
AutoHotkey 帮助文档

Table of Contents

1. AutoHotkey	10
2. 指南（迅速开始）	11
3. FAQ（常见问答）	18
4. 热键	24
5. 热字串 和 自动替换	31
6. 鼠标与键盘的隐射	39
7. 按键列表（键盘、鼠标与操纵杆）	44
8. 脚本	49
9. 变量和表达式	57
10. 函数	72
11. AutoIt2 用户注意事项	83
12. 按字母顺序排列的命令列表	91
13. 脚本展示	103
14. 更新历史	106
15. 环境管理	117
15.1 ClipWait	117
15.2 EnvGet	119
15.3 EnvSet	119
15.4 EnvUpdate	120
16. 文件、目录以及磁盘管理	121
16.1 Drive	121
16.2 DriveGet	123
16.3 DriveSpaceFree	125
16.4 FileAppend	126
16.5 FileCopy	128
16.6 FileCopyDir	130
16.7 FileCreateDir	131
16.8 FileCreateShortcut	132
16.9 FileDelete	133
16.10 FileGetAttrib	134
16.11 FileGetShortcut	135
16.12 FileGetSize	137
16.13 FileGetTime	138
16.14 FileGetVersion	139
16.15 FileInstall	139
16.16 FileMove	141
16.17 FileMoveDir	142
16.18 FileReadLine	144
16.19 FileRead	145
16.20 FileRecycle	147
16.21 FileRecycleEmpty	147

16.22 FileRemoveDir.....	148
16.23 FileSelectFile	149
16.24 FileSelectFolder	152
16.25 FileSetAttrib	155
16.26 FileSetTime	156
16.27 IfExist/IfNotExist	158
16.28 IniDelete	159
16.29 IniRead	160
16.30 IniWrite	161
16.31 Loop (files & folders).....	161
16.32 Loop (read file contents).....	167
16.33 SetWorkingDir.....	172
16.34 SplitPath.....	172
17. 控制流	174
17.1 #Include/#IncludeAgain	174
17.2 Block	175
17.3 Break.....	177
17.4 Continue	178
17.5 Else.....	178
17.6 Exit	180
17.7 ExitApp.....	181
17.8 Gosub.....	182
17.9 Goto	183
17.10 If.....	184
17.11 If (expression)	186
17.12 Loop	188
17.13 Loop (files & folders).....	189
17.14 Loop (parse a string)	195
17.15 Loop (read file contents).....	197
17.16 Loop (registry).....	202
17.17 OnExit	207
17.18 Pause.....	209
17.19 Return	211
17.20 SetBatchLines.....	212
17.21 SetTimer	213
17.22 Sleep	217
17.23 Suspend	218
17.24 While-loop	219
18. 函数	220
18.1 Asc	231
18.2 Chr	231
18.3 DllCall.....	231
18.4 FileExist	241
18.5 GetKeyState	241

18.6 InStr	242
18.7 IsFunc.....	242
18.8 IsLabel.....	249
18.9 NumGet	249
18.10 NumPut.....	249
18.11 OnMessage	250
18.12 RegExMatch	259
18.13 RegExReplace.....	263
18.14 RegisterCallback.....	266
18.15 StrLen.....	271
18.16 SubStr	271
18.17 VarSetCapacity.....	272
18.18 WinActive.....	274
18.19 WinExist	274
18.20 Math	274
18.20.1 Abs	274
18.20.2 Ceil	274
18.20.3 Exp	274
18.20.4 Floor.....	274
18.20.5 Log	274
18.20.6 Ln	275
18.20.7 Mod	275
;例 3: 检测热键的单击、双击和三次按击。;这允许热键可以按你按键的次数来执行不同的操作: #c::if winc_presses > 0 ; SetTimer 已经启动, 所以我们记录按键。{ winc_presses += 1 return};否则, 这是新一系列按键的首次按键。将计数设为 1 并启动定时器: winc_presses = 1SetTimer, KeyWinC, 400 ;在 400 毫秒内等待更多的按键。returnKeyWinC:SetTimer, KeyWinC, offif winc_presses = 1 ;该键已按过一次。{ Run, m:\; 打开一个文件夹。 }else if winc_presses = 2 ;该键已按过两次。{ Run, m:\multimedia ; 打开一个不同的文件夹。 }else if winc_presses > 2{ MsgBox, 检测到三次或更多次点击。};不论上面哪个动作被触发, 将计数复位以备下一系列的按键: winc_presses = 0return18.20.8 Round	275
18.20.9 Sqrt	275
18.20.10 Sin/Cos/Tan.....	276
18.20.11 ASin/ACos/ATan	276
19. GUI, MsgBox, InputBox & 其它对话框	276
19.1 FileSelectFile	276
19.2 FileSelectFolder	280
19.3 Gui.....	282
19.4 Gui control types	304
19.5 GuiControl.....	326
19.6 GuiControlGet.....	330
19.7 Gui ListView control	333
19.8 Gui TreeView control	355
19.9 IfMsgBox	367

