



AU3 入门教程



AU3 爱好者联盟 虫子樱桃 出品

====前言=====

在看这本小小的电子书之前，也许您也和当初的我一样，不知道 Au3 是何种东西，也许您只是有听说过，却一直没有时间来试一试。

结识 Au3 完全是一个偶然，在去年夏天的时候，偶然在网上看见了 SkyFree 大大的《Let's autoit》这本书，没事的时候，看了看，做了点笔记，但是完全不懂。后来第二年我所学的专业要学习网页编程，自己硬着头皮去看了一点 VbScript 的书籍和视频。后来又迷上系统精简。偶然一天，又得到 SkyFree 大大的《XP 不完全攻略》，里面有 Autoit 的常用函数，因为好奇，就自己谷歌了一个 Autoit 的汉化版，照着帮助看，看了好久，完全没有进展，然后就尝试写一些小软件，不明白的就看帮助文档，后来才渐渐得明白 -----AU3 是要在练习中才能明白的。

希望这本小册子一样的电子书，能够让同样喜欢 AU3 的您，有惊喜，有收获！

AU3 爱好者联盟 虫子樱桃

2010 年 9 月 26 日

目录

零、AU3 的下载与安装

AU3 的下载

AU3 的安装

我们的第一个 AU3 程序

一、基本概念

变量与常量

数据类型

判断与循环之判断

判断与循环之循环

二、常用函数串讲

【1】简单的对话框--msgbox 函数

【2】显示一个输入框--inputbox 函数

【3】关机函数 shutdown

【4】执行一个程序

【5】将窗口玩转到底--其他几个与窗体相关的函数

【7】窗口是这样练成的

【8】键鼠模拟--将自动化进行到底

【9】字符串操作

【10】FileInstall

【11】_FileListToArray

篇外篇

你 DllCall 了吗？

四、后记

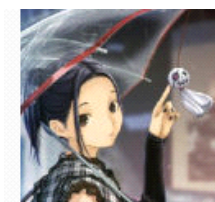
感谢

零、**AU3** 的下载与安装

AU3 的下载

Au3 的原作者是 Jonathan Bennett 及 AutoIt 小组，他们的官方网站是 <http://www.autoitscript.com/autoit3/>，呵呵，外国朋友的作品。如果您英语水平好，可以去下载一个原版耍耍，不过俺英文不咋的，虽然说基本上看得明白一些，但是看久了还是会头疼。还好，国人作出了汉化版。大家可以谷歌一下“AUTOIT 中文”，找到 ACN 的官网 <http://www.autoitx.com/>，然后按照如图示操作





Administrator



帖子 686
精华 0
积分 1910
威望 233 点
金钱 21451 块
贡献 1101 分
阅读权限 200
在线时间 454 小时
注册时间 2008-5-2
最后登录 2010-9-29

[原创] 3.3.6.1 第一汉化版下载(2010-06-07)

最新汉化版本下载:

最新汉化版本所有文件更新于(非独立安装文件):

<http://autoit-cn.googlecode.com/svn/trunk/>

项目所在: <http://autoit-cn.googlecode.com>

对于狂热FANS。请自行使用SVN客户端检出。(永远比现在发布的汉化版本新)

最新的帮助文件,也可以在这里下载。(SVN上比这里新)



这些就是下载地址,我们这个教程用的是3.3.6.1这个版本

新网站,新的开始.

3.3.6.1 第一汉化版(2010-06-07)

http://autoit-cn.googlecode.com/files/AUTOIT_3.3.6.1.exe

3.3.6.1 预发布版本(2010-04-28)

http://autoit-cn.googlecode.com/files/AUTOIT_3.3.6.1-PRE.exe

3.3.5.6 第一汉化版(2010-03-04)

AU3 的安装

将下载下来的 Au3 双击运行,安装即可。**一般的杀毒软件可能会把 AU3 报为病毒,设置为信任即可,如果您的杀毒软件直接把 Au3 给消灭了,建议您卸载该杀毒软件换个小 A OR 者小红伞或者不玩 Au3 OR 裸奔。。**



AUTOIT_3.3.6.1-PRE.exe
AutoIt 安装程序
thesnoW

等您的电脑上安装好了 Au3, 好吧, 给自己加油一下! 因为前面的一些知识可能因为是理论, 比较枯燥, 但是请确保您心中只有一个信念: **I love Au3 very much!**

我们的第一个 AU3 程序

都说光说不练是假把式, 好吧, 那咱来写咱们的第一个程序。

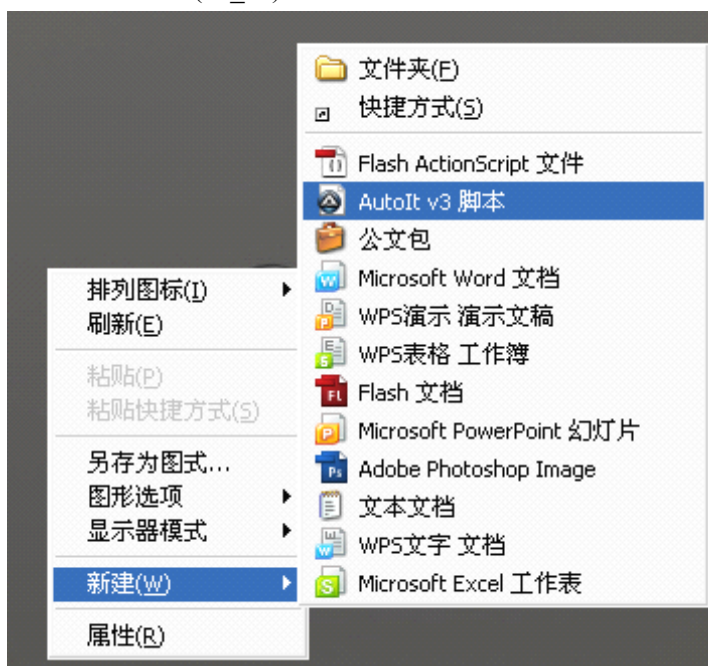
首先建立一个 AU3 文件, 有两种方法

【方法 1】在桌面找到 SCITE 编辑器, 选择“文件--新建脚本”然后“文件--文件另存为”即可。当然您可以先写代码, 然后再“文件--保存脚本”。呵呵





【方法 2】使用这种方法的前提是您之前运行了 SCITE 编辑器，如果您之前没有运行 SCITE 编辑器，在新建中是不会有下图中新建的项目的。另外在 Win7 中好像用 AU3 好像也有一些问题。至于是啥问题，因为我很久没有用 win7 了周围也没有用 win7 的同学，也不好跟大家具体说。O(∩_∩)O~



接下来就是输入脚本啦。用 SCITE 编辑器打开刚才新建的脚本（当然其他文本编辑器也可以，只要文件扩展名是 au3 就行），输入以下代码：

```
MsgBox(0,"hello","我的第一个 AU3 程序")
```

保存，然后选中脚本文件



运行脚本



好了，接下来让我们正式进入 AU3 的学习！

一、基本概念

变量与常量

变量与常量在 AU3 中都是以“\$”为开头的一些东东，那么他们有何区别呢？听俺慢慢道来。

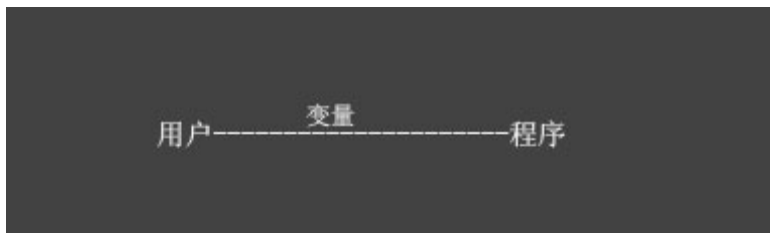
变量

最近看黑羽的《AS3 殿堂之路》里面有一个说法比较形象，原话是：

初学者往往误以为变量就是数据，其实上并不是这样，变量好比是一个遥控器，指向我们要操作的数据。对变量进行操作，变量指向的数据就会发生相应的变化。

变量必须先申明再使用，不然编译器会报错。那么为什么要先申明变量呢？道理很简单，你必须先告诉 Flash Player 建立一个遥控器，才能给遥控器起名字，并使用这个遥控器。不然，你觉得 Flash Player 能怎么做呢？遥控器连名字都没有。Flash Player 怎么找到并操作它呢？

当然，黑羽说的是 ActionScript，不是我们说的 AU3（在以后的文章里，我们就使用 Autoit 通俗的叫法 AU3 来代替了），但是他们之间是有很多共通之处的。虽然 AU3 的语法不如 ActionScript 那样严格，在使用一个变量之前，可以不申明就使用（不申明就使用，就得赋值，当然包括赋值为一个变量与赋值为一个常量，后面我们会讲到），但是我们不建议不申明变量就使用，因为对于很多语法检查严格的脚本甚至语言来说，这样是一个不好的习惯，坏习惯就好像抽烟，是很难戒掉的。您可以把变量想象成一根电线，而把您与电脑分别想成供电器与用电器。简单来说就是下面的图示：



图一 用户通过变量来给程序提供“电能”。

在 Au3 中申明变量一般使用 Dim 来进行，当您申明一个变量时，您可以不立即赋值给它，但是您必须申明一下它，这样有两个好处：一是告诉系统，您要建立一个变量，让系统在内存中开辟一个地方，呵呵，就好比您上银行，提前挂个号，而您并不立即去。当然您可以在要使用变量的时候再来申明，不过，当您的代码很长很长的时候，通过查找找变量是件很痛苦的事情。二是，使用变量之前先申明是一个好习惯，因为 AU3 毕竟只是一种脚本语言而已，很多事情还得靠我们学习高级语言来实现。下面是一些申明变量的正确写法：

【1】 Dim \$a=1

Dim \$b=2

[注]这种方式用在变量比较少的情況下，变量多的时候，咕~~(╯_╯)b，不敢想象啊！肯定十分冗长。

【2】 Dim \$a,\$b

\$a=1

\$b=2

[注]这种赋值的形式一般用在变量比较多时候，各个变量用英文的逗号分隔开来。代码较前者来说会短一点。

【3】 Dim \$a=1,\$b
\$b=2

[注]这种赋值形式，测试可用。是一种不很规范的赋值形式。不推荐使用。

常量

说完变量，接着我们来说下常量。

所谓常量，就是不变的量，O(∩_∩)O~，譬如说我们常见的 π ，它就是个常量，在Au3中，常量用Const来进行申明。Const告诉系统在内存中开辟一个类似于“一次性存储”（有点像光盘一次性刻录的样子，常量被赋值后，一般是“只读”的，不会被轻易改变的）的空间。如果对已经申明并赋值的常量进行赋值，脚本将会报错。例如，我们写如下脚本：

```
Const $a=3
```

```
$a=6
```

```
MsgBox(0,"",$a)
```

运行之，会看见如图二提示：



图二

同样地，对于一个已经存在的变量，常量也是不能够将其拿来申明的。看看下面例子：

```
Dim $a
```

```
Const $a=12
```

```
MsgBox(0,"",$a)
```

运行后，出现如图三对话框：



图三

常量的赋值形式基本和变量的赋值形式差不多，只是Dim被换成Const而已。为节约篇幅，这里就不赘述了。

数据类型

在编程中，存在很多种数据类型，那么多的数据类型有什么用呢？举个例子，譬如说你是个会很多种语言的语言学家，如果你看见黄种人那么你肯定会自然而然地想到用亚洲地区的语言跟他交流。但是如果你没有这个概念，那么你就会一种语言一种语言地使用，跟他交流，这样势必会造成很多麻烦。数据类型就是类似于见到黄种人想到用亚洲语言的思维方式一样的一种东东。也许这种说法不是很妥帖，但是基本就是这个意思。合理使用不同的数据类型，可以提高程序执行效率。数据类型的本质，我觉得就是一种分类，windows 的核心我觉得也在于此。（本人不是学编程的，所以说法难免有不妥之处，敬请谅解！）文件管理效率的提高，也通常从分类的详细程度来提高。

Au3 中数据类型主要有数字型、字符型、布尔型。

数字型

简单来说，数字型就是我们常见的阿拉伯数字 0123456789，呵呵。他们常用来参与数值运算（加减乘除之类的）或者作为排列顺序的工具以及一些函数的参数等。

字符型

字符型数据即是那些不参与算术运算的字符、汉字、数字等。在 AU3 中一般字符型数据都由英文的双引号 (") 或者英文的单引号 (') 包括，未参与数值运算和运算的数字型数据也可看作字符型数据(个人观点，仅供参考)。下面简单列出一些字符型数据

"欢迎光临 Au3 爱好者联盟"

"1234"

3*2

3

布尔值

布尔值就是“真”和“假”这两位。真的就是 true，假的就是 false。在二进制中一般用 1 来表示真，而用 0 来表示 false。如果您认真看帮助的话，您会发现，很多函数都是有返回值的，而且大多数是 1 表示函数执行成功而用 0 来表示函数执行失败或者与预期值不一致。当然，布尔值的功能不只于此，它还可以用来作为一个“开关”，这种用法对于一些小程序是很实用的。下面的代码只是示例，不懂没有关系，以后会慢慢讲到。

