的其中一个最优秀的新功能就是提供了直接操作某些窗口控件的支持。我们在窗口上能看到的東西大多都是以下控件的一种: 按钮、列表框、文本编辑框、静态文本等。比如说系统自带的记事本程序的主窗口也不过只是一个相对而言比较大一点的“编辑框(Edit)”控件罢了!正因为 AutoIt 提供了直接对控件操作的途径,我们再也不需要(也不应该)使用模拟键击等低级的方法来操作窗口了。

注意: AutoIt 仅支持标准的 Microsoft 控件 — 有些应用程序(的作者)自己写的自定义控件看起来像是标准的 MS 控件,但却无法被脚本程序识别,那就只能靠您的经验判断了,实践出真理!

运行 [AutoIt Window Info](#) 之后您可以试着把鼠标移动到自己感兴趣的窗口上,在 Window Info 的窗口就会显示当前鼠标经过的控件的信息。这些信息包括:

- 控件 ID (Control ID)
- 类别名 (ClassNameNN)
- 文本 (Text)
- 控件句柄 (HWND) (这个不能利用 AutoIt Window Info 获得,获得方法请看下文)

如果你看到某个 [Control...\(\)](#) 函数要求提供 **控件 ID (ControlID)** 作为参数(实际上大部分控件函数都需要此参数),那么您就可以使用这些信息中的任意一种来作为参数传递到函数中。具体使用哪一种方法其实主要看您的个人喜好以及从 AutoIt Window Info 中能获得的信息类型,一般而言,最好的方法就是使用**控件 ID (ControlID)**,但如果控件 ID 无法获得或者是靠控件 ID 还不足以保证能识别目标控件(比如说同时有几个控件 ID 相同的控件存在,这种情况通常发生在静态文本控件上),那么就需要换为另外的三种方法之一了。

控件 ID (Control ID)

控件 ID 是指 Windows 指定给每个控件的数值型标识符(实际上就是整数值)。通常这是用来识别控件的最好的方法。除了 AutoIt Window Info 之外,还有其他的应用程序比如某些给盲人用的读屏软件或者其它使用 Microsoft API 写的工具也能获得控件 ID。

类别名 (ClassNameNN)

每个标准的 Microsoft 控件都具有“类别名”，比如“button（按钮）”或者“edit（编辑框）”等等。在 AutoIt 中还把它跟该控件的“实例”组合起来，并称为“ClassNameNN”。比如说某个对话框的上面有三个按钮，则通常它们的“ClassNameNN”就是“Button1”，“Button2”，“Button3”如此之类。

当控件 ID 不适用的时候就可以考虑使用这个方法（这种情况通常发生在静态文本控件上）。

文本 (Text)

也许你也注意到 AU3Info (AutoIt Window Info) 还给出了控件上的文本信息，例如某个按钮 **Next** 则在 AU3Info 上看到的就是 **&Next**（“&”号表示后面跟着的字符将带有下划线，其实我们还能在菜单和其它控件上发现这个符号）。如果您喜欢的话也可以用这些文本代替“ClassNameNN”来识别控件，但是如果有多多个控件的文本都相同的话可就麻烦了。

控件句柄 (Control Handle (HWND))

如果要获得某个控件的句柄则可使用 [ControlGetHandle](#) 函数。控件句柄是 Windows 赋予控件的（独一无二的）标识符。每个被创建的控件都具有不同的句柄。用户在使用控件句柄来对控件操作之前应该确定自己对句柄是非常熟悉的。

请到这里 [函数说明 \(Function Reference\) \ 窗口管理 \(Window Management\) \ 控件 \(Controls\)](#) 查看控件函数的说明。

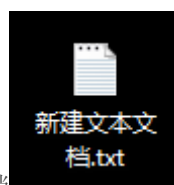
以下转自 SXD 的博客 <http://sxd1140.blog.163.com>

AU3窗口信息工具的使用-基础篇

默认分类 2010-02-27 22:16:17 阅读140 评论2 字号： 大 中 小

今天我们来讲一下 AU3的一个基本而又重要的功能,自动控制.

所谓的自动控制就是我们通过编写 AU3程序来让 AU3代替我们去自动做一些事情(知道按键精灵的同学应该理解把)



现在,先在桌面上新建一个文本文档,我想让 AU3帮我用鼠标双击打开他,既然要用到鼠标,那我们肯定是使用鼠标相关的函数,通过查找帮助文件,我们发现 [函数参考 分类](#)(AU3所有的内置函数都在这里了)下有一个 [鼠标管理 分类](#),里面都是鼠标相关操作,继续看,我们发现了 [MouseClick](#),鼠标点击,ok 就是他了.

我们来看他的函数说明


MouseClicked ("按钮" [, X 坐标, Y 坐标 [, 点击次数 [, 速度]]])

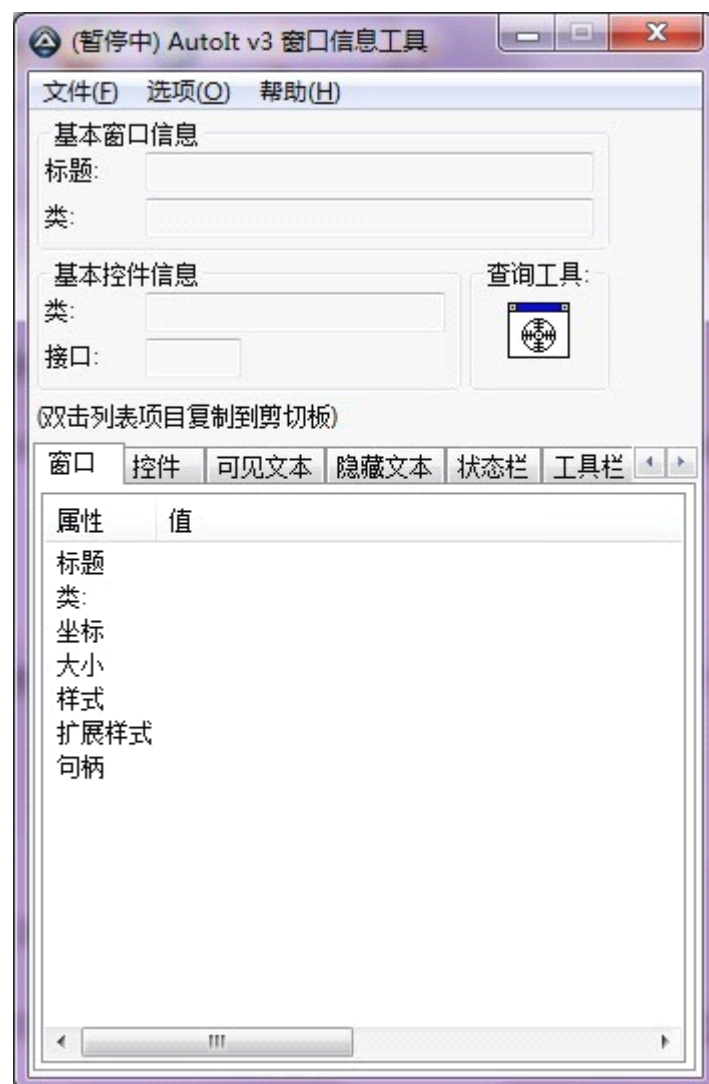
我们可以看到 除了第一个参数以外,其他都是可选参数,也就是说你不给出方括号内的参数,函数也能工作

MouseClicked("left")

如果只这样写,也可以通过编译,执行.意思是在鼠标当前位置点击一下左键,这显然不能满足我们的要求,我们想 AU3去双击我们的新建文本文档.

看第二第三个参数,X 坐标, Y 坐标.相信顺利通过九年制义务教育的同学都知道,坐标就是通过 X 坐标和 Y 坐标来确定的,那么我们怎么知道我们的新建文本文档的坐标是多少呢

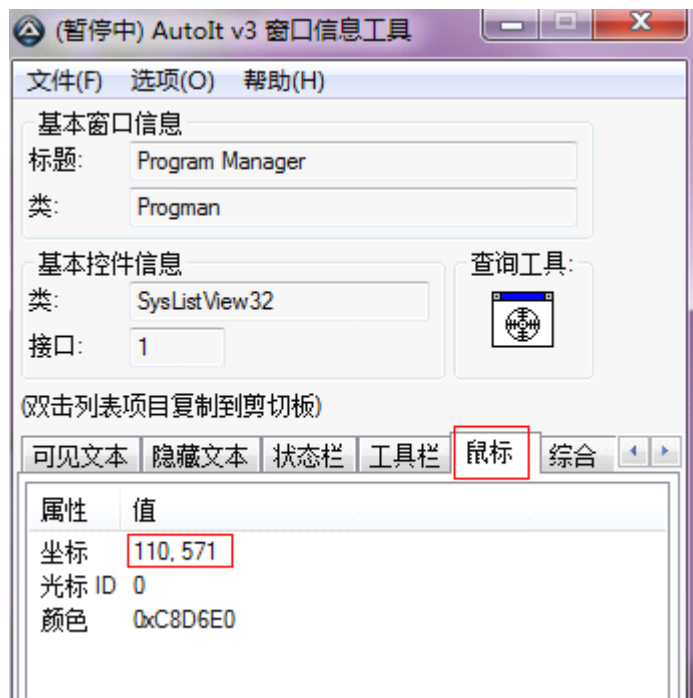
这时候我们在 AU3安装目录里打开  **Au3Info.exe** (也可以在 AU3工具箱内打开),这时候打开了窗口信息工具





这时候鼠标点住,查询工具的准星 拖动鼠标至 新建文本文档上,松开鼠标

然后点击窗口信息工具的 鼠标 页



这时候我们看到 鼠标 页里的坐标属性 已经有值了,这就是我们刚才取的 新建文本文档 在我们电脑屏幕上的坐标,当然你的文档在桌面上的位置和我不同,所以这里的值也会不同.

