AutoHotkey/AutoIt基础

前言：据我了解需要编写AutoHotkey/AutoIt脚本来实现自动化操作的用户很多都是网管，其它则可能是一些个人用户，他们一般都具有相当的技术水平，而且都希望能借助脚本来完成某些以往需要人工操作的重复性劳动，但限于语言条件上的限制可能对官方的帮助文档有较难理解之处。为方便读者，我将从最简单的说起，每个示例尽可能同时给出相应的AHK和AU3版本代码。本文将尽可能用较通俗的语言描述，但并不打算讲解语法基础，所以不一定适合新手阅读。

文中涉及到的AHK/AU3版本：

AutoHotkey 1. 0. 44 . 08

AutoIt 3.1.1

一、关于脚本

1、什么是脚本？

这是个非常“流行”的术语了，通俗而言脚本（Script）一般都是指根据某种语法规则编写的具有特定格式的文本文件。可能大家已经听说过很多种脚本：VBScript、JScript、PHP、ASP、JSP、CGI、CS脚本，甚至游戏外挂脚本。

这些脚本文件都是可执行文件，可执行相应的操作。

AHK 脚本文件扩展名：*.ahk

AU3 脚本文件扩展名：*.au3

2、脚本和程序的不同？

严格来说，所谓“程序”就是指以各种编程语言（比如说C/C++/C#/Delphi）编写、由编译器编译好后的二进制文件，一般就是机器代码，可由系统执行。而脚本则是只是些纯文本文件，包含了各种定义好的命令，这一点很像批处理文件。这样，我们得出一个简单的结论，那就是用户一般无法获得“程序”的源代码，我们只能进行反汇编把它逆向还原为汇编语言代码（或其它），当然，也有些“程序”是可以获得源代码的（比如Java）；脚本则是用户可直接查看的代码文件，而AHK/AU3则提供了把脚本文件“转换”成exe文件的方法。

3、脚本如何运行？

脚本是“解释性”的语言，它的运行依赖一个“解释器”，由这个解释器来“翻译并解释”脚本的每条命令（或者说代码），然后执行相应操作。如果不严格定义的话，HTML和Java都可以认为是解释性语言。AHK/AU3的主程序（分别是AutoHotkey.exe和AutoIt3.exe）就是它们的“解释器”，上面提到脚本可“转换”成可脱离相应的解释器而独立运行的exe可执行文件，而我们还可以使用相应的工具把它们“还原”成脚本文件，由此我们完全可以这么理解：脚本代码是被“压缩”到这个exe文件中，解释器也是在里面，在运行exe时实际上是先“解压”脚本代码然后运行解释器并解释该脚本。

4、如何创建脚本？

使用资源管理器的右键菜单即可创建相应脚本文件，或者新建一个文本文件后改扩展名即可。

5、稍微介绍一点语法规则？

A）对AHK而言，每个内建的功能都是以“命令”的形式提供：

Command, param1, param2,…

而AU3则以“函数”的形式提供：

Function(param1, param2, …)

命令或函数中被符号“[”和“]”围住的参数是可选参数，表示在使用这些命令或函数时可省略它们（不给出具体数值）。

若某个参数含有空格，则最好使用双引号围住该参数。

B）解释器自上而下（从第一行到最后一行）“解释”脚本的每行语句，除非遇到“Return”、“Goto”、“Gosub”、“Exit”等语句、函数、热键或其它能使脚本“跳”到某个标识符的条件成立。

C）关键字和标识符（包括变量名、命令名、函数名等）都不区分大小写。

二、运行程序或打开文件

1、运行程序

Run t命令或者函数用来运行外部可执行文件，AHK还可利用它来直接打开文件。

AHK：

```
Run, 目标文件 [, 工作目录, Max|Min|Hide|UseErrorLevel, 输出PID变量]
```

AU3：

```
Run ( "文件名" [, "工作目录" [, 标志]] )
```

【示例 2.1.1 】

AHK：

```
Run, Notepad.exe
```

AU3：

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

上面的示例中都没有给出程序“Notepad.exe”的路径，为什么仍能执行？这是因为它们都会自动在脚本所在目录下搜寻目标文件，如有则运行，否则就到系统文件夹（%PATH%）中搜寻。