打开 AU3 自带的 SCITE 编辑器，输入以下内容：

```
#include <GUISConstants.au3>
#include <ButtonConstants.au3>
#include <GUISConstantsEx.au3>
#include <WindowsConstants.au3>
Dim $Form1,$Button1,$Button2,$cddrive
Global $status=True
$cddrive=DriveGetDrive("cdrom")
$Form1 = GUICreate("光驱开关", 255, 83, 192, 114)
$Button1 = GUICtrlCreateButton("光驱弹出/关闭", 8, 24, 91, 25)
$Button2 = GUICtrlCreateButton("退出", 112, 24, 75, 25)
GUISetState(@SW_SHOW)
While 1
    $nMsg = GUIGetMsg()
    Switch $nMsg
        Case $GUI_EVENT_CLOSE
            Exit
```

```

Case $Button1
    If $status=True Then
        CDTray($cddrive[1],"open")
        $status=False
    ElseIf $status=False Then
        CDTray($cddrive[1],"closed")
        $status=True
    Else
        MsgBox(0,"AU3 爱好者联盟提示你","程序出错，您可能没有光驱。")
    EndIf

Case $Button2
    Exit
EndSwitch
WEnd

```

保存，运行之。呵呵。希望大伙能够喜欢。

判断与循环之判断

对于我们人来说，判断就是根据由一些客观现象结合自身的道德、理智及自身利益来对一件事情进行的主观感知。对于一个想很好地活在这个世界的人来说，良好的判断是非险恶的能力是必须的。同样地，对于一个程序来说，一个良好的判断机制也是决定其成败与否的要素之一。在 AU3，判断主要有以下两种：

If 判断

select 与 switch 判断

接下来，我们就分别讲讲它们是怎样的。

If 判断

if 判断的基本结构是 if...then...endif,意思是“如果..那么...”，懂点英语的兄弟可以看看我这个句子 **If it rains ,I'll not attend the meeting**.逻辑结构是

中文表示

如果 下雨

<-----等价于----->

那么 我不去参加会议

脚本语言表示

if it rain then

I'll not attend the meeting

endif

实际上 if...then..endif 这个结构是隐含了一个对布尔值的引用的，例如下面这两个代码是等价的。

```
If 2=2 then
```

```
MsgBox(0,"您好","您的命题正确")
```

```
EndIf
```

```
Dim $a,$b,$c
```

```
$a=2
```

```
$b=2
```

```
$c=($a=$b)
```

```
If $c=True Then
```

```
MsgBox(0,"您好","您的命题正确")
```

```
EndIf
```

通常情况下，if..then..endif 在一个语句中是要写全的，否则会报错，我们试试把上面例子中的 endif 删掉，运行该脚本，会得到图四的提示



图四

但是，当 if..then..endif 之后的 then 只有一个动作或者函数且与 if..then..写在同一行的时候，endif 可以省略。如上例中的例子写成如下格式，运行却是正确的。

```
Dim $a,$b,$c
$a=2
$b=2
$c=($a=$b)
If $c=True Then MsgBox(0,"您好","您的命题正确")
```



图五

当然，if..then..endif 这个判断结构只能用来判断一种情况，要是要判断与这种情况的相反情形，就可以用到 if..then..else..endif 这个结构。这个结构呢，咱也给个英文句子（不是为了炫耀咱的英文啊，我英文也不好的）：**Why don't you come a little earlier,everybody else has come.**意思是“你为什么不早点来啊，其他人都来了”，仔细体会这个句子，else 是带一种排除的意味在里面的。在这个英语句子中，是说的“除了'你之外的所有人”，在 if..then..else..endif 这个结构中，则是说的排除 if 所说情况之外的情形。看看下面两段代码，它们是基本等价的：

代码一

```
Dim $a,$b
$a=InputBox("$a 与$b 值比较","请在这里输入一个$a 值,$b 已经被赋值为 10")
$b=10
If $a>$b Then
    MsgBox(0,"$a 与$b 值比较","$a>$b")
Else
    MsgBox(0,"$a 与$b 值比较","$a<$b 或者$a=$b 或者输入的$b 值无法与$a 进行比较或者您没有输入")
EndIf
```

代码二

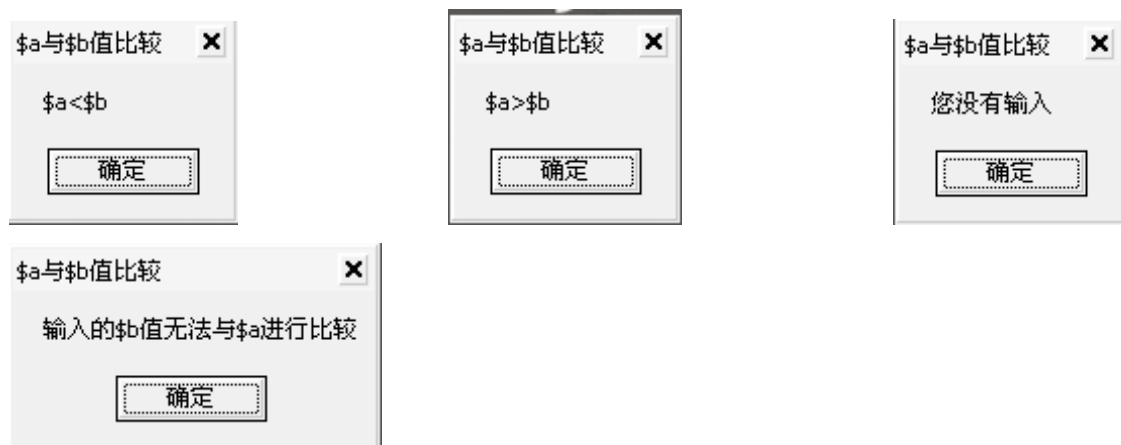
```
Dim $a,$b
```

```

$a=InputBox("$a 与$b 值比较","请在这里输入一个$a 值,$b 已经被赋值为 10")
$b=10
If StringIsDigit($a)=1 And $a>$b Then MsgBox(0,"$a 与$b 值比较","$a>$b")
If StringIsDigit($a)=1 And $a=$b Then MsgBox(0,"$a 与$b 值比较","$a=$b")
If StringIsDigit($a)=1 And $a<$b Then MsgBox(0,"$a 与$b 值比较","$a<$b")
If $a="" Then MsgBox(0,"$a 与$b 值比较","您没有输入")
If StringIsDigit($a)=0 And $a<>"" Then MsgBox(0,"$a 与$b 值比较","输入的$b 值无法与$a 进行比较")

```

您会发现，前面那个代码简洁了很多。呵呵，这就是这个 Else 的作用。运行效果如图六：



图六

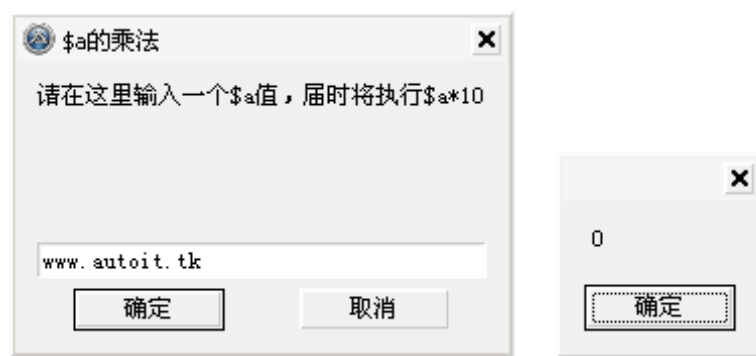
PS:其实并非字符串无法与数字型数据进行大小比较，只是我们这里为了方便理解，要求输入的必须为数字。貌似经过我测试，任何非数字的字符串的数据在参与数值运算的时候，值都是 0。简单一个例子：

```

Dim $a,$b
$a=InputBox("$a 的乘法","请在这里输入一个$a 值，届时将执行$a*10")
MsgBox(0,"",$a*10)

```

运行后的结果如图七



图七

加了一个 else，判断就那么强了，那么现在咱们现在请出 if 判断家族中最牛的 if..then..elseif..then..else..endif 同学。这种结构呢，可以说是前面两者的杂合，因为它兼具两者的优点，而且判断更加准确。看看下面的例子。

```

Dim $a,$b
$a=InputBox("$a 与$b 值比较","请在这里输入一个$a 值,$b 已经被赋值为 10")

```

```

$b=10
If StringIsDigit($a)=1 And $a>$b Then
    MsgBox(0,"$a 与$b 值比较","$a>$b")
ElseIf StringIsDigit($a)=1 And $a<$b Then
    MsgBox(0,"$a 与$b 值比较","$a<$b")
ElseIf StringIsDigit($a)=1 And $a=$b Then
    MsgBox(0,"$a 与$b 值比较","$a=$b")
Else
    MsgBox(0,"$a 与$b 值比较","程序出错，您没有输入或者您输入的不是数字")
EndIf

```

select 与 switch 判断

着俩判断有点像孪生兄弟，所以把他们凑一块介绍给大家了。他们分别是 select...case 和 switch...case。select..case 和上面的 if 系列判断是十分类似的，但通常用于判断多条件，这样会比 if 清晰很多。如下面这个例子：

If 结构

```

Dim $num=InputBox("AU3 爱好者联盟提示您","请您输入一个数字")
If Mod($num,2)=0 Then
    MsgBox(0,"您好","你输入的是一个偶数")
ElseIf Mod($num,3)=0 Then
    MsgBox(0,"您好","你输入的这个数可以被三整除")
ElseIf Mod($num,5)=0 Then
    MsgBox(0,"您好","你输入的这个数可以被五整除")
Else
    MsgBox(0,"呵呵","和程序预期不同")
EndIf

```

Select 结构

```

Dim $num
$num=InputBox("AU3 爱好者联盟提示您","请您输入一个数字")
Select
Case Mod($num,2)=0
    MsgBox(0,"您好","你输入的是一个偶数")
Case Mod($num,3)=0
    MsgBox(0,"您好","你输入的这个数可以被三整除")
Case Mod($num,5)=0
    MsgBox(0,"您好","你输入的这个数可以被五整除")
Case Mod($num,2)<>0 And Mod($num,3)<>0 And Case Mod($num,5)<>0
    MsgBox(0,"呵呵","和程序预期不同")
EndSelect

```

但是我们将 if 结构那个代码改写成这样，就能实现与上面代码不同的功能。

If 结构

```

Dim $num=InputBox("AU3 爱好者联盟提示您","请您输入一个数字")
If Mod($num,2)=0 Then MsgBox(0,"您好","你输入的是一个偶数")
If Mod($num,3)=0 Then MsgBox(0,"您好","你输入的这个数可以被三整除")

```



```
If Mod($num,5)=0 Then MsgBox(0,"您好","你输入的这个数可以被五整除")
If Mod($num,2)<>0 And Mod($num,3)<>0 And Mod($num,5)<>0 Then MsgBox(0,"呵呵","和程序预期不同")
```

Select 结构

```
Dim $num
$num=InputBox("AU3 爱好者联盟提示您","请您输入一个数字")
Select
Case Mod($num,2)=0
    MsgBox(0,"您好","你输入的是一个偶数")
EndSelect
Select
Case Mod($num,3)=0
    MsgBox(0,"您好","你输入的这个数可以被三整除")
EndSelect
Select
Case Mod($num,5)=0
    MsgBox(0,"您好","你输入的这个数可以被五整除")
EndSelect
Select
Case Mod($num,2)<>0 And Mod($num,3)<>0 And Mod($num,5)<>0
    MsgBox(0,"呵呵","和程序预期不同")
EndSelect
```

呵呵，看明白没，要执行几次判断就写几次完整的判断结构，不过在以后的 GUI 事件编程中，建议不要使用后者嵌套在原有的 select 结构中，那样会很混乱的，建议使用 if。呵呵，个人见解，仅供参考。Switch 的用法和 select 差不多，就不多讲了。呵呵。

[注]在 Select 或者 Switch 块中，执行完一个语句之后，其他语句将被忽略，此时可以使用 ContinueCase。在一个 Select 或者 Switch 块中，一个 case 执行结束将继续执行下一个 Case 的表达式(语句)。执行 ContinueCase 将告诉 AutoIt 停止执行当前的 case,并开始执行下一个 case.尝试在一个 Select 或者 Switch 结构外执行 ContinueCase 将会得到一个错误。帮助中的例子：

```
$msg = ""
$szName = InputBox(Default, "请输入一个单词", "", " M", Default, Default, Default, Default, 10)
Switch @error
Case 2
    $msg = "超时"
    ContinueCase
Case 1; 继续上一 Case 事件
    $msg &= "取消"
Case 0
    Switch $szName
    Case "a", "e", "i", "o", "u"
        $msg = "这是元音字母"
    Case "QP"
        $msg = "数学"
    Case "Q" to "QZ"
```

```

    $msg = "自然科学"
Case Else
    $msg = "其它"
EndSwitch
Case Else
    $msg = "出现了非常可怕的错误."
EndSwitch
MsgBox(0, Default, $msg)

```

判断与循环之循环

AU3 里面的几种循环结构

For...Next

While...WEnd

Do...Until

For...In...Next

啥叫循环喃？循环就是在满足一定条件的时候不停地工作、执行相同的动作，但是当条件不满足的时候，就立马罢工，退出来、不干！这就好比咱找了个工作，老板给的工资也挺不错的，然后我们就把工资当做在这工作的条件，哪一天老板说不给发工资了，咱就不干活了，甚至辞职了。（这里不涉及任何就业观之类的东西，只是个简单的比喻而已，好事者请飘过。）循环有两种，死循环和“活循环（我们讲的这些啦，好像没有活循环这个，我发明的？）”，所谓死循环就是一直那样干，直到终结，话说我们人生就是在死循环啊，不断地执行吃饭、喝水的动作，直到心脏停止跳动。电脑呢，遇到有死循环的程序，就会一直执行，直到 down 机为止。记得批处理里面有个不很出名的死循环，代码很简单但是巨强大！

start

%o

运行之后，直接弹出 N 多命令提示符，直到死机（大家不要去试哦，否者后果自负！），呵呵，由此可见死循环是很 NB 的，危害巨大，因此我们在编程的时候，要尽量避免死循环。好了，我们切入正题！