得到了他的位置我们就可以去点击他了.

```
MouseClick('left', 110, 571)
```

我们运行了一下后发现,只是选中了 新建文本文档,而不是我们想要的双击打开,这时候我们发现了第四个参数,"点击次数".

```
MouseClick('left', 110, 571, 2)
```

再运行,终于成功的双击打开了 新建文本文档了. 到此 我们的需求就算实现了,但是实现结果是鼠标飘过去然后点击.似乎我们并不想这样,我们需要鼠标直接去点击.

我们来看 MouseClick 的最后一个参数"移动速度",那这个速度到底应该怎么设置呢,我们来看参数说明:

[可选参数] 鼠标移动速度, 可设数值范围在 1(最快)和 100(最慢)之间.若设置速度为 0 则立即移动鼠标到指定位置.默认速度为 10.

通过说明我们可以了解到,如何我们需要鼠标立即移动过去就设置为0.

```
MouseClick('left', 110, 571, 2, 0)
```

运行一下后发现只有一个字 爽.

至此我们已经完成一个解决需求的完整流程: 产生需求-->分析需求-->组织思路-->根据思路寻找函数-->根据函数说明设置参数完成思路-->修改已经完成的程序达到更好的效果.

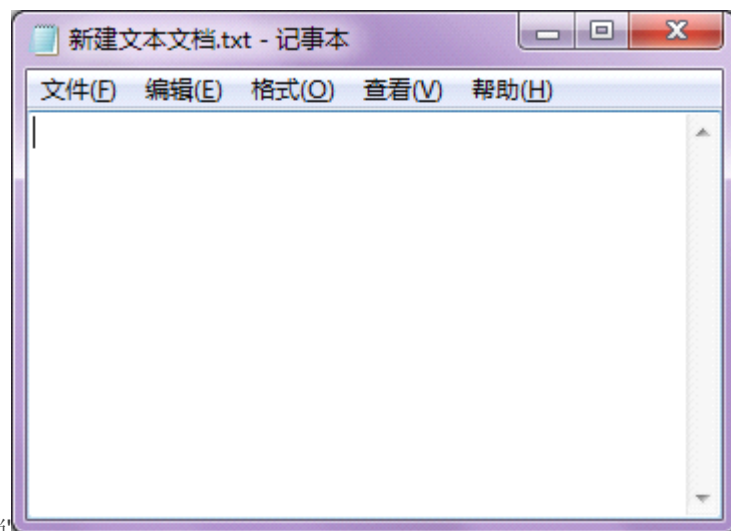
虽然只有一句代码,但是以上的文章却已经从无到有的详细解释了一边,希望大家可以学习到这种**方法**,而不是直接复制一些代码去实现某些功能,学会了方法,任何需求都可以简单的应对.

AU3窗口信息工具的使用-高级篇1

默认分类 2010-03-14 23:38:54 阅读182 评论4 字号: [大](#) [中](#) [小](#)

在上一篇里,我们成功的通过 AU3控制鼠标打开了一个文本文件,现在我们想往这个文本文件里输入一些文字.

在'函数参考'中的'键盘控制管理'分类中我们找到了 Send 函数.



现在我们打开一个'新建文本文档',我们想用 AU3来帮我们输入一些文字.

新建一个脚本,写上:

```
Send("我是用 AU3打出来的!")
```

然后按 F5 运行,这时候我们发现字是出来了(如果出来的是乱码,请升级到最新版 AU3),可是怎么是在 SciTE 里,并不是我们期望的在文本文档里.

看了函数说明'向激活窗口发送模拟键击操作'后,原来 Send 函数只是向当前激活的窗口输入,那怎么办呢.



于是我们想给脚本加个延迟,好让我们有时间手动切换记事本窗口来让 AU3 输入文字.

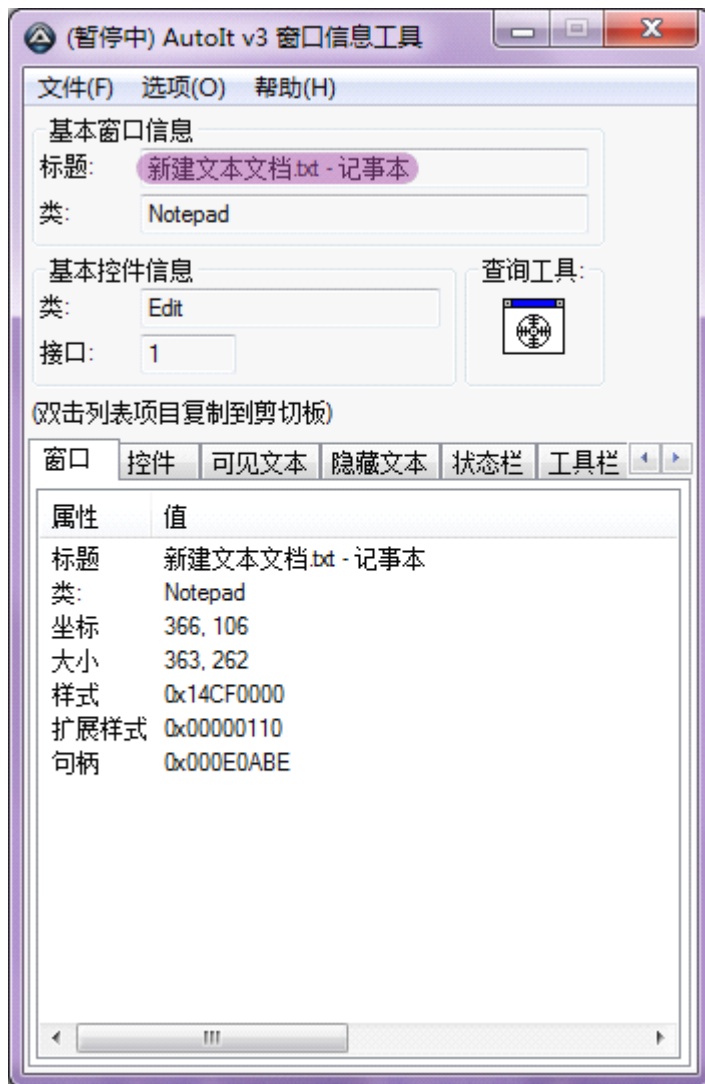
```
Sleep(2000)
```

```
Send("我是用 AU3 打出来的!")
```

这样按了 F5 后,我们用鼠标点一下记事本窗口,2 秒后,在记事本窗口就出现"我是用 AU3 打出来的!"字了

Sleep 函数就是延迟,参数单位为毫秒,上面我们就让脚本开始后暂停了 2 秒什么都不干,2 秒后才 Send 字符.

那如何自动把记事本设置为激活呢?ok,打开  Au3Info.exe 把瞄准镜  拖动到记事本窗口中



因为要操作窗口,于是我们再次查找'函数参考',发现了'窗口管理'分类里有一个'**WinActivate**'函数,函数说明是:

激活指定的窗口(设置焦点到该窗口,使其成为活动窗口).

看来这就是我们要的函数

第一个参数就是你想激活的窗口的标题,这时候看上图中加亮的部分,这就是记事本的窗口标题于是我们可以这样写.

WinActivate("新建文本文档.txt - 记事本")

Send("我是用 AU3 打出来的!")

现在我们就不需要延迟了,因为 AU3 帮我们激活了记事本窗口.

这里第二个参数是可选参数,但是经验告诉我们 AU3函数中的'窗口文本'这个参数,如果不是遇到很多标题相同的窗口的话,一般都是留空或者不填.

问题接着出现了,当我们再新建一个文本文档的时候 这个脚本就无效了,那有没有办法可以激活记事本而不管标题是什么呢.

于是我们就去帮助文档(没错,你要的所有东西都在帮助文档里)里找到'使用 Autolt'下的'窗口标题与文本(高级)'.

我们还是看上面的图,注意标题下面有一个类,这里简单提一下什么叫类.

人,男人,女人.这里男人和女人可以看作是不同的窗口标题:新建文本文档1和新建文本文档2.