19.10 InputBox	368
19.11 MsgBox	370
19.12 Progress	373
19.13 SplashImage	379
19.14 SplashTextOn/SplashTextOff	385
19.15 ToolTip	387
19.16 TrayTip	388
20. 键盘控制	390
20.1 Hotkeys and Hotstrings	390
20.1.1 #HotkeyInterval	396
20.1.2 #HotkeyModifierTimeout	397
20.1.3 #Hotstring	398
20.1.4 #IfWinActive/Exist	399
20.1.5 #MaxHotkeysPerInterval	402
20.1.6 #MaxThreads	403
20.1.7 #MaxThreadsBuffer	404
20.1.8 #MaxThreadsPerHotkey	405
20.1.9 #UseHook	406
20.1.10 Hotkey	407
20.1.11 ListHotkeys	412
20.1.12 Pause	413
20.1.13 Reload	414
20.1.14 Suspend	415
20.2 #InstallKeybdHook	416
20.3 #InstallMouseHook	416
20.4 #KeyHistory	417
20.5 BlockInput	418
20.6 ControlSend/ControlSendRaw	420
20.7 GetKeyState	422
20.8 Key List (Keyboard, Mouse, Joystick)	426
20.9 KeyHistory	430
20.10 KeyWait	431
20.11 Input	434
20.12 Send/SendRaw/SendInput/SendPlay/SendEvent	439
20.13 SendMode	449
20.14 SetKeyDelay	451
20.15 SetNumScrollCapsLockState	452
20.16 SetStoreCapslockMode	453
21. 数学命令	454
21.1 EnvAdd (+=, ++)	454
21.2 EnvDiv (/=)	455
21.3 EnvMult (*=)	456
21.4 EnvSub (-=, --)	456
21.5 if (expression)	458

21.6 If/IfEqual/IfLess/IfGreater.....	459
21.7 If var [not] between Low and High	462
21.8 If var is [not] type.....	463
21.9 Random.....	464
21.10 SetFormat	466
21.11 Transform.....	469
21.12 var :=expression.....	473
22. 杂项命令	474
22.1 #NoTrayIcon	474
22.2 #SingleInstance.....	474
22.3 AutoTrim	475
22.4 BlockInput.....	476
22.5 ClipWait	479
22.6 CoordMode.....	480
22.7 Critical.....	481
22.8 DllCall.....	482
22.9 Edit.....	493
22.10 ImageSearch	493
22.11 ListLines	496
22.12 ListVars.....	496
22.13 Menu	497
22.14 OutputDebug	505
22.15 PixelGetColor	505
22.16 PixelSearch.....	507
22.17 Reload.....	509
22.18 SetBatchLines.....	509
22.19 SetEnv (var=value)	511
22.20 SetTimer	511
22.21 SysGet	516
22.22 Thread.....	521
22.23 Transform.....	523
22.24 URLDownloadToFile	527
22.25 VarSetCapacity.....	528
23. 鼠标控制	530
23.1 Click.....	530
23.2 ControlClick.....	533
23.3 MouseClick	535
23.4 MouseClickDrag	538
23.5 MouseGetPos	540
23.6 MouseMove.....	542
23.7 SetDefaultMouseSpeed	543
23.8 SetMouseDelay.....	544
24. 进程管理	545
24.1 Exit	545