【1】For...Next

这个循环又叫定长(值)循环，主要用于知道执行次数的循环。其基本结构为：

```
For <变量> = <开始> To <停止> [Step <步进值>]
```

```
语句
```

```
...
```

```
Next
```

【注】“开始”指的是你变量的初值 “停止”指的是你变量的终值。

照它的结构，写个例子，反复弹出三次对话框：

```
Dim $i
```

```
For $i=1 To 3 Step 1
```

```
    MsgBox(0,"For,,next 的例子 1","这是第"&$i&"次显示这个对话框，还有"&3-$i&"次窗口将自动关闭")
```

```
Next
```

说明下，那个 step 在开始学的时候可能有点不明白，那么它是啥呢。做个简单地比喻吧，好比说给你十袋大米让你搬到学校大门口，你一次能扛两袋别人只能扛一袋，你扛五次就把大米扛完了，别人却需要扛十袋。在 for..NEXT 这个结构里面，变量初值与终值的差+1 就

是大米，步进值就是你一次扛大米的袋数。举个例子

```
Dim $i
For $i=1 To 6 Step 1
    MsgBox(0,"For,,next 的例子 2","呵呵")
```

```
Next
```

弹出六次对话框

```
Dim $i
```

```
For $i=1 To 6 Step 2
```

```
    MsgBox(0,"For,,next 的例子 2","呵呵")
```

```
Next
```

弹出三次对话框

最后嘛，用我们所学到的写个倒计时的东东。

```
Dim $i
```

```
For $i=10 To 0 Step -1
```

```
    TrayTip("for next 模拟倒计时","正在倒计时，还有"&$i&"秒后程序退出",100)
```

```
    Sleep(1000)
```

```
Next
```

步进值可以是负数哦！呵呵！

【2】While...WEnd

这个循环结构让我想起大街上每年秋季的时候，商店打出招牌低价打折卖夏装的事情。这个循环结构是用来表示，当满足某条件时，一直执行某动作。把上面那个例子用这个循环结构写出来就是(不能执行，只是思路而已)：

```
while 季节=秋季
```

```
    低价打折卖夏装
```

```
wend
```

简单解释下，执行该循环时，循环会先检查季节是否为秋季，为秋季则执行动作，否则不执行。看看帮助文件里面的这个简单的小例子：

```
Dim $i
```

```
$i = 0
```

```
While $i <= 10
```

```
    MsgBox(0, "$i 的值为:", $i)
```

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

```
WEnd
```

程序先检查\$I 是否<=10 因为\$i 预先赋值为 0，所以会执行下去。\$i = \$i + 1 的作用就是每次给\$i 增加 1，最后当\$i=10 执行了 \$i = \$i + 1 后\$i=11,不满足 \$i <= 10，循环停止!大家明白了吧。呵呵。

【3】Do...Until

看到这个结构，想到小学时代造的一个句(那时候造句很厉害哦，呵呵，臭美下)“我一直做作业直到妈妈让我吃饭”，这句话用这个循环结构表达就是

```
do
```

```
    做作业
```

```
until
```

```
    妈妈让我吃饭
```

当满足 until 后的条件时，do 后动作直接停止。举个简单例子：

Do

```
$num=InputBox("提示","请输入密码 123，否则程序一直弹出对话框")
```

Until \$num="123"

【4】 For...In...Next

（暂时不讲，因为平时没有咋用到）

二、常用函数串讲

在 AU3 中，函数是最精彩的部分，有很多好玩的东东，让我带大家一起去函数的是假玩玩吧！

【1】简单的对话框--msgbox 函数

这个函数啊，之前的课程中，我们已经讲到了，现在我们来系统地讲解下。咱们打开帮助文件。标志呢，就是用来控制对话框所显示按钮的，我们先前的标志是 0，只显示“确定”按钮，我们看下标志是 1 的时候是啥情况，是不是和它上面所对应的“确定”和“取消”呢？输入下面脚本

```
msgbox(1,"msgbox 例子 1","显示的是确定和取消两个按钮")
```

呵呵，是不是出现“确定”和“取消”两个按钮了？其他的都差不多，都是只要把标志值换成对应的值就可以了，另外呢，这些标志值是可以相加的，得到的是他们按钮的整合，也就是说标志值相加之后，他们相加的两个值的按钮会一起显示在对话框上面（如果他们相加后的值是已定义的标志值，那么就会显示相对应的标志值，但如果和是未定义的标志值，那么将不会有任何显示，参考后面的表格一）。看看下面一个小例子：

```
msgbox(1+2,"msgbox 例子 2","标志相加显示 N 多按钮")
```

有显示

```
msgbox(3+4,"msgbox 例子 3","标志相加显示 N 多按钮")
```

没有显示，呵呵。因为 3+4=7 7 在标志值里面不存在

接下来就说对话框的图标啦。一般的，我们系统的对话框图标有以下几种：

使用方法就是在标志值那使用按钮列表值+图标列表值。

另外呢，我们还可以设置默认的按钮，方法和上面一样，把“相应的默认按钮”下面的值加过去就 ok 了。注意看，第二个按钮是“默认按钮”

最后呢，要说到 msgbox 的返回值，啥叫返回值呢，我个人的理解就是返回值就是用来说明操作成功与否或者都执行了哪些操作。譬如，在 msgbox 中，用户如果按了“确定”按钮，那么它的返回值将是 1，“取消”按钮的返回值是 2（其他的参考表格二），我们一般是通过返回值来与用户进行交互的。

```
If MsgBox(1,"MsgBox 的返回值测试","随便点一个按钮，我知道你点的是啥")=1 Then
```

```
MsgBox(0,"呵呵","你点的确定按钮")
```

```
Else
```

```
MsgBox(0,"呵呵","你点的是取消按钮")
```

```
EndIf
```

也可以这样写

```
Dim $anniu
```

```
$anniu=MsgBox(1,"MsgBox 的返回值测试","随便点一个按钮，我知道你点的是啥")
```

```
If $anniu=1 Then
```

```
MsgBox(0,"呵呵","你点的确定按钮")
```

```
Else
```

```
MsgBox(0,"呵呵","你点的是取消按钮")
```

```
EndIf
```

个人感觉后面那个要整洁点，推荐使用这种方式

dim \$var=要取返回值的函数

条件选择语句

.....

一般的取返回值都是用的这种方式，呵呵。

表格一

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

表格二

按下的按钮(具体显示
的名字取决于操作系 返回值
统的语言版本)

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

【2】显示一个输入框--inputbox 函数

明白英文的同学应该知道什么叫“input”？呵呵，对了，就是输入的意思。Inputbox 这个函数就是提供一个用来输入的对话框的东东。它的基本语法为

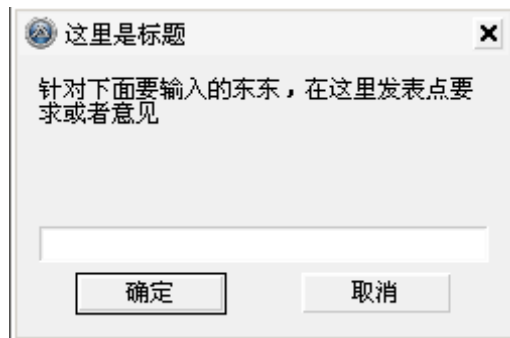
Inputbox("提示输入对话框的标题","输入框上面的提示")

写这样一行代码

InputBox("这里是标题","针对下面要输入的东东，在这里发表点要求或者意见")

运行下的效果如图八

图八



但是，咱们输入了一点，啥也没有了。呵呵，还记得我们之前学的那个 messagebox 函数吗？咱们让它来显示我们输入的内容。代码：

Dim \$text

\$text=InputBox("虫子樱桃提示您","请您输入要显示的文字")

If \$text<>""Then

MsgBox(0,"哈哈","您输入的是"&@LF&\$text)

Else

MsgBox(16,"警告","程序出错或者您木有输入")

EndIf

【3】关机函数 shutdown

这个函数和 CMD 中的 shutdown 是一样的，其函数参数如下

C:\Documents and Settings\Administrator>shutdown/?

用法: shutdown [-i | -l | -s | -r | -a] [-f] [-m \\computername] [-t xx] [-c "comment"] [-d up:xx:yy]

没有参数	显示此消息(与 ? 相同)
-i	显示 GUI 界面，必须是第一个选项
-l	注销(不能与选项 -m 一起使用)
-s	关闭此计算机
-r	关闭并重新启动此计算机
-a	放弃系统关机
-m \\computername	远程计算机关机/重新启动/放弃
-t xx	设置关闭的超时为 xx 秒
-c "comment"	关闭注释(最大 127 个字符)
-f	强制运行的应用程序关闭而没有警告
-d [u][p]:xx:yy	关闭原因代码
	u 是用户代码
	p 是一个计划的关闭代码

xx 是一个主要原因代码(小于 256 的正整数)
yy 是一个次要原因代码(小于 65536 的正整数)

在 AU3 中 shutdown 的具体格式是

Shutdown (代码 [, 理由])

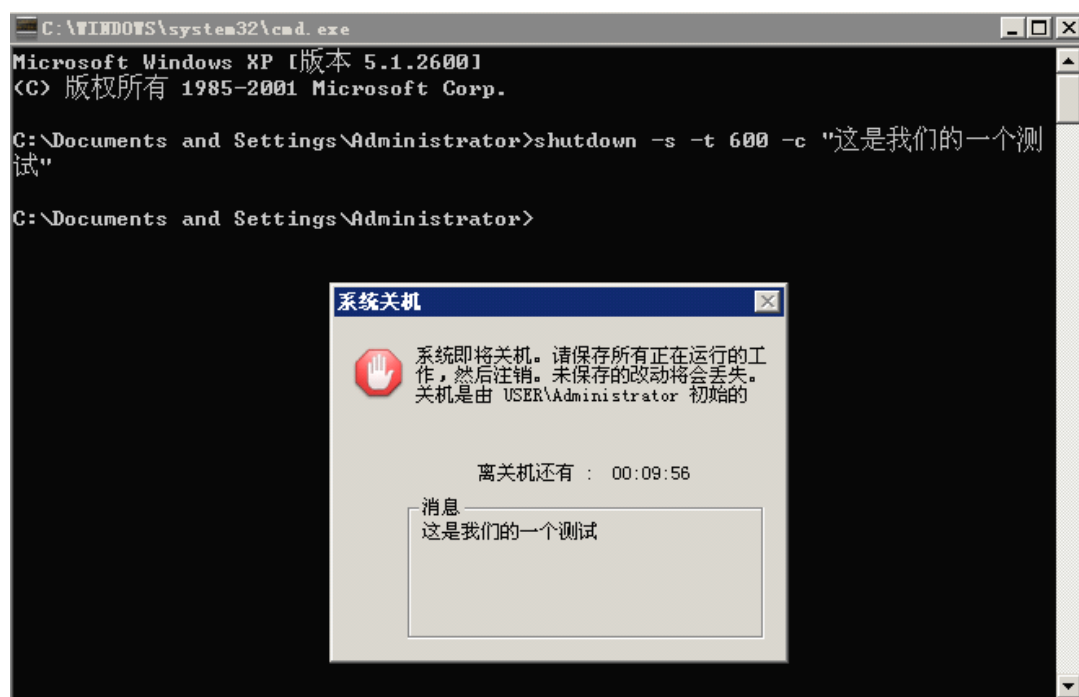
关机代码可以是下面的值:

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)

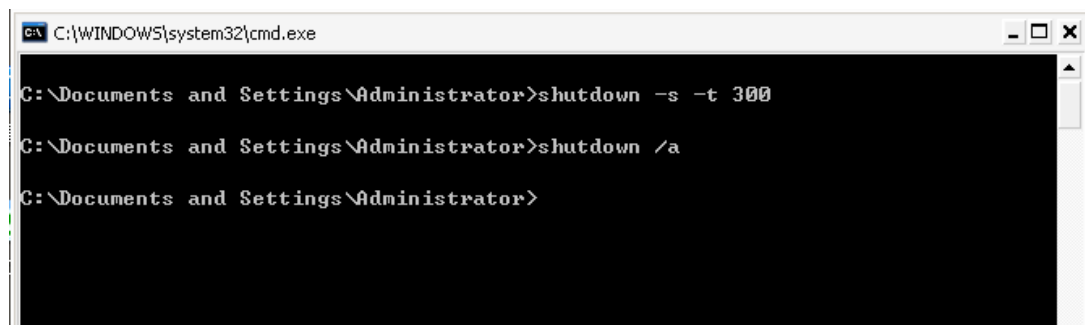
可按需把相应数值相加.比如,要关机并断电,则应指定数值 9 (shutdown + power down = 1 + 8 = 9).

若设置了其它代码则待机或休眠将被忽略.