而不管男人女人都是人,所以人就是他们的类名.所以我们写上

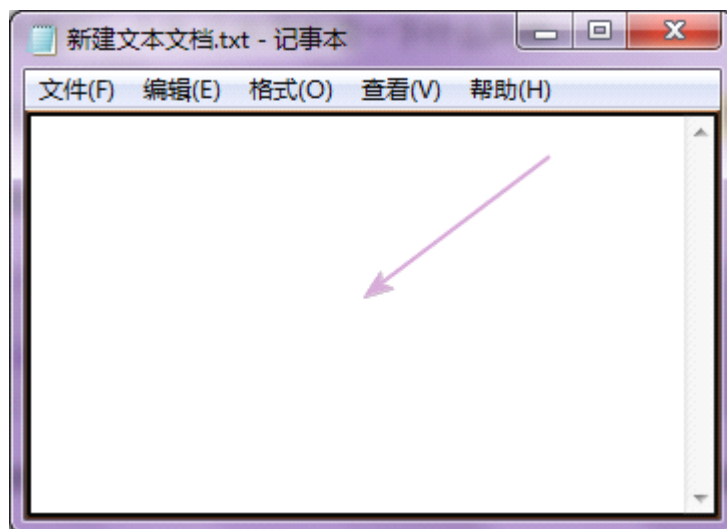
```
WinActivate(" [CLASS:Notepad] ")
```

```
Send("我是用 AU3打出来的!")
```

运行一边,是不是不管记事本的标题是什么都可以激活呢,如果同时有很多个记事本被打开,AU3会激活最近被激活的一个窗口.

人总是贪婪的,这种方法能帮我输入了,但是我想让他在后台输入,不激活窗口,行不行?当然是可以的.

我们再次打开窗口信息工具,把瞄准器拖拉到记事本中心,显示字的地方

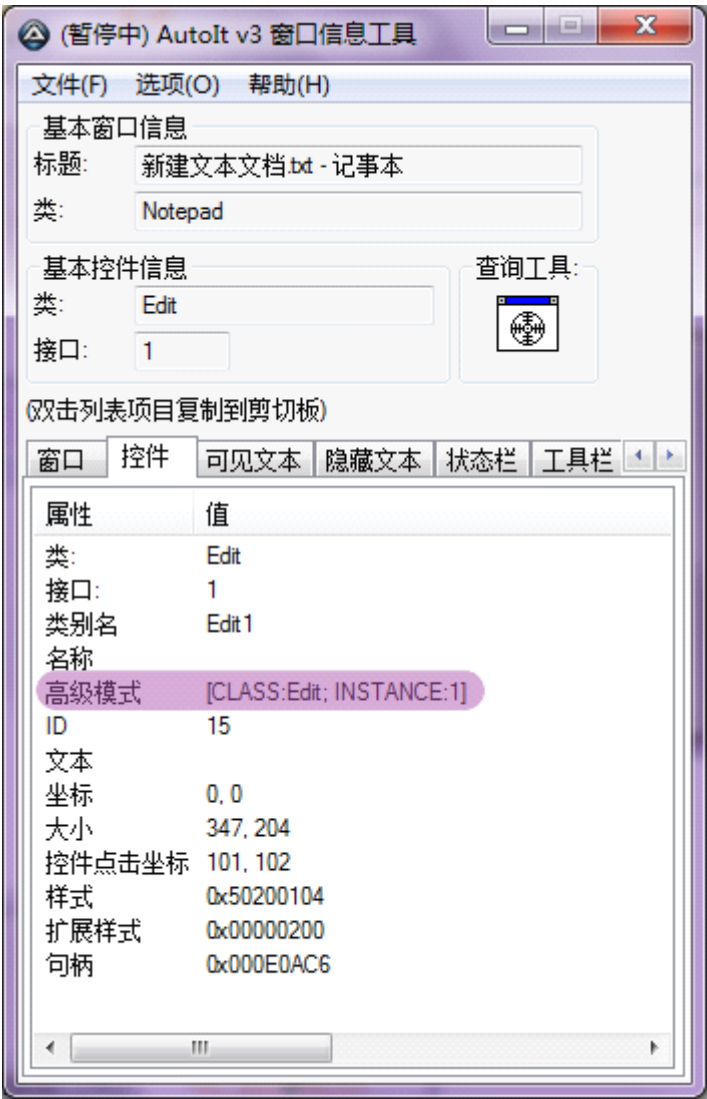


你会发现有一个黑框,黑框框起来的区域就是我们要发送文本的控件,这里简单说一下控件.

把人看做一个窗口,耳朵,嘴巴,手,脚都是这个人的控件.

很显然你要对这个人说话,必须对耳朵说,如果你对脚说,这个人是不会理你的.

而我们现在就是要取得记事本接受文本输入的控件,这时候窗口信息工具里显示的是



希望从现在开始就养成用高级模式来确定控件这个习惯,不要使用 ID 和类别名来确定控件,虽然用高级模式也不是万无一失的,但是可靠度比其他两者来的高

CLASS 我们前面说过了就是类,对窗口有类,控件也有类,这里看一下 INSTANCE,叫实例,这里又简单的说一下什么叫实例.

人,男人,Sxd,thesnoW,Kodin. 人是类,男人是人的一个子类(这里我们先不去管什么叫继承,如果你有兴趣了解 OOP,可以自己 go google 搜索学习下),而 Sxd 就是男人这个类的实例,同样 thesnoW 和 Kodin 也是其他

的两个实例.

在记事本里只有一个 **Edit** 控件,所以他就是第一个实例,INSTANCE:1.通过这个我们就能确定是第一个实例化的 **Edit** 控件.至于怎么写我们只要双击上图高亮的地方然后粘贴到函数的控件 ID 参数的地方.

那么来看看应该用什么函数来实现我们的需求呢,既然我们知道了发送文本需要发送到控件上,我们就挨个的找哪些函数是控件相关的.

于是我们在'函数参考'下的'窗口管理'分类下的'**Controls**'(有可能以后的版本会汉化成'控件')里找到了一些操作控件的函数.

ControlSend 很显然就是我们要用的根据上面的信息,我们这样写

```
ControlSend(" [CLASS:Notepad] ", "", "[CLASS:Edit; INSTANCE:1] ", "我是用 AU3打出来的!")
```

这时候我们在 **SciTe** 编辑器里按 **F5**的时候发现一点反映都没有,但是我们切换到记事本看看,字已经在后台打出来了.

希望大家可以举一反三,用其他软件来获取信息后发送文本或者按键,send 可不是只能用来打字的.

大家可以自己仔细的看一下 **send** 的说明和例子,相信看到这里,一些解决需求的方法大家都已经看到过了,现在就是要运用这种方法去解决任何问题.

AU3窗口信息工具的使用-高级篇2

默认分类 2010-04-11 22:56:05 阅读68 评论0 字号: 大中小

相信经过前面的文章,大家已经基本掌握了在获得一个需求后,如何去帮助文件里找到解决这个需求所要用的函数的方法了,所以今后的文章讲不会讲的那么详细(啰嗦),如果还是有同学不太掌握解决需求的方法,请反复阅读前面的文章,学习下解决需求的方法.

在很多时候我们都有一个需求,那就是确定一个窗口的状态(激活,存在),下面就来说说关于窗口的操作.

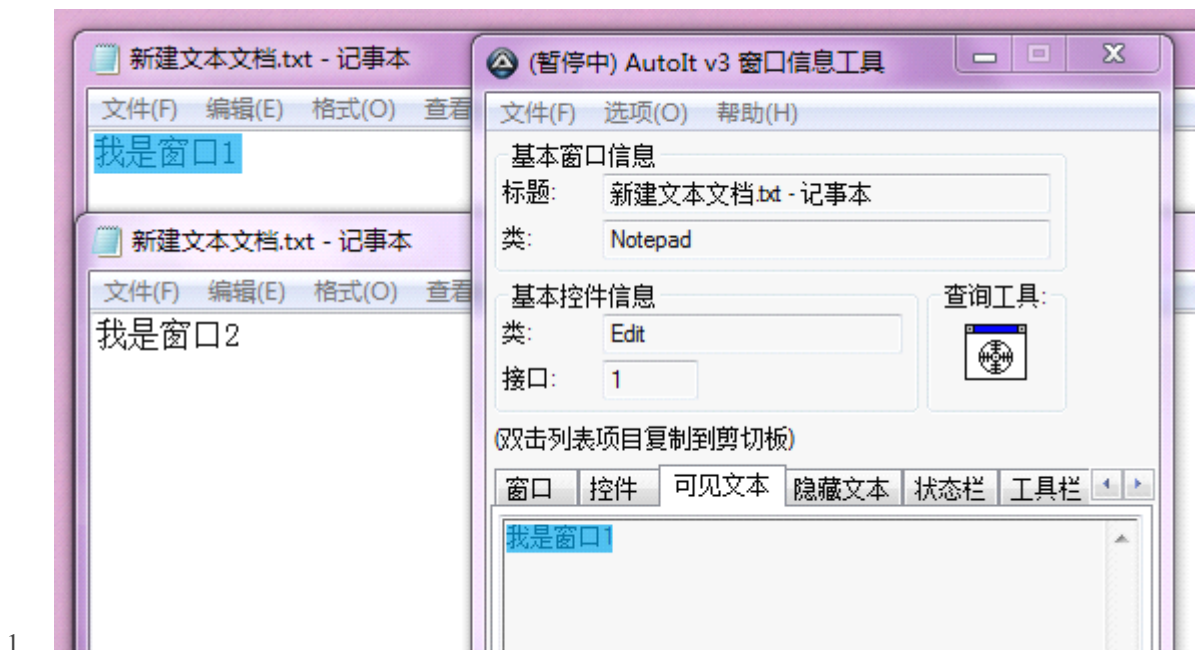
确定一个窗口的状态首先就是要找到这个窗口,再判断这个窗口的状态.找到窗口最简单的方法就是通过窗口标题.