注意：

A）某些程序必须给定“工作目录”才能成功运行！

B）给出完整的文件路径有助于轻微提高程序的可靠性。

C）AHK的Run命令可以用来运行程序和直接打开文件，而AU3的Run函数则只能用来运行程序（可执行文件）或传递参数让某个程序打开目标文件。

当然，运行程序的功能还不仅仅是这么简单，我们还可以指定运行程序的初始状态，比如让运行的记事本窗口以最大化状态显示（或者最小化、隐藏）：

【示例 2.1.2 】

AHK：

```
Run, Notepad.exe, , Max
```

AU3：

```
Run("Notepad.exe", "", @SW_MAXIMIZE)
```

2、打开文件

前面已经提到，AHK的Run命令可以直接打开文件，而AU3的Run函数则只能用来运行程序，因此在打开文件的方式上有点不同：AHK脚本中可直接给出目标文件，而AHK将自动运行该文件的关联程序来打开它；而AU3则必须由用户自己传递参数让某个程序打开目标文件。

【示例 2.2.1 】

AHK：

```
Run, MyFile.txt
Run, Notepad.exe MyFile.txt
```

AU3：

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

3、以命令行形式运行程序

可以考虑运行系统的命令行解释器（cmd.exe/command.com），然后指定要执行的命令并传递参数。

假设我们要执行命令“dir C:/WINDOWS/system 32”，用以列出指定目录的所有文件及子目录。

【示例 2.3.1 】

AHK：

```
Run, %ComSpec% /k dir C:/WINDOWS/system32
```

AU3：

```
Run(@ComSpec & " /k dir C:/WINDOWS/system32")
```

注意：

A）ComSpec是脚本内建的用以指示命令行解释器位置的变量或宏。

B）/k 参数表示“执行字符串指定的命令但保留”，若改为 /c 则表示“执行字符串指定的命令然后终断”。对此比较直观的解释是 /k 将在执行完命令后保留命令提示窗口，而 /c 则将在执行完命令之后关闭命令提示窗口。

C）符号“&”是AU3定义的字符串连接符。

4、特殊应用

A）打开网页

【示例 2.4.1 】

AHK：

```
Run, www.autohotkey.com
Run, %A_ProgramFiles%/Internet Explorer/IEXPLORE.EXE www.autohotkey.com
```

AU3：

```
Run(@ProgramFilesDir & "/Internet Explorer/IEXPLORE.EXE www.autohotkey.com")
```

B）打开特殊文件夹

系统的某些特殊文件夹被定义了相应的**CLSID**（请查看帮助文档），我们可利用它来打开相应的文件夹，比如打开回收站：

【示例 2.4.2 】

AHK：

```
Run ::{645ff040-5081-101b -9f 08-00aa 002f 954e}
```

AU3：

不适用！

C）运行控制面板工具

微软已经为我们提供了通过命令行打开控制面板某个工具或项目的方式，比如打开系统属性窗口：

【示例 2.4.3 】

AHK：

```
Run control sysdm.cpl
```

AU3：

```
Run("control sysdm.cpl")
```

关于访问控制面板项目的详细介绍请查看此文：文章地址。

D）指定搜索位置并打开搜索窗口

假设我们要打开一个搜索窗口，而且要指定搜索位置，比如C:/：

【示例 2.4.4 】

AHK：

```
Run, find C:/
```

AU3：

不适用！

E）显示指定文件的属性窗口

假设我们要打开文件“MyFile.txt”的属性窗口，则使用关键字properties 然后接上目标文件即可：

【示例 2.4.5 】

AHK：

```
Run, properties MyFile.txt
```

AU3：

不适用！

注意：AHK在退出前将自动关闭打开的属性窗口！

F）用“资源管理器”打开指定文件夹

我们知道使用Run, explorer C: 或Run("explorer C:") 即可打开指定的文件夹，可是有时候我们需要在资源管理器中打开它，这时可使用关键字 explore：