他们都可以用来关机,呵呵,我觉得是各有千秋。CMD 命令 shutdown 可以用自身命令参数 -t 来延长关机,但是没有休眠功能。另外 CMD 中的 shutdown 命令还可以通过 -a 参数来取消未执行的关机或者重启动作,而 AU3 中的则是直接执行,并不能取消。如图九,我们设置 600 秒关机,然后取消。

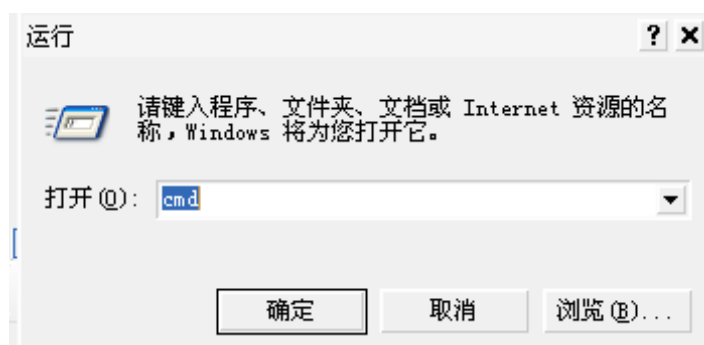


图九 1 关机



图九 2 取消关机

【注】网上有很多什么要求你输入“XX 快说你是猪”之类的恶搞脚本就是利用的 CMD 命令的 shutdown，当出现了如图九 1 的画面之后，打开运行（快捷键是“Win+R”）在运行中输入 CMD，打开命令提示符输入 shutdown /a 即可取消关机。操作如图十



图十

AU3 中的 shutdown 命令是怎么用的呢？举一个例子，譬如说，我要注销，那么我查询帮助看见注销的指令代码是 0，那么我就直接输入 shutdown(0)就注销了。其他的应该会了吧。好奇的朋友有没有发现那个叫“强制执行”的参数，呵呵，这个的意思就是你在执行对应的关机代码的时候，把要执行的关机代码加 4，就是强制执行关机的相关指令。譬如说你要执行重启，那么你输入 shutdown(2)可以关机，输入 shutdown (2+4)也就是 shutdown (6)，就执行强制关机了。

【4】执行一个程序

呵呵，接着咱们来学习下怎么用 AU3 调用一已经存在于您计算机上的程序。

首先要介绍的是 **Run** 这个函数。输入以下代码，运行下，您看到了什么？

```
Run("Notepad.exe")
```

哈哈，记事本被打开了。



图十一

那我想要执行其他程序怎么办啊？嘻嘻，只要把那个 "Notepad.exe" 换成其他程序的名字即可，譬如说我们要打开注册表编辑器，就写 `Run("regedit.exe")`。有的朋友说，我执行 QQ，那我就写 `Run("qq.exe")` 对吧，呵呵，咱们试试！写个脚本输入 `Run("qq.exe")`，运行了，没有任何反映，这个是为啥呢？因为 `regedit.exe` 和 `Notepad.exe` 的路径都是在系统环境中的，啥叫环境变量呢，环境变量简单的来说就是系统给程序的“人际关系”，当执行没有写完全路径的程序时，程序先在同级目录找，同级目录找不到就到先前安排好的“人际关系”那里去找，找到就执行。但是我们的 QQ 不是在“人际关系”之内的，所以 QQ 不会被执行。那我们要怎样通过 AU3 执行 QQ 呢？在桌面找到 QQ 的快捷方式图标，右键选择，选择“属性”，出现图十二的界面。



图十二

赋值“目标”后面的那一串字符，加上 run，写成 run（"C:\Program Files\Tencent\QQ\Bin\QQ.exe"），运行下。出现如图十三的界面



图十三

[注]环境变量的一些资料

环境变量是一个具有特定名字的对象，它包含了一个或者多个应用程序所将使用到的信息。例如 path，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 path 中指定的路径去找。用户通过设置环境变量，来更好的运行进程。

1. 环境变量的设置有几种方式？

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 其他修改方法

在 windows 操作系统中可以通过我的电脑->属性->高级，来设置系统的环境变量，然而在此设置的环境变量是否在注册表中具有对应的项呢？答案是肯定的。而在 .net 中提供了一个类来获取系统的环境变量及其值。环境变量分为两类：用户变量与系统变量，在注册表中都有对应的项。其中用户变量所在位置：HKEY_CURRENT_USER\Environment；系统变量所在位置为：\HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Session Manager\Environment。另外也可以右击我的电脑——高级——环境变量——在系统变量里有 path 选项——双击打开——在原有变量的基础上添加英文状态下的分号——然后将路径名输入即可。（切记，不要删除原先的系统变量，只要用分号隔开，然后添加）

4. 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。

5. 改变环境变量和环境变量中的值应该注意什么？

环境变量和环境变量的值不要含有空格，也不要中文，切记！

6 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准确路径。

%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名称。

%UserProfile% 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。

而今世界是个充满个性的世界，咱们登陆 QQ，怎么能不个性一把呢？接下来咱们把我们登陆 QQ 时的窗口“个性化”一下吧。这就要用到我们的 **WinSetTitle** 函数了，这个函数就是设置窗口标题的意思，用法，(*^__^*) 嘻嘻……，查帮助去。输入以下代码：

```
Run("C:\Program Files\Tencent\QQ\Bin\QQ.exe")
WinWaitActive("QQ2010")
```

WinSetTitle("QQ2010","", "AU3 爱好者联盟专用 QQ")

[注]请根据您的 QQ 安装路径来修改。WinWaitActive 函数用来暂停脚本等到指定窗口为活动时才执行。运行了，出现什么呢？看图十四

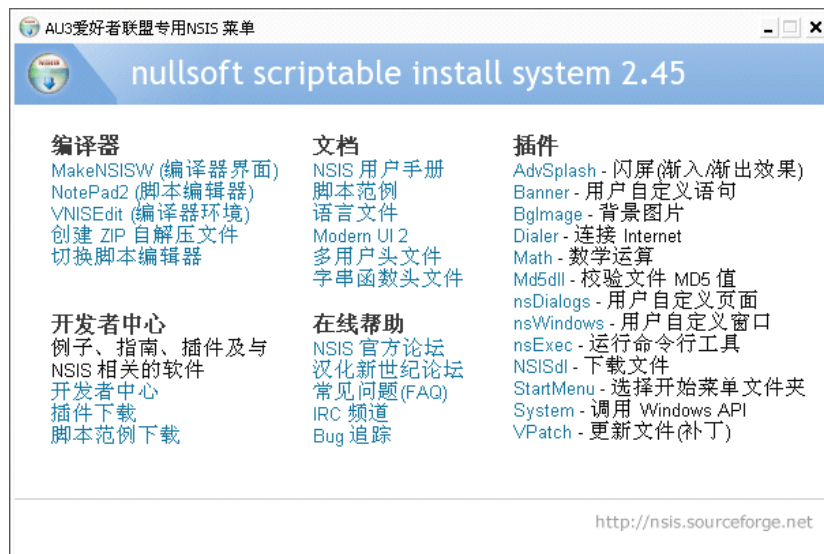


图十四

但是，我们这个脚本写得太局限性了，要是我们想把所有当前活动的窗口的标题都加上“AU3 爱好者联盟专用”该怎么写呢？我们要在这里加两个函数，**WinGetTitle** 函数用来获取窗口标题，**StringInStr** 用来检测是否 A 字符串中含有 B 字符串，同样的，这俩函数去查帮助文件吧，我即使讲也是在 ctrl+c 和 ctrl+v 而已，没有意思。代码如下

```
#include <GUIConstants.au3>
#include <GUIConstantsEx.au3>
#include <WindowsConstants.au3>
Dim $title
AdlibRegister("listen")
GUICreate("",0,0)
While 1
    $nMsg = GUIGetMsg()
    Switch $nMsg
        Case $GUI_EVENT_CLOSE
            Exit
    EndSwitch
WEnd
Func listen()
    $title=WinGetTitle("[active]")
    If StringInStr($title,"AU3 爱好者联盟专用")=0 Then
        WinSetTitle($title,"","AU3 爱好者联盟专用"&$title)
    EndIf
EndFunc
```

执行脚本后，无论您打开什么窗口，前面都会有"AU3 爱好者联盟专用"的字样，是不是很酷。如果您要隐藏右下角的托盘图标，请在脚本最前面加#NoTrayIcon 即可。效果图



图十五

标题是改了，那咱们能不能来点像 VISTA 系统的透明效果呢？我们发现帮助中函数参考--窗口管理中有个 **WinSetTrans** 函数，它就是我们要实现此功能的核心。直接在我们上面那个脚本中添加一行。（这个函数设置的透明度值是 0~255，255=不透明,0=不可见，按照自己的需求酌量修改。O(∩_∩)O~），我修改后的代码如下：

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

Dim $title
AdlibRegister("listen")
GUICreate("",0,0)
While 1
    $nMsg = GUIGetMsg()
    Switch $nMsg
        Case $GUI_EVENT_CLOSE
            Exit
    EndSwitch
WEnd
Func listen()
    $title=WinGetTitle("[active]")
    If StringInStr($title,"AU3 爱好者联盟专用")=0 Then
        WinSetTitle($title,"","AU3 爱好者联盟专用"&$title)
        WinSetTrans("AU3 爱好者联盟专用"&$title,"",170)
    EndIf
EndFunc
```

运行后的效果如图十六



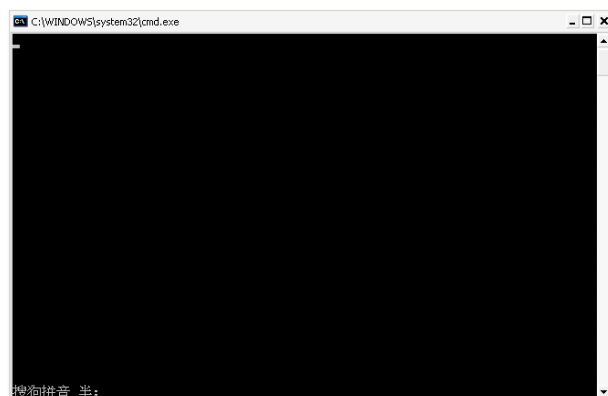
图十六

讲了这么多，差点漏掉了一点最重要的，(*^__^*) 嘻嘻……，那就是-----用 `run` 来隐藏运行程序！我们这里以运行 `cmd` 命令为例，因为这个的应用最典型嘛！一个简单的脚本，输出 C 盘的文件目录到 D 盘 `list.txt` 文件。

不隐藏版

```
Run(@ComSpec&" /c tree/f c:\>d:\list.txt")
```

运行后出现黑框，如图十七



图十七

隐藏版

```
Run(@ComSpec&" /c tree/f c:\>d:\list.txt","",@SW_HIDE,"")
```

执行和上面那个效果一样，但是不显示黑框。

除了 `Run` 之外，还有个 **ShellExecute** 可以用来执行程序，不过它较 `Run` 更为“通吃”一点，为什么呢。在任意位置建一个 `txt` 文件，譬如我们在 D 盘根目录下建立。分别执行下面两行脚本：

```
run("d:\test.txt")
```

```
ShellExecute("d:\test.txt")
```

您会发现执行第一行代码是没有任何反应的，而第二行代码执行后则用记事本打开了在 D 盘的 `test.txt` 的这个文件。这是为什么呢？这是因为 `shellexecute` 执行程序时会调用与程序或者文件相关联的程式来完成操作。而 `run` 只能直接执行 `exe` 文件，执行 `msi` 程序文件或者其他文件时，需要在前面加上执行的关联程序。如果我们把上面的第一行脚本改成如下样子，

您会发现 D:\test.txt 这个文件被打开了。

```
run("notepad.exe d:\test.txt")
```

[注]run 在执行 msi 执行程序时需要写成 Run("msiexec xx.msi"),如果您要运行一个程序或者打开文件并需要等待打开的程序或者文件关闭才运行后面的程序或执行后面的动作, 请用 **Runwait** 或者 **shellexecutewait**.

【5】将窗口玩转到底—其他几个与窗体相关的函数

您是不是个偶尔想搞点恶作剧的人呢? 呵呵, 下面咱们用 AU3 的一个函数 **WinSetState** 来搞个小恶作剧。Winsetstate 这个函数用来设置窗口的状态, 在帮助中, 该函数可设置的状态有七种, 分别是@SW_HIDE (隐藏窗口), @SW_SHOW(显示以前隐藏的窗口), @SW_MINIMIZE(最小化窗口), @SW_MAXIMIZE(最大化窗口), @SW_RESTORE(撤销窗口的最小化或最大化状态), @SW_DISABLE(禁用窗口), @SW_ENABLE(使窗口可用), 函数的格式很简单. WinSetState ("窗口标题", "窗口文本", 状态)。好啦! 看代码, 因为这是个演示, 所以只是恶搞几秒钟, 几秒钟后自动恢复! **尽管如此, 也存在未知的风险, 为避免您的电脑出现各种奇怪的毛病, 请谨慎使用下列代码!**

```
WinSetState("Program Manager","",@SW_HIDE)
```

```
Sleep(8000)
```

```
WinSetState("Program Manager","",@SW_SHOW)
```

[注]运行了上面的这个脚本, 桌面的所有图标会消失, 8 秒钟后自动恢复。

```
WinSetState("Program Manager","",@SW_DISABLE)
```

```
Sleep(8000)
```

```
WinSetState("Program Manager","",@SW_ENABLE)
```

[注]运行了上面的这个脚本, 将不能桌面上的项目进行任何操作, 8 秒钟后自动恢复。当然, 您也可以按 ctrl+alt+del 打开任务管理器, 结束 explorer.exe, 然后选择“文件-新建(任务)”, 输入“explorer.exe”即可。下面呢, 我们来做一个闪烁窗口文字的效果, O(∩_∩)O~, 用到的函数是 **WinFlash**, 具体的用法参考帮助文档。

```
Run("notepad.exe")
```

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

```
WinFlash("[CLASS:Notepad]", "", 4, 500)
```

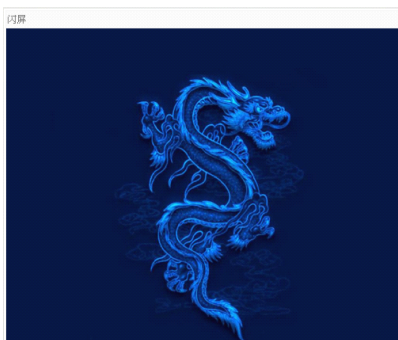
【6】闪屏没问题

不知道朋友们有没有见过有的软件, 是带一张闪屏的, 很是漂亮, 接下来咱们就用 AU3 中的函数 **SplashImageOn** 来做一个! 该函数的基本格式是 SplashImageOn ("标题", "图片文件的路径"), 举个简单的例子:

```
SplashImageOn("闪屏","E:\test.jpg")
```

```
Sleep(5000)
```

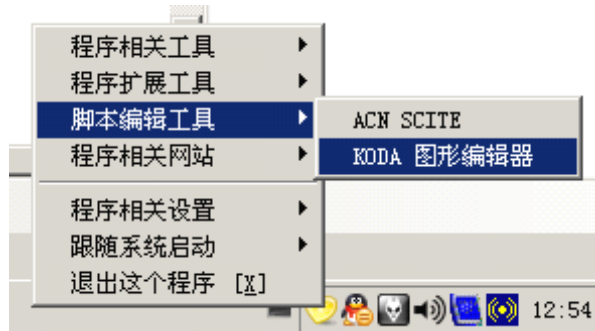
运行效果如图十八 (当然条件是您 E 盘根目录存在一张名为 test 的 jpg 格式图片, 您也可以吧路径换成您自己想要设置的图片路径。该函数支持 BMP,GIF,或 JPG 三种格式的图片。)



图十八

【7】窗口是这样练成的

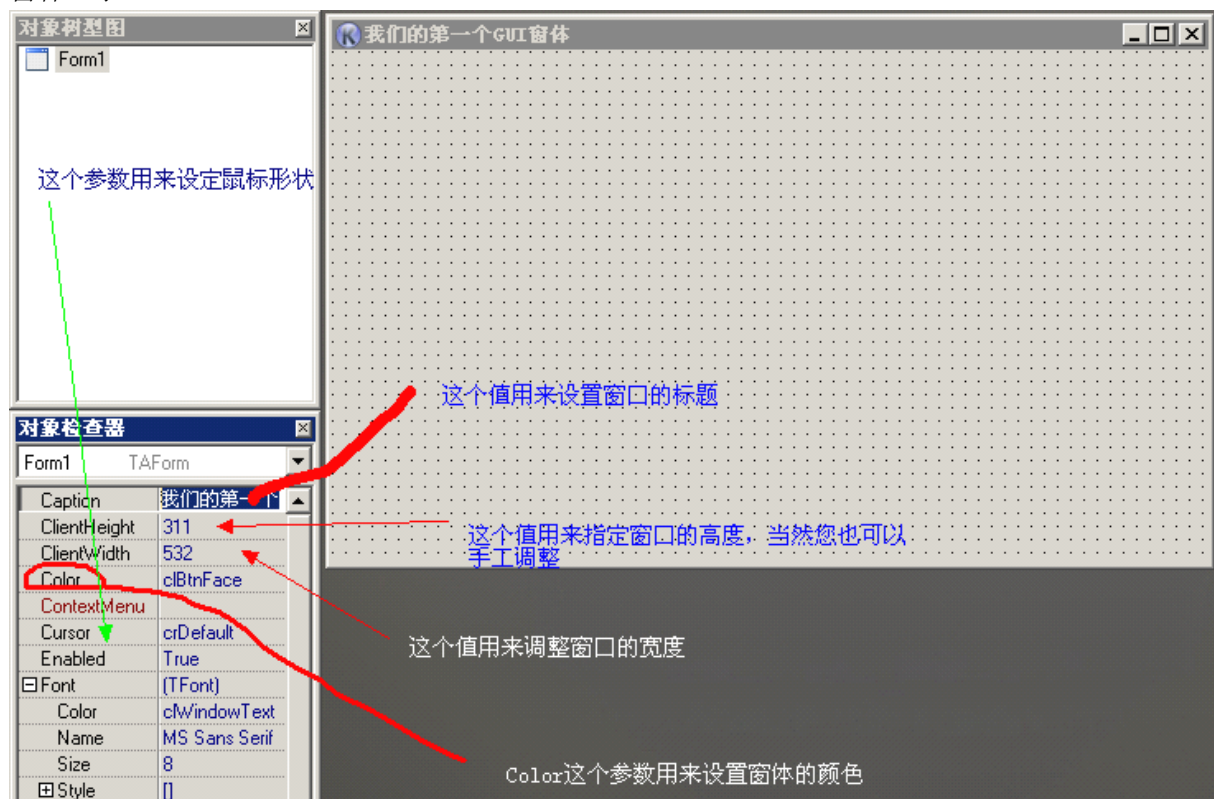
现在我们开始说 GUI，这是我觉得 AU3 最牛 X 的地方，因为我没有见到过有哪一种脚本语言可以像 AU3 一样编写出丰富的 GUI（就是图形用户界面，其英文是 Graphical User Interface）。好吧，其他的就不吹牛了，直接开始写脚本。我们这里是用 KODA 编辑器来做的，我刚学 GUI 的时候，不知道这个东东，走了很长的歪路，不过呢，建议你在看这节内容的时候，先看下帮助文档，至少知道那个函数是啥意思，遇到问题也好解决不是。



图十九 打开 KODA 编辑器

打开之后，该编辑器默认就为我们建立一个 GUI 窗体，下面我会以图的方式来具体的操作：

选中窗体，然后在左边找到 Form1 的 Caption 选项，输入您要自定义的窗口标题，譬如说我们这里输入“我们的第一个 GUI 窗体”，您会发现窗体的标题已经变成“我们的第一个 GUI 窗体”了。

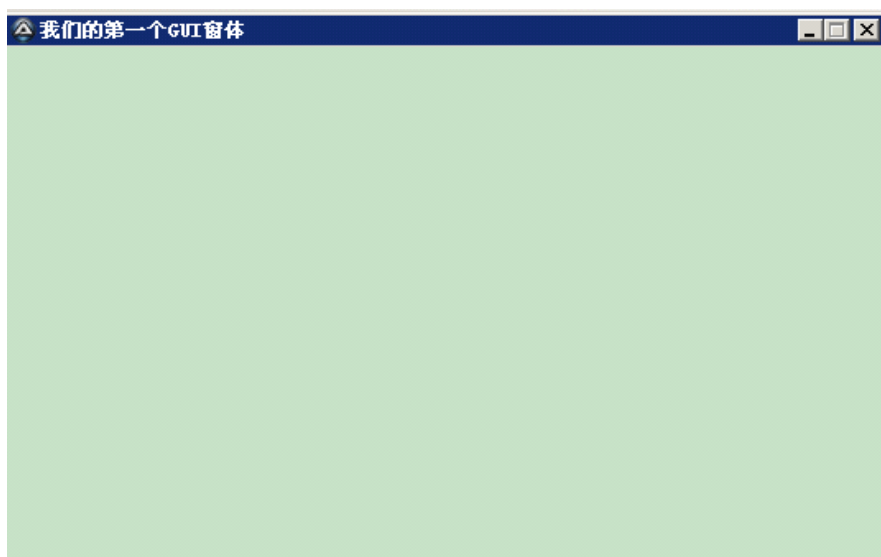


图二十 基本参数说明

设置下窗体颜色，点击右上角的“运行预览窗口”（快捷键是 F10）即可预览窗体：

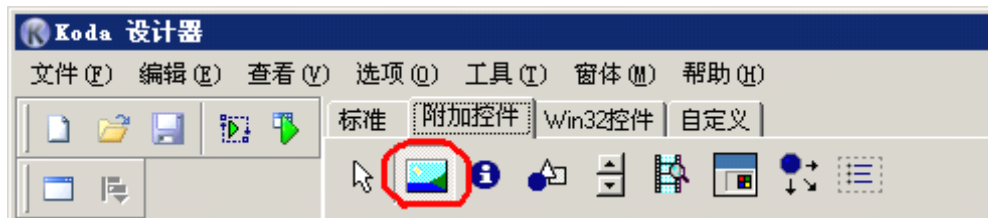


图二十一



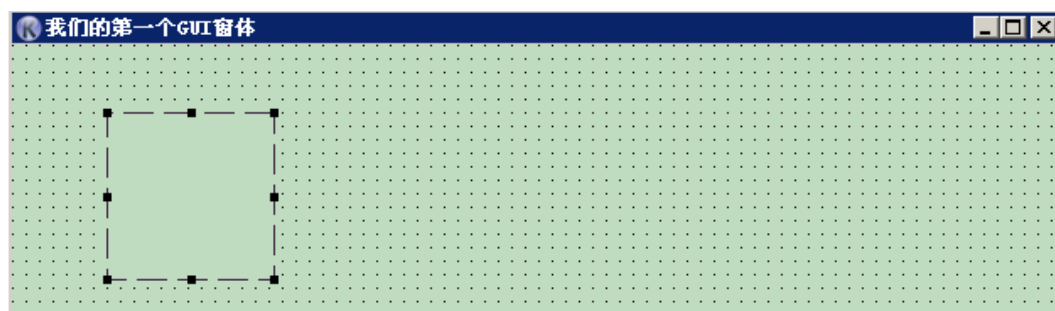
图二十二窗体预览效果

可是，现在这个界面说实话---丑陋至极，那么我们怎么用一张图片来作为它的背景呢？窗体的属性这里并没有提供像“autorun”或者“Apm”这类的光盘界面制作软件一样的用一张图片作为窗体界面的选项。那么我们怎么办呢？我们可以借助创建一个 Image 框来实现。在 Kode 编辑器上面找到如图的项目：



图二十三

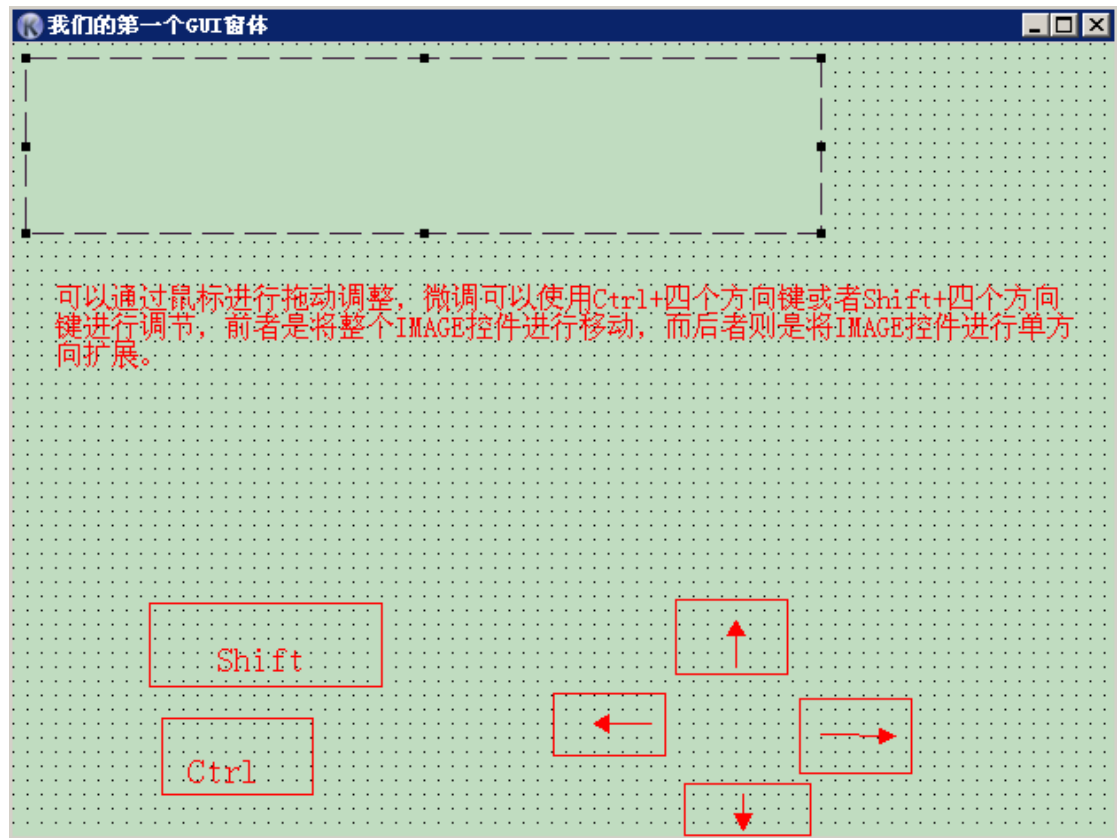
然后在我们之前创建的窗体上单击，出现一个小框，




图二十四

根据自己的需要调节好大小：

具体说明，请看下图：



图二十五

将 IMAGE 控件调整为合适大小之后，选中，然后在左边的对象检查器找到“Picture”，并将鼠标光标移动进去，按下 **Picture** **[None]** ，开始选择您想要加入为背景的图片，选择 **载入(L)...**，选择好图片之后会有您选择图片的预览图



图二十六



图二十七

接下来我们再在界面上加一个按钮，找到如图所示的地方，单击，然后再在 GUI 界面设计那单击一下，就会出现一个名为 `button1` 的按钮，同样的，您可以像调整 `image` 那样来调整它的大小位置，也可以如调整框体的属性那样设置它的按钮显示 (`caption`)、颜色 (`color`) 等。

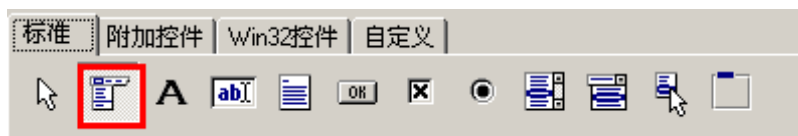


图二十八





图二十九

通过图二十八上面的其他按钮，我们可以创建其他的东西，下面，将对应图标创建的东东贴图出来，方便大家使用：



图三十

主菜单设计：

在 GUI 上添加该控件之后，只会看到这样的小块，您需要选中该小块，（该小块可以放置在 GUI 任何位置，只要不影响使用即可）并在左边的“对象检查器”中找到如下图的地方，按下，然后打开如图三十二显示为“菜单设计器”的界面。

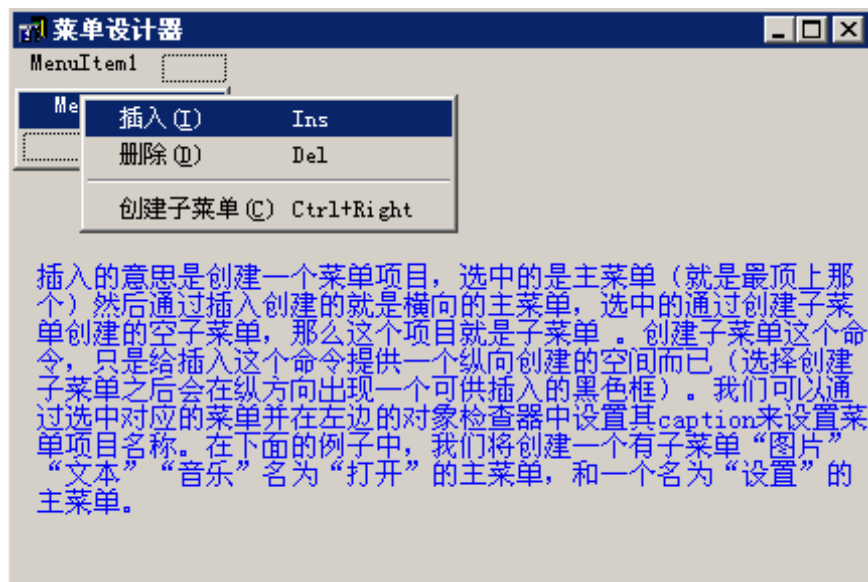


图三十一



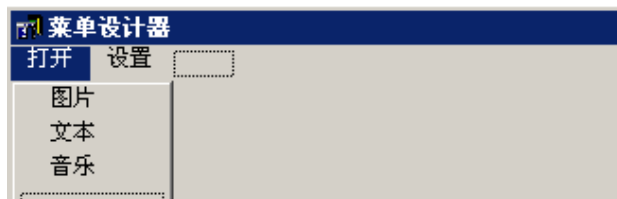
图三十二

在这个界面上会出现一个黑框，具体说明如下：



图三十三

编辑好之后，直接关闭菜单设计器即可。

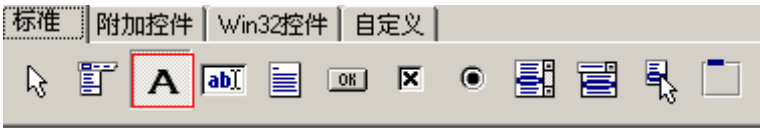


图三十四

这时您会发现，GUI 上面已经有了菜单



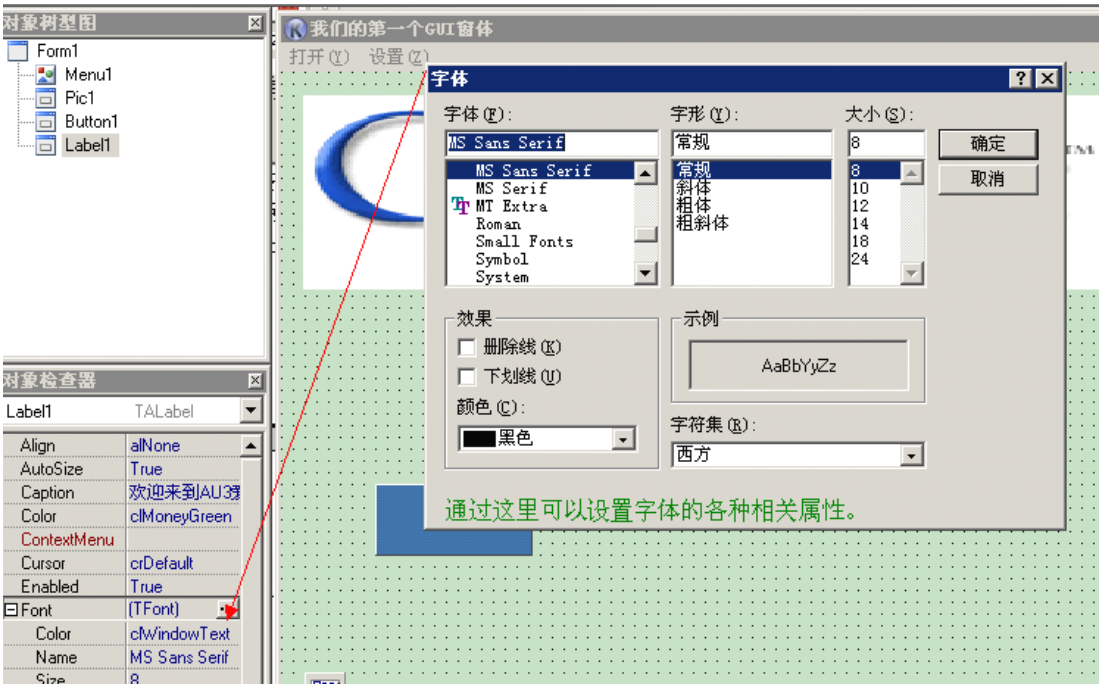
图三十五



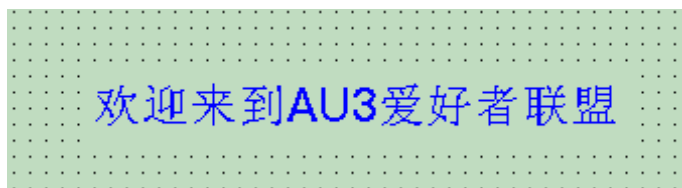
图三十六

标签:

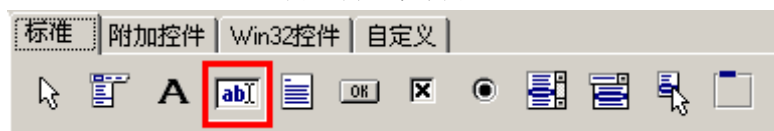
所谓标签，说简单就是显示在 GUI 上面的文字。操作方法和前面的那些控件很相似，主要是通过“对象查看器”来设置字体颜色、字体大小还有 caption 等属性，这里我们来建一个“欢迎来到 AU3 爱好者联盟”的标签，字体颜色设置为蓝色



图三十七



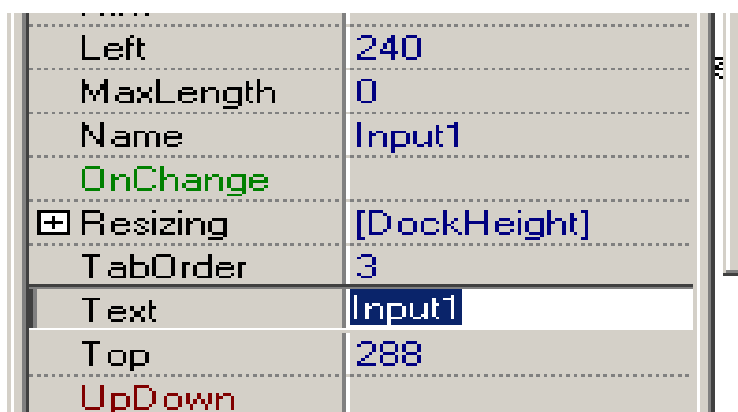
图三十八 效果图



图三十九

输入框:

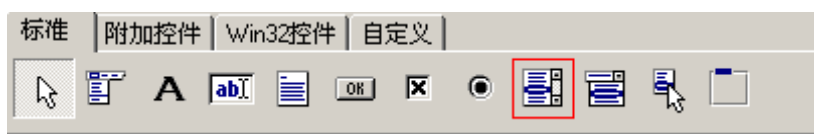
输入框用来提供给予人输入，相当于一个始终显示的 inputbox。



图四十

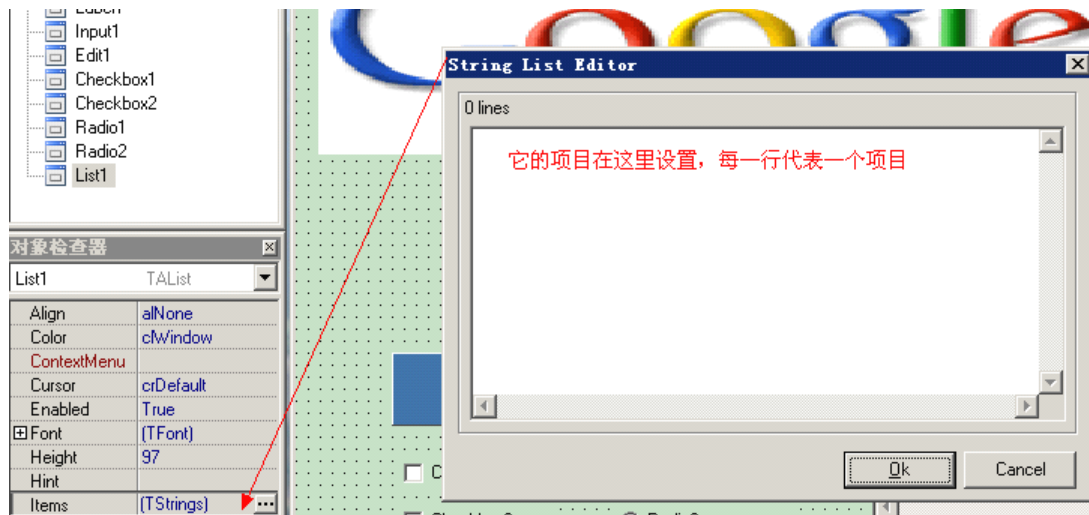
这里的这个 Input1 是用来设置输入框中默认显示的内容。如果您不想在输入框中显示，请输入空格。

后面的几个控件很简单，和前面的内容有很大雷同，大家自己琢磨下吧。只是请注意两个地方。



图四十一

也就是列表框，呵呵，注意的是下面这点。



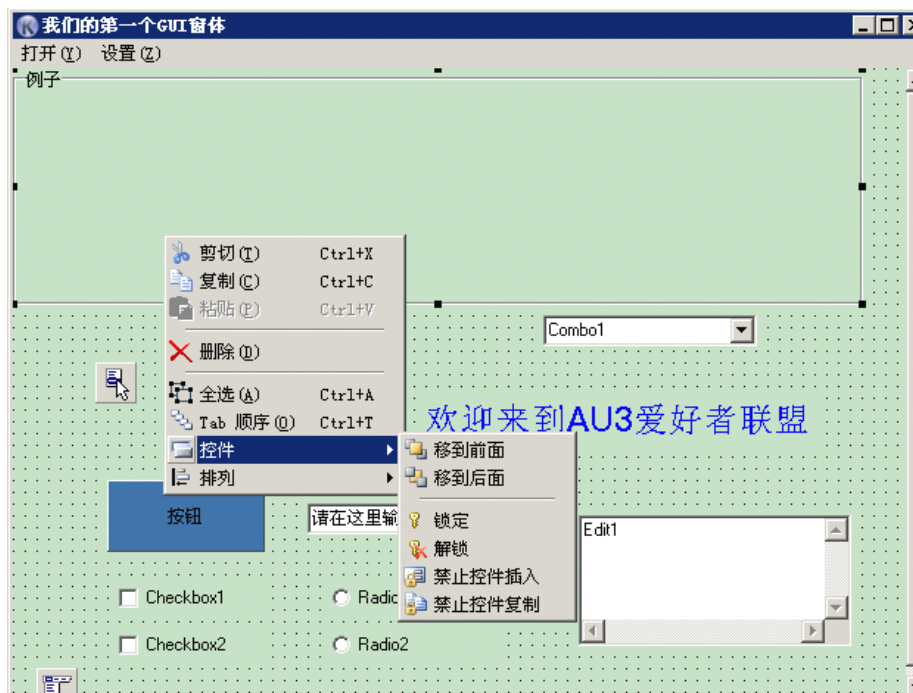
图四十二

在新建组合框即 **groupbox** 以及其他一些控件的时候，很可能会出现下面的情况---先前建立的控件被遮盖了。（一般的 **groupbox** 在设置其相应属性后在旁边空白处点击即可正常恢复，其他控件被覆盖时显示的方法见后）



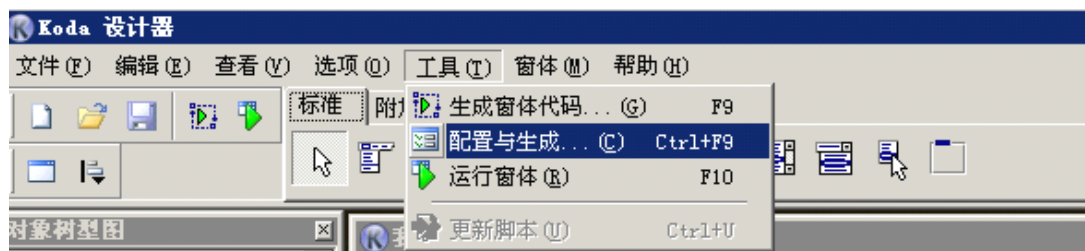
图四十三

解决方法是，选择您要调整的控件，右键选择“控件---移到后面”即可。当然在实际运用中，对于一些容易“跑路”的控件，还需要将他们设置为禁用。具体说明见紧接着的 GUI 常用函数。