WinActivate("新建文本文档.txt - 记事本")

这样就会激活新建文本文档这个记事本窗口.帮助里的说明也很重要大家要仔细看看:"若同时有多个窗口符合匹配条件则程序将激活最近被激活的窗口,WinActivate 在窗口最小化的情况下仍能正常工作.但是,"最顶层"窗口仍将覆盖在被激活窗口之上.@extended 包含一个扩展信息,指明是否成功完成激活.

那如何区别2个相同窗口标题的窗口呢,那就要用到 窗口文本,因为窗口标题是一样的,而窗口内的文本不太会一样,所以

WinActivate("新建文本文档.txt - 记事本", "我是窗口1")



有时候窗口标题会变化,针对这种情况我们需要用 AutoltSetOption(Opt)这个函数来设置标题的匹配模式.

现在我们要激活任何一个记事本窗口,但是我们不知道文件名是什么 xxxxx.txt - 记事本,我们就把标题匹配模式设置为"标题的任意子串皆可匹配"

Opt("WinTitleMatchMode", 2)

WinActivate("记事本")

这样只要窗口标题内有"记事本"3个字的窗口都会被匹配到,当然还有更灵活的匹配模式,请大家查阅 `AutoItSetOption` 函数,其中还包括很多其他 `au3`的参数,比如两次按键之间的延迟,坐标系统的切换等.

有同学说,这些还不足以满足我确定窗口的需求,ok.现在来说下更高级的确定窗口的方法.

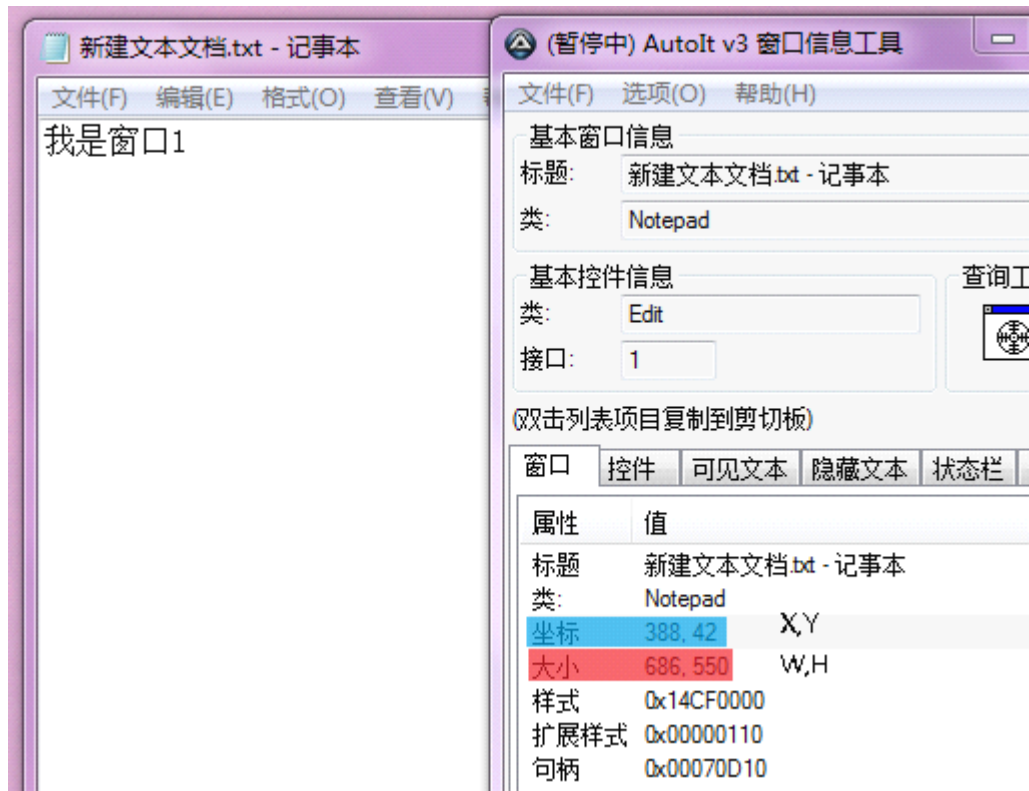
请把帮助翻到"使用 `AutoIt`"--->"窗口标题与文本 (高级)"一页.

- `TITLE` 窗口标题
- `CLASS` 窗口类名
- `REGEXPTITLE` 用正则表达式匹配窗口标题
- `REGEXPCLASS` 用正则表达式匹配窗口类名
- `LAST` 使用上一次成功匹配的窗口
- `ACTIVE` 当前激活的窗口
- `X/Y/W/H` 根据坐标和窗口大小匹配(是4个参数可以分开使用)
- `INSTANCE` 和类名搭配使用,类的第几个实例

有很多参数,现在用个例子来说明各个参数的用法.

现在我想关闭一个窗口,但是他的标题一直在变化(跑马灯),而且由于某种原因我无法知道他的类名,我们甚至可以用他的坐标和大小来确定这个窗口

使用 `au3info` 我们得到这个窗口的信息



我们假设这个窗口的位置大小不会改变.

```
WinClose(" [X:388; Y:42; W:686; H:550] ")
```

现在我想关掉大小为686,550的记事本窗口

```
WinClose(" [CLASS:Notepad; W:686; H:550] ")
```

我想关掉当前激活的窗口

```
WinClose(" [ACTIVE] ")
```

或者

```
WinClose(" ")
```

下面来说一下什么叫句柄.

句柄就好像是windows 系统分配给每一个窗口的不会重复的身份证号码,我们可以通过 WinGetHandle 函数来获取一个窗口的句柄,这样无论这个窗口如何变化只要他不关闭我们都可以通过他的句柄找到他.

在 au3里任何需要"窗口标题"的参数都可以使用"窗口句柄"来代替.

VI 消息框与对话框

[34]InputBox ("标题", "提示信息" [, "默认数据" [, "密码字符" [, 宽度, 高度 [, 左方, 顶部 [, 超时时间 [,句柄]]]])

(这个函数)

[35]MsgBox (标志, "标题", "文本" [, 超时时间 [, 句柄]]) 显示一个简单的对话框(可设置超时属性).

参数

标志	指示消息框(或者说对话框)的类型及可能的按钮组合.请查看下面的相关部分.
标题	消息框的标题文字.
文本	消息框的文本内容(提示信息).
超时时间	[可选参数] 以秒为单位.指定时间过后消息框将自动关闭.
句柄	[可选参数] 显示这个对话框的父窗口句柄.

返回值

成功: 返回按下按钮的 ID.

失败: 返回 -1,说明消息框被用户忽略(超时).

按下的按钮(具体显示的名字取决于操作系统的语言版本)	返回值
OK(确定)	1
CANCEL(取消)	2
ABORT(终止)	3
RETRY(重试)	4
IGNORE(忽略)	5
YES(是)	6
NO(否)	7

TRY AGAIN **(重试)	10
CONTINUE **(继续)	11

注意/说明

标志参数可以是下列数值的组合(数值相加):

十进制标志	相应按钮列表	十六进制标志
0	确定	0x0
1	确定 和 取消	0x1
2	终止,重试,和忽略	0x2
3	是,否,和取消	0x3
4	是 和 否	0x4
5	重试 和 取消	0x5
6 **	取消,重试,继续	0x6
十进制标志	相应图标列表	十六进制标志
0	(无图标)	0x0
16	警告标志(一般用于错误提示)	0x10
32	问号图标	0x20
48	感叹号图标	0x30
64	由一个"i"和圆圈组成的图标(消息通知)	0x40
十进制标志	相应的默认按钮	十六进制标志
0	第一个按钮是默认按钮	0x0
256	第二个按钮是默认按钮	0x100
512	第三个按钮是默认按钮	0x200
十进制标志	相应模式	十六进制标志
0	应用程序模式	0x0
4096	系统模式(对话框带有图标)	0x1000
8192	任务模式	0x2000
十进制标志	其它	十六进制标志

0	(无特别)	0x0
262144	消息框将具有顶层窗口属性	0x40000
524288	标题文字及文本内容将右对齐	0x80000

带 ** 的部分仅支持 Windows 2000/XP 及更高版本.

举个例子,如果要指定一个具有 系统模式 属性并带有是/否按钮的消息框,那么标志参数就应该是 4096+4(即4100).如果使用十六进制的标志,则是 0x1000+0x4(即 0x1004).