24.2 ExitApp.....	546
24.3 OnExit	547
24.4 Process.....	549
24.5 Run/RunWait	554
24.6 RunAs.....	557
24.7 Shutdown.....	558
24.8 Sleep	560
25. 注册表管理.....	561
25.1 Loop (registry).....	561
25.2 RegDelete.....	565
25.3 RegRead	566
25.4 RegWrite	568
26. 声音管理.....	569
26.1 SoundBeep.....	569
26.2 SoundGet	570
26.3 SoundGetWaveVolume.....	572
26.4 SoundPlay	573
26.5 SoundSet.....	574
26.6 SoundSetWaveVolume	579
27. 字串管理.....	581
27.1 FormatTime	581
27.2 If/IfEqual/IfLess/IfGreater.....	586
27.3 IfInString/IfNotInString	588
27.4 If var [not] in/contains MatchList	589
27.5 If var is [not] type.....	591
27.6 InStr()	593
27.7 Loop (parse a string)	593
27.8 RegExMatch()	596
27.9 RegExReplace()	600
27.10 SetEnv (var=value)	603
27.11 SetFormat	604
27.12 Sort	607
27.13 StringCaseSense.....	611
27.14 StringGetPos	612
27.15 StringLeft/StringRight	614
27.16 StrLen()	615
27.17 StringLen.....	615
27.18 StringLower/StringUpper.....	615
27.19 StringMid	616
27.20 StringReplace	617
27.21 StringSplit.....	619
27.22 StringTrimLeft/StringTrimRight.....	621
27.23 SubStr()	622
28. 窗口管理.....	622

28.1 Controls.....	622
28.1.1 Control	622
28.1.2 ControlClick.....	625
28.1.3 ControlFocus.....	628
28.1.4 ControlGet	629
28.1.5 ControlGetFocus	634
28.1.6 ControlGetPos.....	635
28.1.7 ControlGetText.....	637
28.1.8 ControlMove.....	639
28.1.9 ControlSend/ControlSendRaw.....	641
28.1.10 ControlSetText	643
28.1.11 Menu	645
28.1.12 PostMessage/SendMessage	652
28.1.13 SetControlDelay	656
28.1.14 WinMenuSelectItem.....	657
28.2 Window Groups.....	658
28.2.1 GroupActivate	659
28.2.2 GroupAdd	659
28.2.3 GroupClose	662
28.2.4 GroupDeactivate.....	663
28.3 #WinActivateForce.....	664
28.4 DetectHiddenText	664
28.5 DetectHiddenWindows.....	665
28.6 IfWinActive/IfWinNotActive	666
28.7 IfWinExist/IfWinNotExist	668
28.8 SetTitleMatchMode	670
28.9 SetWinDelay	671
28.10 StatusBarGetText	672
28.11 StatusBarWait	673
28.12 WinActivate	676
28.13 WinActivateBottom	677
28.14 WinClose.....	678
28.15 WinGet.....	679
28.16 WinGetActiveStats.....	683
28.17 WinGetActiveTitle.....	684
28.18 WinGetClass.....	685
28.19 WinGetPos	686
28.20 WinGetText	687
28.21 WinGetTitle.....	688
28.22 WinHide	689
28.23 WinKill.....	690
28.24 WinMaximize	692
28.25 WinMinimize.....	693
28.26 WinMinimizeAll/WinMinimizeAllUndo.....	694