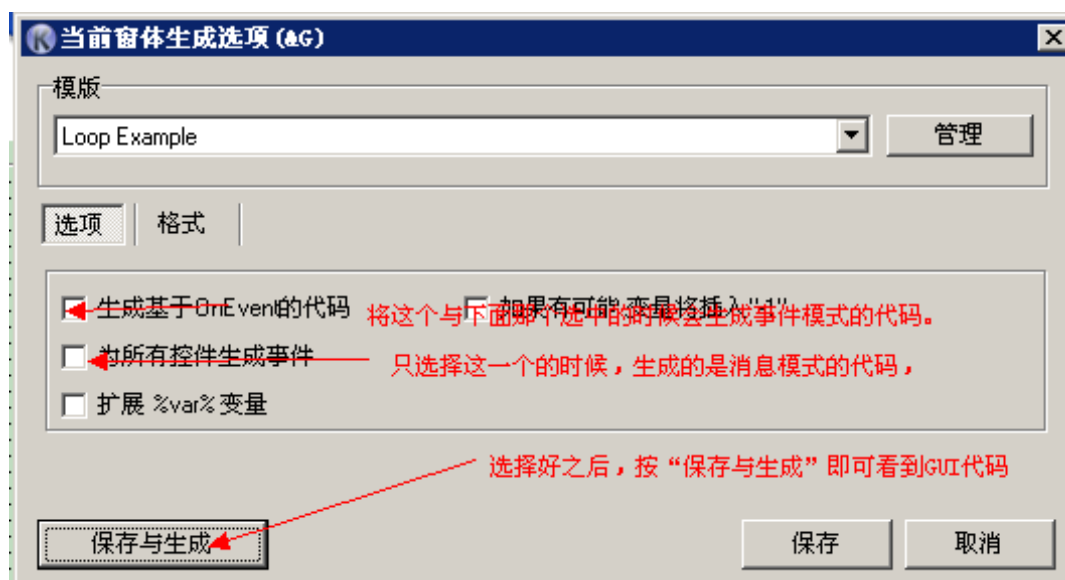


图四十四

编辑好您的 GUI 之后，我们接下来需要做的是生成代码。具体操作请看下面的图：



图四十五



图四十六

GUI 的常用函数

GUI 是英文 Graphical User Interface 的简写，翻译成中文就是图形化用户界面。之前讲了使

用 Koda 编辑器来建立 GUI，现在我们来具体讲一些与 GUI 相关的函数。

GUICreate 函数

创建一个 GUI 界面的第一步是建立一个主窗口，而这个函数正是这第一步要用到的。通过这个函数，我们可以按照自己的意愿设定窗口的大小、位置，为创建其他控件提供条件与空间。该函数的语法格式为

`GUICreate ("窗口标题" [, 宽度 [, 高度 [, 左侧 [, 顶部 [, 样式 [, 扩展样式 [, 父窗口]]]])`

和 MsgBox 灰常类似，很多 GUI 相关的函数也是有参数的。在 GUI 里面的参数主要有两类：样式与扩展样式。如果您查看一下帮助文件中关于样式的内容，您会发现有很多种方法可以用来改变和设置您的 GUI 以及它们上面的控件的样式。但是，需要提醒您的是，当您在同时使用多个样式（复合样式）的时候，就必须使用 BitOr 这个函数。即使在帮助文件中偶尔看见有用+的操作来定义复合样式，那也是预先定义您了使用 BitOr 的。具体说明如下：

为了帮助您理解为何我们要优先使用 BitOr 而非其他的那些+的操作，我们先来说说 BitOr 是什么意思？概括地说，BitOr 发生在两个或两个以上的整数值，并返回按位“或”“二进制表示”的整数。返回的结果始终是一个整数，万一参数都是零/空值，零/空值参数将不会在按位处理“或”。因此，正如帮助文件中所显示的那样，执行3按位操作（0011二进制）和6（在0110二进制）是7（二进制0111），而不是9（二进制1001），因为其结果必然是使用+运算符。要显示如何参数为空值 /零不处理，我们可以添加一个0到我们要操作的数的二进制前面。为加深理解，用帮助中那个例子来张图，(*^__^*) 嘻嘻。

十进制数	二进制数
3	0 0 1 1
6	0 1 1 0
BitOr	0 1 1 1

BitOr的意思是只要两个同位置的数不全都为0，则为1，即除了(0,0)均为1。通过对3与7二进制的观察，我们发现他们对应位除了首位全部是0外，其余位都有1，故除其首位外，其余位全部为1。

图四十七

此外，在 AU3中还有个 BitAnd，它和 BitOr 是相反的，也就是只有对应位全部为1的时候，才为1。如

`$x = BitAND(13, 7);x == 5`,因为 `1101 AND 0111 = 0101`

另外，有个问题就是，通过 GUICreate 创建的 GUI 可能会不显示出来，这是为啥呢？呵呵，因为在 GUI 创建中还需要使用 GUICtrlSetState 这个函数。如果只是单纯的设置 GUI 的属性为显示，写成 GUICtrlSetState () 或者 GUICtrlSetState (@sw_show) 是一样的。但是您要设置其他控件就要写成 GUICtrlSetState (控件 ID, 状态) 的样式。具体是怎么操作的，请看帮助文件。

创建控件

您可以往您的 GUI 上面添加很多的控件用以丰富您的 GUI。这里呢，我们只讲几个

基本的控件，其他的控件您可以通过查阅帮助文件进行学习，因为那个详细的很。要时刻相信 **GUI is simple** 好啦，切入正题。

创建一个图片控件

基本格式为 **GUICtrlCreatePic** (文件名, 左侧, 顶部 [, 宽度 [, 高度 [, 样式 [, 扩展样式]]])

如果您选择用一个图片作为背景，那么，您还需要使用 **GUICtrlSetState** 来设置其为不可操作（禁用）。这样做的好处是可以避免图片控件和其他控件间相互的干扰。

GUICtrlSetState (图片控件 ID, @sw_disable)

【关于控件 ID】在 Au3 中，我们可以通过“变量名=创建 GUI 的函数”来指定控件的 ID，方便后面的调用。例如我们要建立一个图片控件并指定它的 id 为 \$image 可以这样写代码：

\$image = GUICtrlCreatePic("..\GUI\mslogo.jpg", 50, 50, 200, 50)

创建一个图片控件之后，以后要是您想换个图片，该怎么办呢？请用下面这个函数：

GUICtrlSetImage (控件 ID, 文件名 [, 图标名 [, 图标类型]])

创建按钮

创建按钮，请使用下面的这个函数

GUICtrlCreateButton ("文本", 左侧, 顶部 [, 宽度 [, 高度 [, 样式 [, 扩展样式]]])

设置按钮的数据

GUICtrlSetData (控件 ID, 数据 [, 默认值])

创建输入框

输入框使您可以从用户那里获得文本信息，它有单行与多行两种

单行输入框的创建：

GUICtrlCreateInput ("文本", 左侧, 顶部 [, 宽度 [, 高度 [, 样式 [, 扩展样式]]])

多行输入框的创建

GUICtrlCreateEdit ("文本", 左侧, 顶部 [, 宽度 [, 高度 [, 样式 [, 扩展样式]]])

设置控件的数据，用的和上面那个是一样的，就不写了。

创建标签

使用函数

GUICtrlCreateLabel ("文本", 左侧, 顶部 [, 宽度 [, 高度 [, 样式 [, 扩展样式]]])

设置控件数据，同上。

获取控件数据

建立好我们的 GUI 之后，您是否会想给您的 GUI 加一点功能呢？比如说您有一个输入框，在里面输入东西以后，您希望您的脚本在您按下按钮之后把文字为您读出来（因为我的系统没有中文的语音库，所以只支持英文啦）。这里我们需要用到一个函数。

GUICtrlRead (控件 ID [, 高级])

以下是我的脚本

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

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