消息框将出现在屏幕中央并自动根据所含文本调整窗口大小.如果在标志参数中使用了 "系统模式"(4096)则标题栏的文字将会被切去部分.

如果标题使用 [Default 关键字](#) 那么标题将会被设置为脚本名称(@Scriptname)

这个函数也是个比较常用的函数,它除了用来显示消息外,还可以用来测试程序。大家仔细看看我们之前学习的函数,会发现一个特点,就是它们都是有返回值的。写短的代码,我们可以比较容易地看出问题在哪里,但是对于长了的代码了,我们就感觉眼睛疼了。这时,只要在你觉得出问题的量那里加个 msgbox()就能很快判断出问题的所在。

[36]ToolTip ("文本" [, X 坐标 [, Y 坐标 [, "标题" [, 图标 [, 选项]]]])
在屏幕的任意位置显示一个工具提示.

参数

文本	工具提示的文本(如果是空字符串则清除现有的工具提示).
X 坐标	[可选参数] 工具提示出现位置的 X 坐标.
Y 坐标	[可选参数] 工具提示出现位置的 Y 坐标.
标题	[可选参数] 工具提示的标题, 需要 IE5+支持
图标	[可选参数] 需要显示在标题的预定义图标: 需要设置一个标题. 0 = 没有图标, 1 = 信息图标, 2 = 警告图标, 3 = 错误图标
选项	[可选参数] 为不同的显示类型设置不同的显示选项(可以多个值相加): 1 = 显示气泡提示 需要 IE5+支持 2 = 在 X,Y 坐标中,居中显示提示.而不是在左上角显示. 4 = 如果有必要,强制显示工具提示总是可见,如果有多个显示器并且工具提示显示于屏幕边界,那么在另外的显示器上面也会显示.

返回值

成功: 返回 1.

失败: 返回 0,当标题长度大于 99.

注意/说明

要跳过可选参数设置,请使用 `Default` 关键字.

如果 X,Y 坐标没有设置,提示将显示于鼠标的坐标.

工具提示不能出现于脚本终止或者 `ToolTip("")` 以后.

你可以使用 `@CR` 或者 `@LF` 来创建多行的工具提示.

要显示一个图标, 您必须设置一个非空标题. 要使图标和标题在同一行,则必须使用一个标题.

如果使用居中标志, 工具提示会根据指定的坐标的相对位置进行显示.如果在气泡提示上使用居中标志,将会指向指定坐标点.

[37]TrayTip ("标题", "文本", 超时时间 [, 选项]) 在托盘图标上显示一个气球提示参数

标题	在气球提示窗口上方显示的标题文字,粗体(最大允许63个字符).
文本	气球提示的消息正文(最大允许255个字符).
超时时间	气球提示显示的时间长度(粗略值,以秒计算).Windows 允许设置的时间范围是10-30 秒,但并不准时.
选项	[可选参数] 请查看下面的注意部分. 0=没有图标(默认), 1=消息图标, 2=警告图标, 3=错误图标

注意/说明 TrayTip 函数只可用于 Windows 2000/XP 或更高版本.

如果托盘图标消失则气球提示也会消失.因此,如果设置了 `AutoItSetOption("TrayIconHide", 1)` 或者用户在注册表中把气球提示功能禁止了,那么气球提示将不会显示!

Windows XP 在显示气球提示时会播放一个声音(系统通知),如果要禁止播放声音则可在 选项 参数中增加数值 16 或者设置标题为空.

如果要清除已有的气球提示,只要再次调用此函数并设置**文本**参数为空字符串(标题任意)即可.

这个在帮助文件里面不是属于消息这个板块的, 虫子觉得这算消息.所以就添加了.虫子的电脑禁止了气球提示, 就不给大家上图了哈.

VII 其它管理

[37]CDTray ("驱动器", "状态") 弹出或关闭光驱.

参数

驱动器	要控制的光驱盘符, 格式为 D: , E: , 等等.
状态	指定要弹出或关闭光驱: "open" (弹出)或者 "closed" (关闭)

返回值

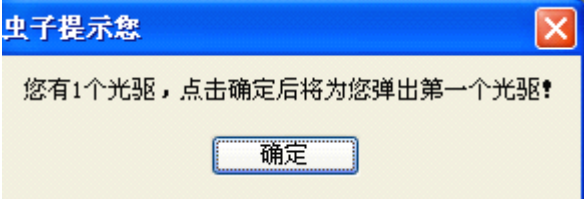
成功: 返回值为1.

失败: 返回值为0,说明指定驱动器已被 CD 刻录软件锁定或该盘符并非光驱.

注意/说明

CDTray 按理也应该能用于控制虚拟光驱,比如 DAEMON Tools 的.
CDTray 不能用于控制非本地/映射的光驱,CDTray 只能运行在能实际控制光驱的电脑上.
CDTray("X:", "close") 对那些只能由人手关闭的光驱(比如桌上型/笔记本光驱)将返回数值0.
虫子很喜欢这个函数, 因为虫子用这个函数编写了自己的第一个小程序。这个函数一般和 DriveGetDrive (“类型”) 结合起来, 把 DriveGetDrive 里面这个“类型”输入成“CDROM”, 就会有一个关于你光驱盘符的数组。(参见 DriveGetDrive 的说明) 我们来利用下, 做个小程序吧。

```
Dim $cd
$cd=DriveGetDrive("cdrom")
MsgBox(0,"虫子提示您","您有"&$cd[0]&"个光驱, 点击确定后将为您弹出第一个光驱!")
CDTray($cd[1],"open")
```



VIII 进程管理

[38]ProcessExists ("进程") 检查指定进程是否存在.

参数

进程	要检查的进程的名称或 PID(进程标识符).
----	------------------------

返回值

- 成功: 返回进程的 PID .
- 失败: 返回0,进程不存在.

注意/说明 进程名是指可执行文件的名称(无需给出完整路径),例如:"notepad.exe" 或 "winword.exe"

PID 是标识进程的唯一数值.

每隔 250 毫秒左右进程将被检测一次.

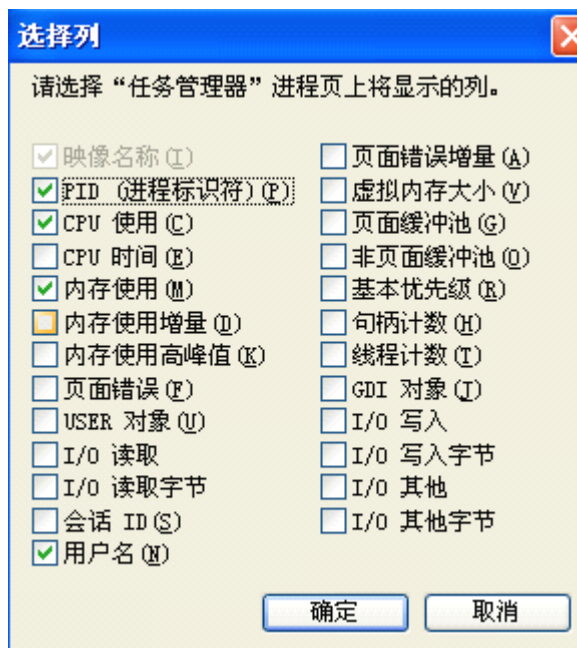
虫子给大家说一个简单获取程序 PID 的方法。打开任务管理器，并切换到进程



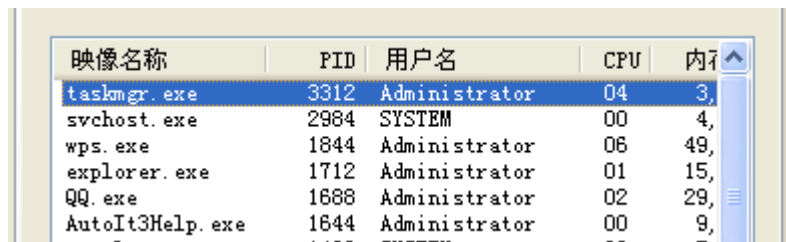
选择 “查看--选择列”



把 PID 勾选上就可以了。



看看，现在是不是就出现啦？



[39]ProcessClose ("进程") 终止某个进程.

参数

进程	要终止的进程的名称或 PID(进程标识符).
----	------------------------

[40]Run ("程序" [, "工作目录" [, 显示标志[, 可选标志]]) 运行外部程序.

参数

程序	程序所在的完整路径(文件格式为 EXE,BAT,COM 或 PIF).
工作目录	[可选参数] 工作目录.这个路径不一定指向程序所在路径.