【示例 2.4.6 】

AHK：

```
Run, explore C:
```

AU3：

不适用！

G）打印指定文件

要打印指定文件，可使用关键字 print：

【示例 2.4.7 】

AHK：

```
Run, print MyFile.txt
```

AU3：

不适用！

三、窗口操作

注意：窗口标题和窗口文本参数总是对大小写敏感的。

1、等待窗口系列命令/函数

AHK和AU3都提供了用法类似的一组窗口等待命令/函数：WinWait/WinWaitActive/WinWaitClose。它们分别用于等待窗口出现、等待窗口被激活、等待窗口被关闭。由于这些命令/函数的参数类似，现仅以WinWait为例说明。

AHK：

WinWait [, 窗口标题, 窗口文本, 超时时间, 排除标题, 排除文本]

AU3：

WinWait ("窗口标题" [, "窗口文本" [, 超时时间]])

WinWait 的作用是在目标窗口出现之前不再执行后面的所有语句。

假设我们要运行记事本程序，并在其窗口出现时提示用户：

【示例 3.1.1 】

AHK：

```
Run Notepad
WinWait, 无标题 - 记事本
MsgBox 记事本窗口已被打开！
```

AU3：

```
Run("Notepad")
WinWait("无标题 - 记事本")
MsgBox(0, "", "记事本窗口已被打开！ ")
```

2、激活窗口相关命令/函数

让目标窗口成为活动窗口的办法就是激活它，可用的命令/函数是WinActivate：

AHK：

WinActivate [,窗口标题, 窗口文本, 排除标题, 排除文本]

AU3：

WinActivate ("窗口标题" [, "窗口文本"])

3、关闭窗口

关闭窗口有两种方式，一种是正常的关闭窗口（WinClose），另一种则是强行关闭窗口（WinKill）：

AHK：

WinClose/WinKill [,窗口标题, 窗口文本, 超时时间,, 排除标题, 排除文本]

AU3：

WinClose/WinKill ("窗口标题" [, "窗口文本"])

现在我们已经可以实现一个比较简单的功能了，比如我们可以打开系统属性窗口并等待其出现，窗口出现后激活它，接着等待3秒再关闭它：

【示例 3.1.2 】

AHK：

```
Run, Sysdm.cpl
WinWait, 系统属性
WinActivate, 系统属性
WinWaitActive, 系统属性
Sleep, 3000
WinClose, 系统属性
```

WinClose, 系统属性

WinWaitClose, 系统属性

AU3：

Run("Control Sysdm.cpl")

WinWait("系统属性")

WinActivate("系统属性")

WinWaitActive("系统属性")

Sleep(3000)

WinClose("系统属性")

WinWaitClose("系统属性")

建议：如果程序中频繁地出现要用到这些窗口标题的地方，会带来一个问题：脚本的可读性，也许你会想，这不是很直观吗？可问题是如果这个重复出现的窗口标题是个很长的字符串呢？这将严重影响整个代码的排版美观。而且我们也无从了解这些窗口标题的“来头”，不知道这个窗口标题究竟是怎么来的。而如果我们定义一个变量（假设变量名是“AppWindow1”）保存这个窗口标题，我们就能在命令/函数中用变量来表示它，这样就达到了让代码用意更清晰一点的目的。另外，就算目标软件因某些原因（比如升级）而改变了它的窗口标题，我们也能很方便地作出修改。

4、更准确的标识窗口的方法（主要针对AHK脚本）

程序在运行时起码会有一个进程，如果能获得这个进程ID就能在一定程度上保证对窗口的准确标识。另外，每个窗口都有定义窗口类名（Class，比如说记事本窗口的类名就是Notepad），所以我们可以以此排除与目标窗口不同的其它窗口类。其实，我们还有一个更准确的方法：

每个窗口（包括控件在内）都被Windows指派了一个可区别于其它窗口的唯一的标识符（ID），我们称之为窗口句柄（HWND）。

直接给定窗口标题来表示窗口的一个缺点就是无法保证在脚本运行的过程中始终以该窗口为操作目标，因为在这个过程中很有可能会有其它“同名”窗口（或者说满足匹配条件的窗口）出现，而如果我们使用这个标识符来表示窗口自然就能保证命令/函数的操作窗口总是同一个窗口了。