28.27 WinMove	694
28.28 WinRestore	696
28.29 WinSet	697
28.30 WinSetTitle	702
28.31 WinShow	703
28.32 WinWait	704
28.33 WinWaitActive/WinWaitNotActive	706
28.34 WinWaitClose	707
29. #指令	709
29.1 #AllowSameLineComments	709
29.2 #ClipboardTimeout	709
29.3 #CommentFlag	710
29.4 #ErrorStdOut	710
29.5 #EscapeChar	712
29.6 #HotkeyInterval	713
29.7 #HotkeyModifierTimeout	714
29.8 #Hotstring	715
29.9 #IfWinActive/Exist	716
29.10 #Include/#IncludeAgain	719
29.11 #InstallKeybdHook	720
29.12 #InstallMouseHook	721
29.13 #KeyHistory	722
29.14 #MaxHotkeysPerInterval	723
29.15 #MaxMem	723
29.16 #MaxThreads	724
29.17 #MaxThreadsBuffer	725
29.18 #MaxThreadsPerHotkey	726
29.19 #NoEnv	727
29.20 #NoTrayIcon	727
29.21 #Persistent	728
29.22 #SingleInstance	729
29.23 #UseHook	729
29.24 #WinActivateForce	730
30. Addenda	731
30.1 高级快捷键功能- 鼠标和键盘快捷键	731
30.2 Last Found Window	733
30.3 License	735
30.4 AutoHotkey Acknowledgements	739
30.5 Arrays	740
30.6 Clipboard, ClipboardAll, 以及 OnClipboardChange	742
30.7 CLSID List (Windows Class Identifiers)	744
30.8 ErrorLevel	746
30.9 Standard Windows Fonts	746
30.10 Language Codes	749

30.11 Creating a Keyboard Macro or Mouse Macro	752
30.12 Overriding Hotkeys	753
30.13 Performance	754
30.14 正则表达式 - 快速参考	754
30.15 Remapping a Joystick to Keyboard or Mouse	758
30.16 PostMessage/SendMessage Tutorial	763
30.17 List of Windows Messages	765
30.18 GUI Styles	773
30.19 Threads	793
30.20 Automating Winamp	794
30.21 Context Sensitive Help in Any Editor -- by Rajat	795
30.22 Easy Window Dragging (requires XP/2k/NT)	797
30.23 Easy Access to Favorite Folders -- by Savage	799
30.24 IntelliSense -- by Rajat (requires XP/2k/NT)	805
30.25 Using a Joystick as a Mouse	812
30.26 Joystick Test Script	817
30.27 On-Screen Keyboard (requires XP/2k/NT) -- by Jon	820
30.28 Minimize Window to Tray Menu	828
30.29 Mouse Gestures -- by deguix	837
30.30 Changing MsgBox's Button Names	837
30.31 Numpad 000 Key	837
30.32 Using Keyboard Numpad as a Mouse -- by deguix	840
30.33 ToolTip Mouse Menu (requires XP/2k/NT) -- by Rajat	866
30.34 Volume On-Screen-Display (OSD) -- by Rajat	871
30.35 Window Shading (roll up a window to its title bar) -- by Rajat	876
30.36 WinLIRC Client	877
30.37 AutoHotkey Tutorial - Launch a program or document (continued)	888

1. AutoHotkey

Version 1.0.48.00 (See 14.)

www.autohotkey.com

©2003-2009 Chris Mallett, 部分 ©AutoIt Team

AutoHotkey 是一款免费的、Windows 平台下开放源代码的热键脚本语言。有了它，您可以：

- 通过发送键盘键击和鼠标点击自动化几乎所有的操作。您可以手工编写宏(See 30.11)或者使用宏录制器。
- 为键盘，游戏杆和鼠标创建热键(See 4.)。几乎任何按键、按钮或组合键都可以设置为热键。
- 在您键入缩写时扩展缩写(See 5.)。例如，键入 "btw" 能自动地生成 "by the way"。
- 创建自定义的数据输入表格、用户界面和菜单栏。详见图形用户界面(See 19.3)。
- 重新映射(See 6.)您键盘、游戏杆和鼠标上的按键和按钮。
- 通过 WinLIRC 客户端脚本(See 30.36)对手执遥控器的信号作出反应。
- 运行现有的 AutoIt v2 脚本，并用新功能(See 11.)增强它们。
- 将任何的脚本转换为 EXE 文件(See 8.)，使其在没有安装 AutoHotkey 的计算机上也能运行。