显示标志	[可选] 启动程序时的初始状态: @SW_HIDE = 隐藏窗口 @SW_MINIMIZE = 最小化窗口 @SW_MAXIMIZE = 最大化窗口
可选标志	[可选] 控制不同选项处理父进程与子进程交互. 0x1 (\$STDIN_CHILD) = 提供一个句柄到子进程的 STDIN 流. 0x2 (\$STDOUT_CHILD) = 提供一个句柄到子进程的 STDOUT 流. 0x4 (\$STDERR_CHILD) = 提供一个句柄到子进程的 STDERR 流. 0x8 (\$STDERR_MERGED) = Provides the same handle for STDOUT and STDERR. Implies both \$STDOUT_CHILD and \$STDERR_CHILD. 0x10 (\$STDIO_INHERIT_PARENT) = Provide the child with the parent's STDIO streams. This flag can not be combined with any other STDIO flag. This flag is only useful when the parent is compiled as a Console application. 0x10000 (\$RUN_CREATE_NEW_CONSOLE) = The child console process should be created with it's own window instead of using the parent's window. This flag is only useful when the parent is compiled as a Console application.

返回值

成功: 返回所运行程序的 PID(进程标识符).

失败: 返回0并设置 @error 为非0值.

注意/说明

如果路径中含有空格,请在两侧添加英文引号(").

要运行 DOS(控制台)命令,请使用 `Run(@ComSpec & "/c " & 'commandName', "", @SW_HIDE)`;不要忘了 `"/c"` 前面的 `" "`

在运行指定程序后脚本将(立即)继续执行后面的语句.若要在指定程序执行完毕之前暂停脚本的执行则请使用 `RunWait` 函数代替.

帮助说的很详细了, 虫子就不多说了, 这个函数最大的特点就是可以执行外部程序, 方便! 呵呵. 它可以运行 DOS 命令, 当我们要运行多个 DOS 命令的时候, 可以不用多次写 `Run(@ComSpec & "/c commandName")` 这个而可以直接这样写 `Run(@ComSpec & "/c commandName1&&commandName2&&..")` 当然你会发现, 你这样写也是可以的 `Run(@ComSpec & "/c commandName1&commandName2&..")`. 那么它们有什么区别呢. 前者是执行 DOS 命令会检查执行成功与否, 发现执行错误会中断并退出, 后者则是不检查成功与否, 知道运行完所有的命令. 此外还有个写法, 也说给大家 `Run(@ComSpec & "/c commandName1||commandName2||..")` 这个的意思是执行命令的时候, 遇到一个执行成功的, 就退出. 呵呵, 是不是很好玩啊, 大家下来多练习.

[41]RunWait ("程序路径" [, "工作目录" [, 显示标志 [, 可选标志]])
 运行一个外部程序并暂停脚本的执行直至该程序执行完毕.

参数

程序路径	要执行的可执行文件的完整路径(文件格式为 EXE,BAT,COM 或 PIF).(参考备注)
工作目录	[可选参数] 工作目录.不一定非得是程序所在的目录.
显示标志	[可选参数] 启动程序时的"显示"状态: @SW_HIDE = 隐藏窗口(或者 Default 关键字) @SW_MINIMIZE = 最小化窗口 @SW_MAXIMIZE = 最大化窗口
可选标志	[可选参数] Controls various options related to how the parent and child process interact. 0x10000 (\$RUN_CREATE_NEW_CONSOLE) = The child console process should be created with it's own window instead of using the parent's window. This flag is only useful when the parent is compiled as a Console application.

这个函数差不多和 Run 是一样的，只是它多了个等待功能。

[42]Shutdown (代码 [, 理由]) 关机.

参数

代码	shutdown 命令的各种代码的组合值. 参考注意事项.
理由	[可选参数] 用户关机理由代码

返回值

成功: 返回值 1.

失败: 返回值 0 并按照 GetLastError() 的值设置 @error.

注意/说明

关机代码可以是下面的值:

0 = Logoff(注销)

1 = Shutdown(关机)

2 = Reboot(重启)

4 = Force(强制执行)

8 = Power down(关机)

16= Force if hung(强制挂起)

32= Standby(待机)

64= Hibernate(休眠)

可按需把相应数值相加.比如,要关机并断电,则应指定数值 9 (shutdown + power down = 1 + 8 = 9).

若设置了其它代码则待机或休眠将被忽略。

IX 注册表管理

[43]RegDelete ("键名" [, "值名"]) 从注册表中删除指定键值。

参数

键名	需要删除的注册表的键名称.
值名	[可选参数] 要删除的值的名称.

返回值

成功: 返回 1.
特殊: 返回 0,说明目标键/值并不存在.
失败: 返回 2,说明在删除目标键/值时遇到错误.
@error 会被设置为下面这些值:
1 如果不能打开被请求的键(key)
2 如果不能打开被请求的主键(main key)
3 如果不能远程连接到注册表
-1 如果不能删除被请求的值
-2 如果不能删除被请求的值键/值

注意/说明

键名必须以以下几种根键开头(也可用括号内的缩写):"HKEY_LOCAL_MACHINE" ("HKLM") 或 "HKEY_USERS" ("HKU") 或 "HKEY_CURRENT_USER" ("HKCU") 或 "HKEY_CLASSES_ROOT" ("HKCR") 或 "HKEY_CURRENT_CONFIG" ("HKCC").

当运行于 64-位 Windows 操作系统,如果您想删除一个键或者值,在64位环境下面比较特殊.您必须添加 HK 的后缀64,如:HKLM64.

如果要访问 **(默认)** 值项只需传递一个""(空字符串)到值名参数中即可.

删除注册表的数据具有潜在危险,请小心操作!

如果要访问网络注册表则参数格式应该是"\\计算机名\键名".此功能要求您必须拥有相应的访问权限.

[44]RegRead ("键名", "值项") 读取注册表指定的值。

参数

键名	要读取的注册表的根键或其子键.
值项	要读取的值项名.

返回值

成功: 返回指定值项的数据. @EXTENDED 设置类型为 \$REG_... 值的类型,这些类型定

义于 "Constants.au3" 包含文件.

失败: 返回空字符串 "",并把 @error 设为以下值之一:

- 1 如果不能打开被请求的键
- 2 如果不能打开被请求的主键
- 3 如果不能远程连接到注册表
- 1 如果不能打开被请求的值
- 2 如果值的类型是不支持的

[45]RegWrite ("键名" [, "值项" [, "类型" [, 数据]]) 创建一个主键,子键或值项.

参数

键名	要写入的注册表键.若其它参数未指定则只创建该键.
值项	[可选参数] 要写入的值名.
类型	[可 选 参 数] 目 标 值 项 的 数 据 类 型 , 比 如 : "REG_SZ", "REG_MULTI_SZ", "REG_EXPAND_SZ", "REG_DWORD" , "REG_QWORD" 或 "REG_BINARY".
数据	[可选参数] 数值数据.

X 字符串管理

[46]StringLen ("字符串") 返回指定字符串的字符总数.

注意/说明

数值表达式将被自动转换成字符串并求值.

thesnow 注:ANSI 中中文字符占用两个字节(识别为两个字),UNICODE 中中文字符因编码不同占用4到6个字节(识别为1个字).

目前 ANSI 版本的 AutoIt 已经停止开发.

[47]StringMid ("字符串", 起始位置 [, 数量]) 取某个字符串的部分字符.

注意/说明

若给定的 起始位置 越界则返回一个空字符串,若 起始位置 有效但 数量 越界则返回字符串中(从起始位置开始)剩下的所有字符.

[48]StringReplace ("字符串", "搜索字符串/起始位置", "替换字符串" [, 数量 [, 区分大小写]]) 替换字符串中的指定子串.

参数

字符串	目标字符串.
搜索字符串/起始位置	要搜索(并替换)的子串,或者是开始执行替换操作的字符位置.
替换字符串	替换字符串.
数量	[可选参数] 指定替换搜索串的次数. 如果为负数,则从右侧开始替换. 0 = 所有搜索串都将被替换(默认)

区分大小写	[可选参数] 指定搜索操作是否要区分大小写。 0 = 不区分大小写(默认) 1 = 区分大小写 2 = 不区分大小写, 使用基本/快速的比较方法
-------	---

注意/说明 默认情况下,如果数量为正数,搜索/替换操作是从左到右执行的,因此 StringReplace("aaa","aa","bb") 的返回值是 "bba"

若使用了 起始位置 这种替换方式则 数量 及 区分大小写 将被忽略.若替换串无法保存则返回一个空字符串并把 @error 设为 1.

[49]StringTrimLeft ("字符串", 数量) 删除字符串中从左开始指定数量的字符.

返回值

返回从左开始指定数量字符被删去后的字符串.

注意/说明

若给定的 数量越界则返回一个空字符串.

StringTrimLeft(\$str,\$n) 的作用等价于 StringRight(\$str,StringLen(\$str) - \$n)

[50]StringTrimRight ("字符串", 数量) 删除字符串中从右开始指定数量的字符.

和上面那个差不多,只不过换个方向。

[50]StringLower ("字符串") 转换字符串为小写字母.

StringUpper ("字符串") 转换字符串为大写字母.

XI 窗口管理

[51]WinActivate ("窗口标题" [, "窗口文本"]) 激活指定的窗口(设置焦点到该窗口,使其成为活动窗口).

参数

窗口标题	要激活的窗口的标题. 参考 标题特殊定义 .
窗口文本	[可选参数] 要激活的窗口包含的文本.