我们先来了解一下获得窗口句柄的命令/函数：

AHK：

WinGet[, 输出变量, ID, 窗口标题, 窗口文本, 排除标题, 排除文本]

AU3：

WinGetHandle ("窗口标题" [, "窗口文本"])

其中WinGet获得的窗口ID将通过“输出变量”返回，而WinGetHandle的返回值就是获得的窗口ID。

我们在进行自动化操作时是要先运行某个程序，如何获得这个程序成功运行后显示的窗口句柄？一个比较保险的办法是先获得这个程序的进程ID，然后根据这个进程ID获得它的窗口句柄，AHK支持使用进程ID作为窗口标题使用；但AU3不支持这样使用，只能先获得该窗口的类名再根据该类名来获得窗口句柄（不够保险）：

【示例 3.1.3 】

AHK：

Run, NotePad, , , ThisPID

WinWait, ahk_pid %ThisPID% ;这里的ahk_pid表明跟在后面的变量是进程ID

WinGet, ThisID, ID, ahk_pid %ThisPID% ;ThisID将保存获得的窗口句柄

AU3：

```
Opt("WinTitleMatchMode", 4)
    Run("Notepad")
    $handle = WinGetHandle("classname=Notepad")
```

现在暂且先忘记了AU3吧，因为它的窗口函数一般都不支持使用窗口句柄作为（窗口标题）参数。

至于如何在AHK中使用窗口句柄，简单的说，凡是有“窗口标题”参数的命令就可以用窗口句柄来代替，比如：

【示例 3.1.4 】

AHK：

```
    Run, Notepad, , , ThisPID ;先获得运行的记事本程序的进程ID
    WinWait, 无标题 - 记事本 ahk_pid %ThisPID% ;等待该进程窗口的出现
    WinGet, ThisHWND, ID, 无标题 - 记事本 ahk_pid %ThisPID% ;获得窗口句柄
    WinActivate, ahk_id %ThisHWND% ;这里的ahk_id表明跟在后面的变量是窗口句柄
    WinWaitActive, ahk_id %ThisHWND%
    Sleep, 3000
    WinClose, ahk_id %ThisHWND%
    WinWaitClose, ahk_id %ThisHWND%
```

四、模拟键击和鼠标点击

1、模拟鼠标点击（按钮等）控件

既然是模拟用户操作，自然就包括了模拟鼠标点击在内。

适用命令/函数：Click/MouseClick/ControlClick

其中Click/MouseClick用来模拟用户的物理操作（点击），把鼠标点击事件发送到指定坐标位置（相对当前窗口或绝对位置）上，但这种方法并不能保证100%的准确性，屏幕分辨率、用户干扰和系统环境等都会影响到它们的执行结果，而ControlClick则直接把鼠标点击事件发送到目标窗口的目标控件上，因而更准确，一般我们不考虑使用坐标位置方式的点击，下面仅以ControlClick为例说明：

AHK：

```
    ControlClick [, 目标控件或坐标位置, 窗口标题, 窗口文本, 鼠标按钮, 点击次数, 选项,排除标题, 排除文本]
```

AU3：

```
    ControlClick ( "窗口标题", "窗口文本", 控件ID [, 按钮] [, 点击次数]] )
```

对AHK而言，“目标控件”参数是指要点击的控件的类别名（ClassNN）或控件文本，另外还可以使用控件句柄（若用的是控件句柄则第一个参数需留空，并在第二个参数中使用`ahk_id %控件句柄%`）。