```
#include <EditConstants.au3>
```

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

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

```
#Region ### START Koda GUI section ### Form=
```

```
Dim $nr
```

```
$Form1 = GUICreate("简易英文阅读器", 409, 223, 192, 114)
```

```

$edit=GUICtrlCreateEdit("", 16, 16, 305, 137)
$Button1 = GUICtrlCreateButton("开始阅读", 40, 168, 235, 25)
$Button2 = GUICtrlCreateButton("退出", 304, 168, 75, 25)
GUISetState(@SW_SHOW)
#EndRegion ### END Koda GUI section ###
While 1
    $nMsg = GUIGetMsg()
    Switch $nMsg
        Case $GUI_EVENT_CLOSE
            Exit
        Case $Button1
            $tnr=GUICtrlRead($edit)
            If $tnr=""Then
                $nr="You must type in words"
            EndIf
            speak($nr)
        Case $Button2
            Exit
    EndSwitch
WEnd
Func speak($nr)
    $Voice = ObjCreate("Sapi.SpVoice")
    $Voice.Speak ($nr)
EndFunc

```

【8】键盘模拟—将自动化进行到底

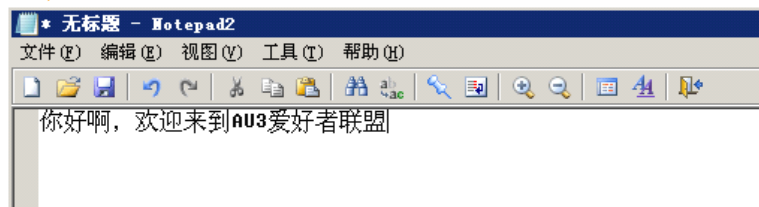
首先，打开您的 **SCITE 编辑器**，输入以下代码，

```

Run("notepad.exe")
WinWaitActive("无标题 - Notepad2")
Send("你好啊，欢迎来到 AU3爱好者联盟")

```

额，我电脑用的是用 Notepad2 替换过的 Notepad，所以标题会不一样，使用系统自带记事本的话，将代码中的 `WinWaitActive("无标题 - Notepad2")` 改成 `WinWaitActive("无标题 - 记事本")` 即可。你会发现脚本自动在打开的记事本中输了汉字：



图四十八

怎么样啊，忒有趣吧！当然在 AU3 中，键盘鼠标模拟不止是能做这些。接下来，就是见证奇迹的时刻！

在 AU3 中模拟键盘鼠标操作的函数，主要是以下几个。

`HotKeySet ("热键" [, "函数名"])`

`Send ("按键" [, 标志])`

ControlClick ("标题", "文本", 控件 ID [, 按钮 [, 点击次数 [, X 坐标 [, Y 坐标]]]))

ControlFocus ("窗口标题", "窗口文本", 控件 ID)

请大家参考帮助文档来学习这几个函数。本人才疏学浅，而这几个函数又是特别重要的，希望大家好好学习，不明白的可在论坛发帖讨论！

【9】字符串操作

您是否有时候想把提取某个字符串的部分字符串或者想去掉它两头的字符呢？通常情况下，我们把这种通过不同方法来实现以上目的的操作叫做字符串操作。在 AUTOIT 中，关于字符串操作的函数都是位于帮助文件“字符串管理”这一块的。在这节，我们将讲讲几个简单的函数。

在接下来的大多数例子中，我将使用下面的这个字符串，因为它包含了英文中的所有字母，而且操作也行对简单。

"The quick brown fox jumps over the lazy dog"

[1]字符串长短——StringLen

这个函数用于返回字符串的长度

StringLen ("字符串")

例如：

```
MsgBox(0, "Length of string", 'The length of "The quick brown fox jumps over the lazy dog" is ' &StringLen("The quick brown fox jumps over the lazy dog") & ' characters')
```

[2] StringInStr

此函数用来返回指定字符串或者字符在目标字符串中出现的位置。

StringInStr ("字符串", "子字符串" [, 区分大小写 [, 出现次序 [, 开始 [, 数量]]]))

例如

```
MsgBox(0, "Length of string", 'The first occurrence of fox is at' & _  
StringInStr ("The quick brown fox jumps over the lazy dog", "fox") & _  
' characters from the left-hand side of the string')
```

[3]字符串拆分——StringSplit

这个函数用来以指定分隔符把字符串拆分成若干子串。

StringSplit ("字符串", "分隔符" [, 标志])

例如

```
$string = "The quick brown fox jumps over the lazy dog"  
$aReturn = StringSplit ($string, " ")  
For $i = 1 to $aReturn[0]  
MsgBox (0, $i, $aReturn[$i])  
Next
```

[4]删去“空白符”——StringStripWS

这个函数用于删去字符串中的“空白符”。

StringStripWS ("字符串", 标志)

例如

```
$string = "The" & @CR & "quick" & @CR & "brown" & @CR & "fox" & @CR & "jumps" &  
@CR & "over" & @CR & "the" & @CR & "lazy" & @CR & "dog"  
$string2 = "The" & @CRLF & "quick" & @CRLF & "brown" & @CRLF & "fox" & @CRLF &  
"jumps" & @CRLF & "over" & @CRLF & "the" & @CRLF & "lazy" & @CRLF & "dog"  
$string2 = "The      quick      brown      fox      jumps      over      the  
lazy      dog      "
```

```
MsgBox (0, "", StringStripCR ($string))
MsgBox (0, "", StringStripWS ($string2, 4))
```

[5]其他

也许有时候您需要处理很多字符串，下面是一些简单的方法：

例子1

```
#include <Array.au3>
$string = "1234567890"
$array = _StringSplitBy($string, 2)
_ArrayDisplay($array)
Func _StringSplitBy($string, $count)
    $c = 1
    $split = StringSplit($string, "")
    Local $rArray[UBound($split)]
    ConsoleWrite(UBound($split) & @CRLF)
    For $i = 1 To UBound($split) - 1 Step $count
        ConsoleWrite("I = " & $i & @CRLF)
        For $x = $i To $i + $count - 1
            ConsoleWrite("X = " & $x & @CRLF)
            $rArray[$c] &= $split[$x]
        Next
        $c += 1
    Next
    ReDim $rArray[$c]
    $rArray[0] = $c - 1
    Return $rArray
EndFunc ;==>_StringSplitBy
```

例子2