开始使用 AutoHotkey 也许比您想的更为简单。看下快速入门指南(See 2.)吧！

AutoHotkey 释出您的键盘、游戏杆和鼠标的所有潜能。例如，除了典型的 Control 、Alt 和 Shift 这些辅助键外，你还可以使用 Windows 键和 Capslock 键作为辅助键。实际上，你可以使任何按键或鼠标按钮充当辅助键。对于这些和其他的功能，详见高级热键(See 30.1)。

- 调节音量、消音和任意声卡的其他设置(See 26.5)。
- 使任意窗口透明(See 28.29)，置顶(See 28.29)或改变外形(See 28.29)。
- 用游戏杆(See 30.25)或键盘(See 30.32)代替鼠标。
- 监控(See 17.21)你的系统。例如，在不想要的窗口出现时关掉它们。
- 获取并改变剪贴板的内容(See 30.6)，包括从资源管理器窗口复制的文件名。
- 禁用或改变(See 30.12) Windows 自有的快捷键，例如 Win+E 和 Win+R。

- 替代 [Alt-Tab](#)(See 4.) (通过按键、鼠标滚轮或按钮)从而缓解 RSI(肢体重复性劳损)。
- 用您自己的图标、提示框、菜单项和子菜单来自定义[托盘图标的菜单](#)(See 22.13)。
- 显示[对话框](#)(See 19.11)、[提示框](#)(See 19.15)、[气泡提示](#)(See 19.16)和[弹出菜单](#)(See 22.13)来和用户交互。
- 执行脚本动作来[响应](#)(See 17.17)系统关闭或注销。
- 检测用户[空闲](#)(See 9.)了多久。例如，仅在用户离开时运行 CPU 密集型任务。
- 通过探测[图像](#)(See 22.10)和[像素颜色值](#)(See 22.15)来自动执行游戏中的动作(这对于缓解肢体重复性劳损来说是合法的)。
- 比在其他语言中更容易[读取](#)(See 16.32)、[写入](#)(See 16.4)和[解析](#)(See 17.14)文本文件。
- 对一组与[通配符类型](#)(See 16.31)匹配的文件进行操作。
- 可对[注册表](#)(See 25.3)和[INI 文件](#)(See 16.29)进行操作。

软件许可: [GNU 通用公共许可](#)(See 30.3)

特别感谢 Jonathan Bennett，他于 1999 年把 AutoIt v2 作为免费软件慷慨地发布出来，使其成为我自己以及世界上许多其他人获得灵感和节省时间的工具。另外，AutoHotkey 许多针对 AutoIt v2 指令集的增强功能，以及 Window Spy 和脚本编译器，都是直接改编自 AutoIt v3 的源代码。所以同样为此感谢 Jon 以及 AutoIt 的其他开发人员。

最后，AutoHotkey 离开了[这些人](#)(See 30.4)也不会有今天。

翻译：天堂之门 menk33@163.com 2008 年 10 月 21 日

2. 指南（迅速开始）

这个简短的介绍将帮助你马上开始编写你自己的宏和热键脚本。

- [创建一个脚本](#)
- [启动一个程序或文档](#)
- [发送键击和鼠标点击](#)
- [激活与操纵窗口](#)
- [从用户使用的 MsgBox, InputBox 等命令来获取输入的数据](#)
- [使用变量和剪贴板](#)
- [一遍遍地重复一系列动作](#)
- [操纵文件和文件夹](#)
- [其他功能的概述](#)

每个脚本都是一个包含命令的要被程序(AutoHotkey.exe)执行的纯文本文件。一个脚本也可能包含[热键\(See 4.\)](#)和[热字串\(See 5.\)](#)，甚至于全部由它们组成。不过，在没有热键和热字串时，一个脚本从它被启动时起，将从头至尾顺序地执行它的命令。