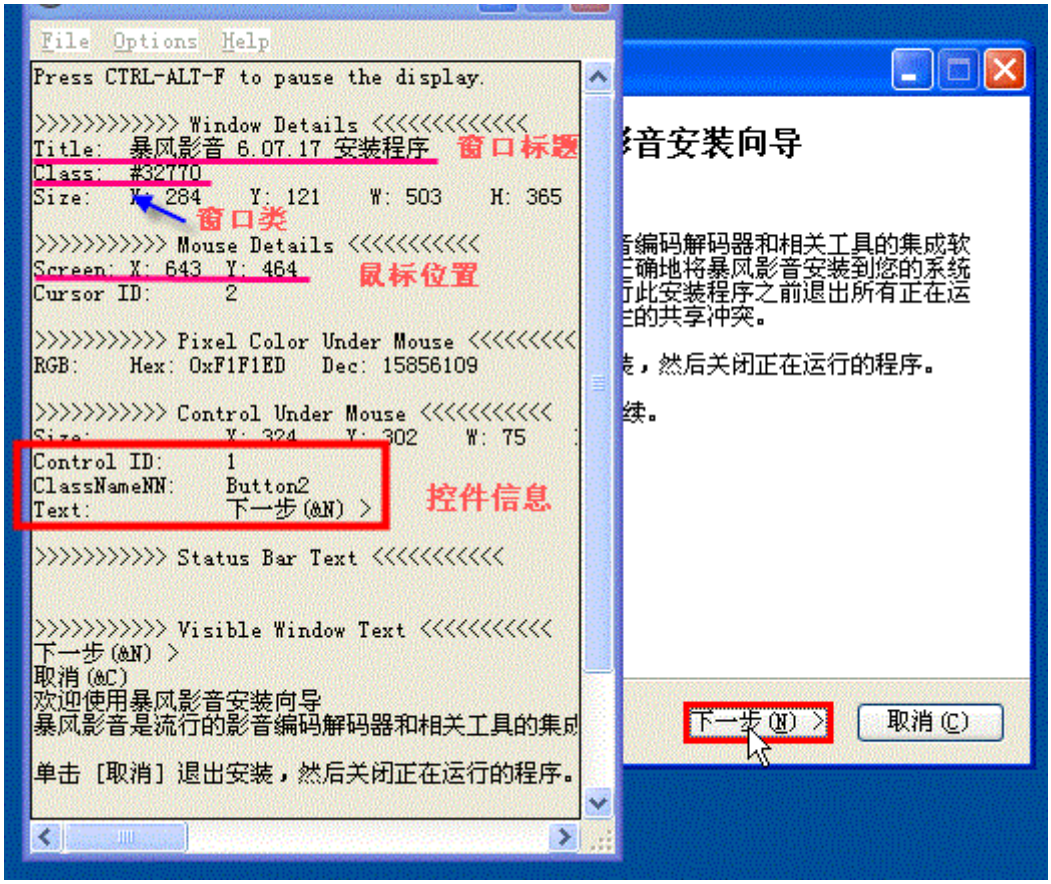
Q：用什么工具来获得目标控件的这些信息呢？

A：AHK用户请使用 AutoIt3 Window Spy，AU3用户则请使用AutoIt Window Info，你可以在相应的开始菜单项目里找到它们，或者到安装目录下寻找。