```
#include <Array.au3>
$string = "1234567890"
$array = _StringSplitBy($string, 2)
_ArrayDisplay($array)
Func _StringSplitBy($string, $count)
    Local $rArray[StringLen ($string)]
    Local $x = 1
    For $i = 1 To StringLen ($string) Step $count
        $rArray[$x] = StringMid ($string, $i, $count)
        $x += 1
    Next
    ReDim $rArray[$x]
    $rArray[0] = $x - 1
    Return $rArray
EndFunc ;==>_StringSplitBy
```

例子3

```
#include <array.au3>
```



```

$string = "1234"
$aArray = MyStringSplit($string)
_ArrayDisplay($aArray)
Func MyStringSplit($string)
    Local $len = StringLen($string) / 2
    Local $aArray[$len + 1]
    $aArray[0] = $len
    For $i = 1 To $len
        $aArray[$i] = StringMid($string, $i*2-1, 2)
    Next
    Return $aArray
EndFunc

```

试试以上脚本，并选择您最喜欢的脚本，他们最终实现的功能都是一样的，所以无论您用哪个都是可以的。它们主要表明了字符串操作中达成一个目的可以用很多种方式。所以，任何问题的解决方式，都不是唯一的。

【10】FileInstall

这个函数用于包含并装入指定文件到编译后的脚本程序中。，在一些情况下，我们的脚本需要调用外部的文件或者执行程序，但是我们又不想弄 N 多的文件拷来拷去，我们就可以使用这个函数来将我们要用到的文件或者程序编译进我们的脚本中去。

FileInstall ("源文件", "目标路径" [, 标志])

譬如：

我们在 C 盘根目录建立一个名为 our.txt 的文本文件，然后使用如下脚本可以将其编译进我们的脚本中，并用记事本打开。

```

FileInstall("c:\our.txt", @TempDir & "\")
Run("notepad.exe "& @TempDir & "\our.txt")

```

【11】_FileListToArray

这是一个用户自定义函数（UDF），在使用的时候必须使用 #Include <File.au3> 来将脚本包含进来，这点很像 ActionScript 里面的在使用一些属性或者方法之前必须导入包，Au3 在使用用户自定义函数的时候，必须使用 #Include <函数文件> 来包含。这个函数的使用格式是

```

#include <File.au3>
_FileListToArray(路径 [, $sFilter = "*" [, $iFlag = 0]])

```

例子

```

#include<file.au3>
Dim $path,$list,$i
$path=FileSelectFolder("请选择您要生成目录的文件夹","")
$list=_FileListToArray($path,"*",1)
    For $i=1 To $list[0]
        Run(@ComSpec & " /c echo "&$list[$i]&">>D:\list.txt", "", @SW_HIDE, "")
    Next
MsgBox(0,"哈哈","目录生成完毕，请到 D 盘根目录查看")

```


三、篇外篇

你 DllCall 了吗？

首先，申明下，关于这个 DllCall 的内容不是我写的，我是参考 Andreas Karlsson 2009 |《Dealing with Dlls in AutoIt》来半翻译半写的，英文不好，所以会很生涩，请大家莫怪！

我的第一个 Dllcall ()

我们首先来实现 sleep () 这个函数，这个功能驻留在 Kernel32.dll 这个文件中。您可以使用 autoit 中的 sleep () 来实现类似的功能。实质上，autoit 仅仅是您和这个函数之间的一个中介而已。该函数的 MSDN 文档 [http://msdn.microsoft.com/en-us/library/ms686298\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/ms686298(VS.85).aspx) 查看该函数的文档的时候，您会发现这个函数仅有一个参数，并且没有返回值，接下来，让我们使用我们今天的主角函数--dllcall 来实现同 autoit 中 sleep () 相同的功能！

DllCall ("dll", "return type", "function" [, "type1", param1 [, "type n", param n]])

让我们仔细来看它的参数，首先是“dll”这个参数，这是我们所要运用的函数功能就存储在这个 dll 文件中。之前我们说起，sleep 这个功能函数存储在 Kernel32.dll 这个文件中，所以这里，我们将 Kernel32.dll 放在“dll”这个参数的位置。

其次，是返回值，该函数功能在 MSDN 文档中，没有返回值，所以，在“return type”这里，我们赋值为“none”。接下来的就不是太难了，我们知道我们要实现的函数功能的名字为 sleep，所以，这里将其赋值为 sleep。

紧接着，让我们来看看参数，正如您所见，现在亟需做的事情，是指定参数的类型。在 MSDN 中，这被称之为“DWORD”，而在 Autoit 中，则简单的叫做“dword”。所以，在 type1 这个参数这里，我们设置其为“dword”。参数的类型就这样被指定了，然后通过“param1”这个参数，您可以指定您 sleep 时间长短的值。

现在在 DllCall() 里面已经没有其他的参数了，但是如果这个 sleep 函数有更多的参数，您就应当重复使用。DllCall 提供可选量的参数，这样对于用户来说，确为一件很酷的事。

如果函数有返回值，那么返回值将储存于一个数组内，在这个数组中以 0 为下标（例如 array[0]）的数据就是从 dll 中返回的数据。

希望您现在已经有自己得意的代码了，即使您现在还没有也不必泄气！

```
; Sleeps for 1 second
```

```
DllCall("Kernel32.dll","none","Sleep","dword",1000)
```

希望你现在明白了这个函数用法！在结束这一小节内容之前，请您查阅相关资料，实现以下函数功能。

LockWorkStation -----提示：BOOL 是一个整型 (int)，当没有参数的时候，您可以完全跳过 DllCall 的参数选择。

GetCurrentProcessId ----提示：返回值是作为数组返回的，数组中下标为 0 的，即是返回值。

DllCall 中的字符串

正如您所知，在 Dll 文件中的字符串实质上是阵列字符，幸运的是 Autoit 帮助我们解决了这个问题，您所要做的，只是选择“str”还是“Wstr”，然后在某个参数位置输入您的字符串。另外有一件很重要的事情要提醒您的是在 win2000 系统出现之前 windows 系统并不完全支持 Unicode，几乎所有的函数在处理字符串的时候都是以两种编码来处理的。这两种编码通常在它们的名字末尾有一个“A”或者“W”，譬如这个“MyStringFunctionA”。一般的，“A”代

表 ANSI 编码 , "W"代表宽字符, 即 Unicode 编码。

现在让我们来看一下这节要用的 dll 中的函数 [MessageBox](#), 当然在 autoit 中也有收录。正如您所看见的那样, 这个函数有很多不同的可以供您设定的标志值, 但是很遗憾的是 MSDN 只指定了它的符号名城, 这就意味着您所看到的不同的名称实际上只是一个数字而已。如果您想找到对应值, 建议您在谷歌搜索下这个名称或者查询一个 winAPI 语言的扩展版本, 如针对于 C/C++ 的 windows.h。

这里我们使用 0 来作为 “uType” 的值。

下面的是这节要用到的源码, 请认真研读, 然后作适当改写。例如, 试着将 uType 的值改为 16。

; 我们写成 MessageBoxW 是因为我们要用 unicode 编码, 这样意味着我们只能使用 wstr
; hwnd 的值设置为 0 是因为没有和 messagebox 句柄关联的窗口。

DllCall("user32.dll","int","MessageBoxW","hwnd",0,"wstr","我爱 AU3!","wstr","Info","uint",0)

这就是这节的内容, 认真揣摩以上代码, 并完成以下内容!

将以上代码改写成 ASCII 版本

执行 [FindWindow](#)

参数传递

如果您曾经在一个函数声明中见到过 Autoit 中的 “ByRef” 关键字, 您就会明白这是怎么回事。简单来说, 就是您发送一个值将值传递给 Dll 文件, Dll 文件为您对值进行一些处理。这对于 Dll 通过返回值来表明它执行成功但仍要返回值给其调用者的这种情况来说, 是很实用的。

在这节中, 我们要用到的是 windows 中的一个示例函数 [GetDiskFreeSpace](#)。如果要传递一个参数作为参考值, 只需要在参数类型那里加上一个 “*” 即可。

以下是函数实现的过程:

; 变量作为参数传递

Local \$SectorsPerCluster

Local \$BytesPerSector

Local \$NumberOfFreeClusters

Local \$TotalNumberOfClusters

\$calldata=DllCall("Kernel32.dll","int","GetDiskFreeSpaceW","wstr","C:\\","dword*",\$SectorsPerCluster,"dword*",\$BytesPerSector,"dword*",\$NumberOfFreeClusters,"dword*",\$TotalNumberOfClusters)

; 即使变量的值被改变, 返回值仍然以数组形式被返回

\$SectorsPerCluster=\$calldata[2]

\$BytesPerSector=\$calldata[3]

\$NumberOfFreeClusters=\$calldata[4]

\$TotalNumberOfClusters=\$calldata[5]

MsgBox(0,"","Total number of clusters: "&\$TotalNumberOfClusters)

正如您所见到的那样, 所有的数据都存在于数组中, 并且变量没有完全被改变。如果您想要完全改变数组中变量的值, 您就必须重赋新值用以覆盖掉原值。

以上就是这一章节的所有内容, 我找不到任何适当的可以用来供练习的题目, 所以请务必确保您非常明白这一章节的内容!

与调用规则相关的一个词

说到 Dllcall 就不得不提一下“调用规则”。调用规则是用来规定函数的调用如何被执行。在 windows 中，所有 Dll 文件都遵循 stdcall 调用规则，在 Autoit 中它同样也是作为一个标准模式来使用的。然而一个叫“cdecl”的调用规则的使用也十分普遍。而且在 Autoit 中使用它十分简单，只需将“cdecl”放在返回类型之后即可。所以

```
; With stdcall
```

```
DllCall(" SomeDll.dll" ," int" ," Func" )
```

可以变为

```
; With cdecl
```

```
DllCall(" SomeDll.dll" ," int:cdecl" ," Func" )
```

DllStructs——极为难的一个东东

下面我们开始学习真正需要很好了解的，特别实用的东西——DllStructs,，那么什么是 DllStructs,呢？请把它们想象成内存中一个个用来装数据的包，在这些包里面装着紧紧挨着彼此变量数据。

那么我们为什么需要这些“包”呢？难道我们不能把所有的数据以参数形式传递给函数？在某些情况下，我们是那样做的，但是在大多数情况下，这是一种费力不讨好的做法。因为在某些时候变量的数量可能会发生变化，这些变化会将您的代码变得冗长而复杂。同样地，如果一个函数需要返回很多值，使用“包”也是很有必要的。因为使用“包”之后，可以返回指针（内存地址）到一个(dll)包含很多变量（或者元素，因为从现在开始我将把他们看作是表格一样的东西）的结构中去。另外一种方法就是将指针作为一个参数传递到(DLL)结构中，并让其处理其指向的（DLL）结构。

这样做的另一个好处就是能够使 Autoit 进行直接的内存存取。也许这不是特别有用，但是当您使用 advanced low-level(相对的 LOV 级)代码的时候，您会发现没有它，Autoit 会被削弱很多。

作为第一个例子，请参看 [GetSystemTime](#) 这个函数，这个函数是一个比较简单的函数，它只有一个参数。并且这个参数是一个指向 SYSTEMTIME 结构的指针。如果您想调用这个结构，您就必须将其用一个 Autoit 的 DllStruct 表示出来。请点击链接查看 [SYSTEMTIME](#) 的相关文档，其结构如下：

```
typedef struct _SYSTEMTIME {  
    WORD wYear;  
    WORD wMonth;  
    WORD wDayOfWeek;  
    WORD wDay;  
    WORD wHour;  
    WORD wMinute;  
    WORD wSecond;  
    WORD wMilliseconds;  
} SYSTEMTIME, *PSYSTEMTIME;
```

您最先注意到的，可能是 WORD 这种类型，这种类型在 Autoit 中是没有被指定的，所以您必须得弄明白微软用 WORD 所表达的实质意思。通过谷歌搜索“WORD typedef”发现，原来 WORD 在 Autoit 中实际上是一个无符号的又名“ushort”的 16 位整数(请查阅 Dllcall 函数帮助文件的相关项)。因此，该结构使用 Autoit 的完整表达（和简单的调用）如下：

```
$SYSTEMTIME=DllStructCreate("ushort wYear;ushort wMonth;ushort wDayOfWeek;ushort  
wDay;ushort wHour;ushort wMinute;ushort wSecond;ushort wMilliseconds")
```

```
DllCall("Kernel32.dll","none","GetSystemTime","ptr",DllStructGetPtr($SYSTEMTIME))
MsgBox(0,"Current
time:",DllStructGetData($SYSTEMTIME,"wHour")&":"&DllStructGetData($SYSTEMTIME,"w
Minut
e"))
```

当您觉得您已经理解上面代码的含义的时候，请多加练习，并发觉其中新的。试着做下面的练习：

使用 [SetLocalTime](#) 函数将您当前系统的时间改到明天下午的六点，并修改回来。

四、后记

感谢

感谢鸿哥一直以来对 AU3 爱好者联盟的支持！也感谢 Acn 的大大们先前所作的一切工作。感谢思远的朋友们，是你们让我知道，我在这个世界上还有值得引以为自信的东西！

经过好几天的努力，终于完成了这本电子书的编写，编写的过程是枯燥和寂寞的，但是想到更多的人能够来到 Au3 爱好者联盟这个大家庭中，我也暗地里高兴。

本人不是什么高手，只是一个从菜鸟过来的大菜鸟，我只是将我在学习 Au3 的过程中的这些心得写下来，让大家更加喜欢 AU3，喜欢这个东西！

参考书目

《Au3 帮助文档》

《思远技术论坛 AU3 教程》

《Dealing with DLLs in AutoIt》

《Learning to Script with AutoIt V3 》

AU3 官方论坛

AU3 中文站

AU3 爱好者联盟 www.autoit.tk

思远技术论坛 <http://bbs.seeyoon.com>

虫子樱桃

2010 年秋 于 成都