要创建一个新脚本：

1. 打开 Windows 资源管理器并进入一个你选择的文件夹。
2. 拉下文件菜单并选择 新建 >> AutoHotkey Script (或 新建 >> 文本文档)。
3. 给文件键入一个名称，确保它以 .ahk 结尾。例如：Test.ahk
4. 鼠标右键点击此文件并选择 Edit Script。
5. 在一个新的空行，键入下述内容：

```
#space::Run www.google.com
```

符号 # 表示 Windows 键，因此 #space 意味着按住 Windows 键然后按下空格键来激活一个热键。符号 :: 意味着每次按下此热键时，随后的命令将会被执行，在此例中将转到 Google 网站。要试用此脚本，按下述内容继续操作：

1. 保存并关闭此文件。
2. 在 Windows 资源管理器中，鼠标双击来启动脚本。一个新的系统托盘图标出现。
3. 按住 Windows 键并按下空格键。一个网页在默认浏览器中打开。
4. 要退出或编辑此脚本，鼠标右键点击它的系统托盘图标。

注意：多个脚本能被同时运行，每个带有它自己的托盘图标。此外，每个脚本能拥有多个[热键\(See 4.\)](#)和[热字串\(See 5.\)](#)。

Run(See 24.5) 命令用来启动一个程序、文档、URL(统一资源定位符) 或者快捷方式。这里有一些普通的例子：

```
Run Notepad
Run C:\My Documents\Address List.doc
Run C:\My Documents\My Shortcut.lnk
Run www.yahoo.com
Run mailto:someone@somedomain.com
```

一个热键通过包含一个[热键标记\(See 4.\)](#)能被分配给以上任何一个例子。在下面的第一个例子中，被分配的热键是 Win+N，而在第二个里是 Control+Alt+C：

```
#n::Run Notepad
^!c::Run calc.exe
```

上面的例子被称为单行热键，因为每个热键仅由单个命令组成。要通过一个热键执行多个命令，把首行放在热键定义的下面并让末行有一个 [return\(See 17.19\)](#)。例如：

```
#n::
Run http://www.google.com
```

```
Run Notepad.exe
return
```

如果要运行的程序或文档没有与系统结合在一起，指定它的完整路径来使它启动：

```
Run %A_ProgramFiles%\Winamp\Winamp.exe
```

在上面的例子里，`%A_ProgramFiles%` 是一个[内置变量](#)(See 9.)。通过使用它而不是类似像 `C:\Program Files`，脚本将更便携，意味着它将很有可能在其他计算机上运行。注意：命令的名称和变量都不区分大小写。例如，"Run" 和 "run" 是一样的，而且 "`A_ProgramFiles`" 也和 "`a_programfiles`" 是一样的。

要让脚本在继续执行前等待程序或文档关闭，使用 [RunWait](#)(See 24.5) 代替 Run。在下述例子中，[MsgBox](#)(See 19.11) 命令将不会执行，直到用户关闭了记事本之后：

```
RunWait Notepad
MsgBox 用户已完成(记事本已被关闭)。
```

要学习更多关于启动程序的内容--像传递参数、指定工作目录和了解一个程序的退出代码--点击[这里](#)(See 30.37)。

键击通过使用 [Send](#)(See 20.12) 命令发送到活动的(最前面的)窗口。在下述例子中，`Win+S` 变为一个热键去键入一个签名(确保在按下 `Win+S` 前，像一个编辑器或起草电子邮件信息的窗口是活动的)：

```
#s::
Send Sincerely,{Enter}John Smith
return
```

在上面的例子里，所有的字符精确地发送，除了 `{Enter}`，其模拟按下了 Enter 键。下一个例子说明了其他一些常用的特殊字符：

```
Send ^c!{tab}pasted:^v
```

上面这行发送了一个 `Control+C` 紧跟一个 `Alt+Tab` 紧跟字串 "pasted:" 紧跟一个 `Control+V`。要得到一个完整的特殊字符和按键的列表，请见 [Send](#)(See 20.12) 命令。

最后，键击也能对你键入的缩写做出反应而被发送，这称为[热字串](#)(See 5.)。例如，每当你键入 `Btw` 紧跟一个空格或逗号，下述这行会把它替换为 "By the way"：

```
::btw::by the way
```