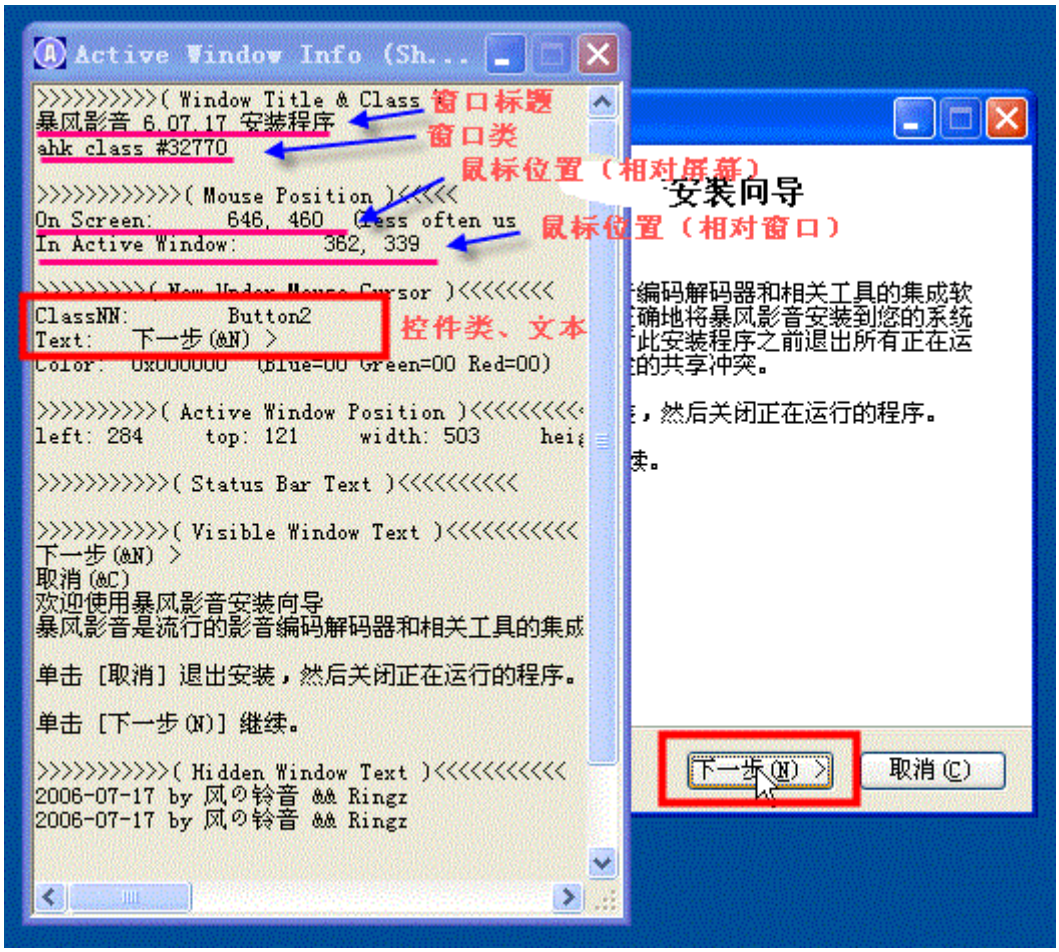
Q：如何使用这两个工具？

A：先打开你要进行操作的目标窗口，然后运行 AutoIt3 Window Spy 或 AutoIt Window Info，接下来就是把鼠标移到目标控件上（比如某个按钮）：

AutoIt3 Window Spy 使用演示截图：



AutoIt Window Info 使用演示截图：



现在我们假设已打开并激活了“系统属性”窗口，而任务是点击它的“确定”按钮，则可用以下几种方法：

【示例4.1.1】

AHK：

```
ControlClick, 确定, 系统属性
```

ControlClick, Button2, 系统属性

AU3：

ControlClick("系统属性", "", 1)

ControlClick("系统属性", "", "Button2")

ControlClick("系统属性", "", "确定")

提醒：即使目标窗口或控件是隐藏状态，ControlClick命令还是可以“点击”目标控件，但不能保证成功率。

2、模拟键盘操作

键盘也是我们在操作窗口时会用到的工具，比如说在安装软件的时候经典的“一路回车大法”。下面简单介绍一下模拟键盘操作的方法。

Send

这个是最直接的方法，就是模拟用户按键行为，直接发送键击命令，用法请参考官方文档，在此不予说明。

最简单的应用——按回车：

AHK：

Run, Control Sysdm.cpl

WinWait, 系统属性

Send, {Enter}

AU3：

Run("Control Sysdm.cpl")

WinWait("系统属性")

Send("{Enter}")

常见的组合键——Alt+X / Ctrl+N等等，在安装软件的时候经常会有提供一个按钮“下一步(N)”，表示按下Alt+N即可触发等同于点击该按钮的效果，其它的可触类旁通。以打开记事本窗口的“文件”菜单为例：

AHK：

Run, Notepad

WinWait, 无标题 - 记事本

WinActivate, 无标题 - 记事本

WinWaitActive, 无标题 - 记事本

Send, !f

AU3：

Run("Notepad")

WinWait("无标题 - 记事本")

WinActivate("无标题 - 记事本")

WinWaitActive("无标题 - 记事本")