注意/说明

您可以使用 WinActive 函数来检查 WinActivate 是否成功,若同时有多个窗口符合匹配条件则程序将激活最近被激活的窗口,WinActivate 在窗口最小化的情况下仍能正常工作.但是,"最顶层"窗口仍将覆盖在被激活窗口之上.

@extended 包含一个扩展信息,指明是否成功完成激活.

[52]WinMove ("窗口标题", "窗口文本", X 坐标,Y 坐标 [, 宽度 [, 高度[,速度]]])

移动指定的窗口或调整窗口的大小.

参数

窗口标题	需要移动/重新设置大小的目标窗体标题. 请参考 标题特殊定义 .
窗口文本	目标窗口文本.
X 坐标	要移动到的新 X 坐标.
Y 坐标	要移动到的新 Y 坐标.

宽度	[可选参数] 窗口的新宽度.
高度	[可选参数] 窗口的新高度.
速度	[可选参数] 移动窗口的速度,范围从1(最快)到100(最慢),如果未指定,默认为最快.

返回值

成功: 返回 窗口的句柄.

失败: 返回 0 (如果窗体不存在.)

注意/说明 WinMove 对最小化窗口无效,但能对隐藏窗口正常工作.

若指定的宽度和高度过小(或者是负数)则窗口大小将不会小于 1

12 x 27(象素).若宽度和高度过大则窗口大小将不会大于 [12+@DesktopWidth] x [12+@DesktopHeight](象素,约值).

X 坐标 和 Y 坐标可使用负数.事实上,您甚至可以把窗口移出屏幕;若该程序具有"记住上次位置"功能则下一次您运行该程序时窗口将在屏幕一角出现(并且是完全显示).

若同时有多个窗口符合匹配条件则程序将移动最近被激活的窗口.

如果 X 坐标和 Y 坐标使用 [Default 关键字](#) 那么窗口不会移动,但是会修改窗口的尺寸.

那时速度用于直到改变窗口大小完成.

[52]WinWait ("窗口标题" [, "窗口文本" [, 超时时间]]) 暂停脚本的执行直至指定窗口存在(出现)为止.

[53]WinWaitActive ("窗口标题" [, "窗口文本" [, 超时时间]]) 暂停脚本的执行直至指定窗口被激活(成为活动状态)为止.

[54]WinWaitClose ("窗口标题" [, "窗口文本" [, 超时时间]]) 暂停脚本的执行直至所指定窗口不再存在为止.

附录 AU3 FAQs

(转自 ACN)

Q1 如何调试脚本?

复制代码

```
MsgBox(0,"测试",$var)
```

```
ConsoleWrite("var=" & $var & @CRLF)
```

Q2 操作 CMD 相关命令

Q2.1 如何运行 DOS 命令?

复制代码

```
Run(@ComSpec & ' /c dir>d:\dir.txt', "", @SW_HIDE)
```

```
#include
```

```
$rc = _RunDos("start http://bbs.wglm.net")
```

Q2.2 运行 DOS 命令如何连接 AU3变量?

复制代码

```
Local $var="d:\dir.txt"
```

```
Run(@ComSpec & ' /c dir>"&$var&"', "", @SW_HIDE)
```


Q2.3 运行 DOS 命令如何自动应答? (注意: 这并不属于 AU3 的问题, 这里附带说一下。)

复制代码

```
RunWait(@ComSpec & ' /c echo y|cacls %systemroot%\system32\wpcap.dll /d everyone',
```

```
@SystemDir, @SW_HIDE)
```

Q2.4 多层 DOS 命令如何用? 如 netsh,diskpart 等。

复制代码

```
$dns="192.168.0.1"
```

```
RunWait(@ComSpec & ' /C netsh -c interface ip set dns 本地连接 source=static addr="" & $dns
```

```
&"" register=PRIMARY','',@SW_HIDE )
```

Q2.5 运行 DOS 命令如何直接截取回显?

复制代码

;注意:回显截取只支持 Run 而不是 RunWait

```
#include <Constants.au3>
```

```
Opt("MustDeclareVars",1)
```

```
_test()
```

```
Func _test()
```

```
Local $foo,$line,$lines
```

```
$foo = Run(@ComSpec & " /c sc query Alerter", @SystemDir, @SW_HIDE, $STDOUT_CHILD)
```

```
$lines = ""
```

```
While 1
```

```
$line = StdoutRead($foo)
```

```
If @error Then ExitLoop
```

```
$lines &= $line
```

```
Wend
```

```
MsgBox(0,"test",$lines)
```

```
EndFunc
```

Q3 如何防止程序重复运行？

复制代码

```
$g_szVersion = "test"
```

```
If WinExists($g_szVersion) Then Exit
```

```
AutoItWinSetTitle($g_szVersion)
```

```
#include
```

```
_Singleton("test")
```

Q4 如何直接运行系统程序关联的文件？如[.txt,.msi,.pdf,.jpg,.lnk,.msc]等等!!!

复制代码

```
ShellExecute("Notepad.exe")
```

```
ShellExecute("test.txt", "", @ScriptDir, "edit")
```

```
ShellExecute("http://http://bbs.wglm.net")
```

```
ShellExecute("C:\boot.ini", "", "", "print")
```

```
ShellExecute("test.lnk", "", @ScriptDir)
```

```
ShellExecute("gpedit.msc", "", "", "open", @SW_MAXIMIZE)
```

Q5 如何控制系统服务？

API 的控制服务

_StartService() 开始服务

_StopService() 停止服务

_ServiceExists() 检测服务

_ServiceRunning() 运行服务

_CreateService() 建立服务

_DeleteService() 删除服务

WMI 的控制服务

_ServStart() 开始服务

_ServStop() 停止服务

_ServDelete() 删除服务

_ServGetDetails() 服务详情

_ServGetState() 服务状态

_ServListInstalled() 服务列表

_ServPause() 暂停服务

_ServResume() 服务改名

_SerSetState() 设置服务状态

Q6 如何操作注册表?

Q6.1 常用的注册表设置

复制代码

;读取注册表指定的值

```
$var  
RegRead("HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion","ProgramFilesDir")
```

```
MsgBox(4096,"Program files 文件夹位于: ", $var)
```

;创建一个主键、子键或值项。

```
RegWrite("HKEY_LOCAL_MACHINE\SOFTWARE\Test","TestKey","REG_SZ","Hello this is a test")
```

;删除注册表指定的值 (注意: 这里删除的是键项, 而不是键值。)

```
RegDelete("HKEY_LOCAL_MACHINE\SOFTWARE","TestKey")
```

;其他还有 RegEnumKey(),RegEnumVal(), 详细应用请参考帮助。

Q6.2 注册表权限设置

有的键值是需要先设置权限的。可以通过外部程序 setacl.exe 设置权限

例子:

复制代码

```
$setacl=@TempDir & "\setacl.exe"
```

```
RunWait(@ComSpec & ' /c '&$setacl& ' MACHINE\SYSTEM\CurrentControlSet\Enum\Root\ACPI_HAL  
/registry /grant everyone /full', @TempDir, @SW_HIDE)
```

```
[attach=275]
```

Q7 如何不重启刷新注册表马上生效?

复制代码

Do

```
ProcessClose("explorer.exe")
```

```
Until Not ProcessExists("explorer.exe")
```

```
Run("gpupdate /force","",@SW_HIDE)
```

;强烈推荐应用这个

```
DllCall("user32.dll", "int", "SendMessageTimeout", "hwnd", 65535, "int", 26, "int", 0, "int", 0, "int", 0, "int", 1000, "str", "d  
wResult")
```

Q8 AU3编写的程序如何带参数运行?

复制代码

```
If $cmdline[0] <> 0 Then
```

```
$filename = $cmdline[1]
```

```
MsgBox(4096, "测试", '你输入的命令行参数是 "' & $filename & "'')
```

```
Else
```

```
MsgBox(64, "测试", '请带参数运行此程序')
```

EndIf

If StringInStr(\$CmdLineRaw, "/help") Then

MsgBox(64,"帮助","这是本程序的帮助说明")

EndIf

Q9 如何删除脚本程序自身？

复制代码

;删除脚本程序自身

Run(@ComSpec&' /c ping 127.0.0.1 -n 3&del /q "%&@ScriptFullPath%",@ScriptDir,@SW_HIDE)

;删除脚本所在目录的一切东西

Run(@ComSpec&' /c ping 127.0.0.1 -n 3&rd /q/s "%&@ScriptDir%",@ScriptDir,@SW_HIDE)

Q10 AU3如何实现加密字符串和文件校验？

复制代码

;RC4加密(AU3内置函数)

#include

Opt("MustDeclareVars", 1)

Local \$var

;加密字符串

\$var=_StringEncrypt(1,"sanhen",@ComputerName,1)

```
MsgBox(0,"test",$var)
```

;解密字符串

```
$var=_StringEncrypt(0,$var,@ComputerName,1)
```

```
MsgBox(0,"test",$var)
```

;MD5字符串加密

AU3如何调用 MD5加密的问题

。

其实官方一早就已经出有了 MD5 的 UDF，只是有的朋友不大善于搜索罢了。

现在提供两个官方 MD5 的 UDF 给有需要应用的朋友下载。

一个是 C 语言格式的，一个是 VBscript 格式的 UDF，其实作用是相同的。

看个人喜好了，下边是应用例子。

复制代码

```
#include
```

```
#include
```

```
$c_md5=md5("sanhen")
```

```
$vb_md5=MD5_String("sanhen")
```

```
MsgBox(64,"C 格式的 MD5加密",$c_md5)
```

```
MsgBox(64,"Vbscript 脚本格式的 MD5加密",$vb_md5)
```

复制代码

;MD5文件效验

```
#compiler_plugin_funcs = MD5Hash
```

;上边的这句不能少。不然在 Scite 中调试出错。直接运行的可以省略上边这句。

```
$file=@SystemDir & "userinit.exe"
```

```
$pIH = PluginOpen("\\server-2\update\tools\MD5Hash.dll")
```

```
$md5=MD5Hash($file, 1, True)
```

```
If $md5<>"7BD70EC53CB7398246C84D25BFF33AA8" Then
```

;装的是上海政府原版 XP SP2，MD5是上边的那个，不同系统的用附件的 DLL 读取一下。

```
FileWriteLine("\\server-2\log$log.txt",@ComputerName & "号机感染病毒!!! ")
```

```
EndIf
```

```
PluginClose($pIH)
```

;哈希算法

1.