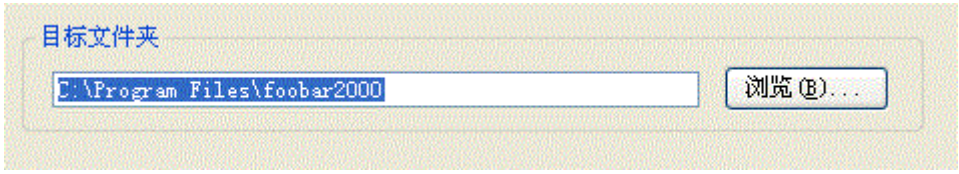
Send("!f")

五、控件操作

然而，在真正实现自动化时仅靠上面的技术往往难以达到预期目的。下面开始进入最为重要的控件操作。

1、设置文本

在安装软件的过程中用户往往需要提供一些必需信息，比如安装目录。很多用户并不喜欢把软件安装到默认的C盘而更愿意把它们安装到别的地方，那么脚本究竟提供了什么方法能让我们修改如下图所示的路径呢？



我们先用上文中提到的AutoIt3 Window Spy 或 AutoIt Window Info 来获得这个路径的编辑框的信息，假设这个窗口的标题为Setup foobar，该路径编辑框的类名是Edit1，而我们需要把它改成“D:/foobar2000”，接下来就可以使用下列命令/函数来设置它的文本了：

AHK：

ControlSetText [, 目标控件, 新文本, 窗口标题, 窗口文本, 排除标题, 排除文本]

AU3：

ControlSetText ("窗口标题", "窗口文本", 控件ID, "新文本")

具体用法如下：

【示例5.1.1】

AHK：

```
ControlSetText, Edit1, D:/foobar2000, Setup foobar
```

AU3：

```
ControlSetText("Setup foobar", "", "Edit1", "D:/foobar2000")
```

2、选中和取消选中单选框和复选框项目

有时程序为了满足用户的个性化设置而需要用户提供更多的信息，我们经常会遇到这样的情况：

如何保证选中所需项目并取消某些项目呢？

下面先来介绍AHK和AU3中用来对控件进行各种属性设置的命令/函数：

AHK：

Control [, 命令, 值, 目标控件, 窗口标题, 窗口文本, 排除标题, 排除文本]

AU3：

ControlCommand ("窗口标题", "窗口文本", 控件ID, "命令", "选项")

其中，“命令”就是让我们指定要进行何种设置的参数。对这些单选框/复选框按钮来说，适用的命令是“Check”和“UnCheck”。

假设这个窗口的标题是为Setup foobar，我们打算进行下来操作：

选中它的“桌面”复选框（Button5），取消选中“快速启动栏”复选框（Button7）

选中它的“桌面”复选框（Button5）、取消选中“快速启动栏”复选框（Button7）；

选中“0.7x”单选框（Button14）。

那么具体的用法示例如下：

【示例5.1.2】

AHK：

```
Control, Check, , Button5, foobar
Control, UnCheck, , Button7, foobar
Control, Check, , Button14, foobar
```

AU3：

```
ControlCommand("foobar", "", "Button5", "Check", "")
ControlCommand("foobar", "", "Button7", "UnCheck", "")
ControlCommand("foobar", "", "Button14", "Check", "")
```

2、选择下拉列表的项目

相信你肯定遇到过下面这种情况：

问题又来了：如何选中自己需要的项目？

答案仍是使用上面提到的命令/函数。对这种控件而言，AHK适用的命令是“**Choose, N**”和“**ChooseString, String**”，分别表示选中第N个项目和选中与字符串String匹配的项目；而AU3适用的命令则是“**SetCurrentSelection, N**”和“**SelectString, String**”，分别表示选中第N+1（注意是从零开始表示！）个项目和选中与字符串String匹配的项目。

假设我们要选中第五个项目“简体中文”，那么具体的用法示例如下：

【示例5.1.3】

AHK：

```
Control, Choose, 5, ComboBox1, Installer
Control, ChooseString, 简体中文, ComboBox1, Installer
```

AU3：

```
ControlCommand("Installer", "", "ComboBox1", "SetCurrentSelection", 4)
ControlCommand("Installer", "", "ComboBox1", "SelectString", "简体中文")
```

来源：<http://blog.csdn.net/liuyukuan/article/details/5958829>