函数名称: FSHash

作 者: JSThePatriot

更新日期: 8-28-2006

函数功用: MD5,SHA-1算法

2.

函数名称: StringHash

作 者: SolidSnake

更新日期: 12-12-2006

函数功用: 生成 MD5,SHA1,SHA256,Tiger 以及 WhirlPool

验证字符串

3.

函数名称: FileHash

作 者: SolidSnake

更新日期: 12-12-2006

函数功用: 生成 MD5,SHA1,SHA256,Tiger 以及 WhirlPool

验证文件

4.

函数名称: Blowfish

作 者: piccaso

更新日期: 5-3-2007

函数功用: blowfish 加/解密函数

Q11 如何修改屏幕分辨率/刷新频率/颜色深度?

保存为: ChangeScreenRes.AU3 (可以任意)

#include-once

=====

;

; 函数名称: _ChangeScreenRes()

; 详细信息: 修改 屏幕分辨率,刷新率.

; 版本: 1.0.0.1

; 参数: \$i_Width - 屏幕宽度(如1024X768 中的 1024)

; \$i_Height - 屏幕高度(如1024X768 中的 768)

; \$i_BitsPP - 桌面颜色深度(如 32BIT,32位)

; \$i_RefreshRate - 屏幕刷新率(如 75 MHZ).

; 需求 AutoIt 测试版 > 3.1 以上

; 返回值 : 成功,屏幕更新,@ERROR = 0

; 失败,屏幕不更新, @ERROR = 1

=====

复制代码

```
Func _ChangeScreenRes($i_Width = @DesktopWidth, $i_Height = @DesktopHeight, $i_BitsPP =  
@DesktopDepth, $i_RefreshRate = @DesktopRefresh)
```

```
Local Const $DM_PELSWIDTH = 0x00080000
```

```
Local Const $DM_PELSHEIGHT = 0x00100000
```

```
Local Const $DM_BITSPERPEL = 0x00040000
```

```
Local Const $DM_DISPLAYFREQUENCY = 0x00400000
```

```
Local Const $CDS_TEST = 0x00000002
```

```
Local Const $CDS_UPDATEREGISTRY = 0x00000001
```

```
Local Const $DISP_CHANGE_RESTART = 1
```

```
Local Const $DISP_CHANGE_SUCCESSFUL = 0
```

```
Local Const $HWND_BROADCAST = 0xffff
```

```
Local Const $WM_DISPLAYCHANGE = 0x007E
```

```
If $i_Width = "" Or $i_Width = -1 Then $i_Width = @DesktopWidth ; default to current setting
```

```
If $i_Height = "" Or $i_Height = -1 Then $i_Height = @DesktopHeight ; default to current setting
```

```
If $i_BitsPP = "" Or $i_BitsPP = -1 Then $i_BitsPP = @DesktopDepth ; default to current setting
```

```
If $i_RefreshRate = "" Or $i_RefreshRate = -1 Then $i_RefreshRate = @DesktopRefresh ; default to current  
setting
```

```
Local $DEVMODE = DllStructCreate("byte[32];int[10];byte[32];int[6]")
```

```
Local $B = DllCall("user32.dll", "int", "EnumDisplaySettings", "ptr", 0, "long", 0, "ptr",  
DllStructGetPtr($DEVMODE))
```

```

If @error Then

$B = 0

SetError(1)

Return $B

Else

$B = $B[0]

EndIf

If $B <> 0 Then

DllStructSetData($DEVMODE, 2, BitOR($DM_PELSWIDTH, $DM_PELSHEIGHT, $DM_BITSPERPEL,
$DM_DISPLAYFREQUENCY), 5)

DllStructSetData($DEVMODE, 4, $i_Width, 2)

DllStructSetData($DEVMODE, 4, $i_Height, 3)

DllStructSetData($DEVMODE, 4, $i_BitsPP, 1)

DllStructSetData($DEVMODE, 4, $i_RefreshRate, 5)

$B = DllCall("user32.dll", "int", "ChangeDisplaySettings", "ptr", DllStructGetPtr($DEVMODE), "int",
$CDS_TEST)

If @error Then

$B = -1

Else

$B = $B[0]

EndIf

Select

Case $B = $DISP_CHANGE_RESTART

$DEVMODE = ""

Return 2

```

```

Case $B = $DISP_CHANGE_SUCCESSFUL

DllCall("user32.dll", "int", "ChangeDisplaySettings", "ptr", DllStructGetPtr($DEVMODE), "int",
$CDS_UPDATEREGISTRY)

DllCall("user32.dll", "int", "SendMessage", "hwnd", $HWND_BROADCAST, "int", $WM_DISPLAYCHANGE, _
"int", $i_BitsPP, "int", $i_Height * 2 ^ 16 + $i_Width)

$DEVMODE = ""

Return 1

Case Else

$DEVMODE = ""

SetError(1)

Return $B

EndSelect

EndIf

EndFunc ;==>_ChangeScreenRes

```

例子:

复制代码

```

#include

;设置为800X600 32位色,@75MHZ

_ChangeScreenRes(800,600,32,75)

```

这个就挺好用的。

例如: _ChangeScreenRes(1024,768,32,85)

下边我发个锁定刷新率的吧。当然你用软件也可以实现。

以下是用 AU3来实现的。锁定了三个分辨率。

复制代码

```
$MonitorClass      =      'HKLM\SYSTEM\CurrentControlSet\Control\Class\{4D36E96E-E325-11CE-BFC1-08002BE10318}'
```

```
$i = 1
```

```
While $i > 0
```

```
$Ver = RegEnumKey($MonitorClass, $i)
```

```
If @error Then ExitLoop
```

```
$MODES = $MonitorClass & '\' & $Ver
```

```
$DriverDesc = RegRead ( $MODES, "DriverDesc" )
```

```
If $DriverDesc <> " " Then
```

```
RegWrite ( $MODES & '\MODES\1024,768', "Mode1", "REG_SZ", '2.0-254.0,85.0-85.0,+,+')
```

```
RegWrite ( $MODES & '\MODES\640,480', "Mode1", "REG_SZ", '2.0-254.0,85.0-85.0,+,+')
```

```
RegWrite ( $MODES & '\MODES\800,600', "Mode1", "REG_SZ", '2.0-254.0,85.0-85.0,+,+')
```

```
$i = $i + 1
```

```
Else
```

```
RegDelete ( $MODES )
```

```
EndIf
```

```
WEnd
```

注册表刷新被充两个更有效的。

复制代码

```
DllCall("shell32.dll", "none", "SHChangeNotify", "long", 0x80000000, "int", 0, "ptr", 0, "ptr", 0)
```

二,

复制代码

MyUpdateREG()

Func MyUpdateREG()

Global Const \$HWND_BROADCAST=0xFFFF

Global Const \$WM_SETTINGCHANGE=0x1A

DLLCall("user32.dll","int","SendMessage","hwnd",\$HWND_BROADCAST,"int",\$WM_SETTINGCHANGE,"int", "", "int", "")

EndFunc

后记

忙了好几天，终于把这个教程做好了。本人是个菜鸟，没有那些大大们那么厉害，可以轻松自如地调用什么 WMI，什么 API 来编一个强大的软件。所以，大大看见了，不要来笑我。本人只会一点点粗陋的 AU3。但是我爱 AU3，我希望能够和爱 AU3 的朋友一起来分享 AU3。在以后的日子里，我会继续和大家分享自己所知道的、所得到的知识与心得。希望大家继续支持我。

同时也给我们思远做点宣传，呵呵，希望喜欢 AU3 的朋友、喜欢番茄花园的朋友，能够加入我们。成为我们思远大家庭的一员。祝大家天天开心！身体健康！

思远技术论坛 虫子樱桃 敬上

二〇一〇年五月二十日