鼠标点击：要对一个窗口发送一个鼠标点击，首先必须要定义将要点击的位置的 X 和 Y 坐标轴。这个能用 `AutoScriptWriter` 或 `Window Spy` 来完成，它们已包含在 `AutoHotkey` 中。下述步骤是适用于 `Window Spy` 的方法：

1. 从脚本的托盘图标菜单或系统开始菜单启动 `Window Spy`。
2. 通过鼠标点击感兴趣的窗口的标题栏、`alt` 标签栏或其他方法来激活窗口（被设计好的 `Window Spy` 将处于 "置顶"）。

3. 在目标窗口中移动鼠标指针到想要的位置并记下由 **Window Spy** 显示的鼠标坐标轴 (或按下 Shift-Alt-Tab 来激活 **Window Spy** 以便 "冻结的" 坐标轴能被复制和粘贴)。
 4. 在 **Click**(See 23.1) 命令中使用上面发现的坐标轴。下述例子点击了鼠标左键:
- Click 112, 223**

要移动鼠标而不是点击, 请用 **MouseMove**(See 23.6)。要拖选鼠标, 请用 **MouseClickDrag**(See 23.4)。

要激活一个窗口 (使它在最前面), 使用 **WinActivate**(See 28.12)。要探测一个窗口是否存在, 使用 **IfWinExist**(See 28.7) 或 **WinWait**(See 28.32)。下述的例子阐明了这些命令:

```
IfWinExist 无标题 - 记事本
{
    WinActivate
}
else
{
    Run Notepad
    WinWait 无标题 - 记事本
    WinActivate
}
```

上面的例子首先搜索标题以"无标题 - 记事本" (区分大小写) 开始的任何存在的窗口。如果这样的一个窗口被找到, 它将被激活。否则, 记事本被启动并且脚本等待无标题窗口出现, 那时它将被激活。上面的例子也运用了[最近找到的窗口](#)(See 30.2)来避免需要在每个 **WinActivate** 右边指定窗口的标题。

其他一些常用的窗口命令是:

- **IfWinActive**(See 28.6): 检查指定的窗口当前是否激活。
- **WinWaitActive**(See 28.33): 等待指定的窗口变为活动的窗口 (典型地使用在发送一个窗口激活键击之后, 例如按下 Control-F 来“查找”。
- **WinClose**(See 28.14): 关闭指定的窗口。
- **WinMove**(See 28.27): 移动及(或)调整指定窗口的大小。
- **WinMinimize**(See 28.25), **WinMaximize**(See 28.24), **WinRestore**(See 28.28): 分别最小化、最大化或还原指定的窗口。

下述例子显示一个带两个按钮的对话框 (是 和 否):

```
MsgBox(See 19.11), 4, , 你想继续吗?
IfMsgBox(See 19.9), No
    return
;否则, 用户选择了是。
```

MsgBox 你按了是。

使用 [InputBox\(See 19.10\)](#) 命令来提示用户键入一个字串。使用 [FileSelectFile\(See 16.23\)](#) 或 [FileSelectFolder\(See 16.24\)](#) 来让用户选择一个文件或文件夹。为了更高级的任务，使用 [Gui\(See 19.3\)](#) 命令来创建自定义用户界面和数据输入表单。

提示：你也许已经从其他例子中注意到任何命令的第一个逗号可以被省略（除了当第一个参数为空或者命令是单独在一个[连续部分\(See 8.\)](#)的顶端）。例如：

MsgBox 这是可以的。

MsgBox, 这也是可以的（它有一个明显的逗号）。

一个**变量**是脚本储存文本或数字的一个内存区域。尽管所有变量将它们的内容存为字符串，一个仅包含数位的变量会在一个数学运算或比较需要时，自动地转换为一个数字。相反地，当一个数学运算的结果需要被存进一个变量时会被转换回一个字串。

除了在[函数\(See 10.\)](#)里的局部变量外，所有变量都是全局的；就是说，它们的内容可以在脚本的任意部分被读取或更改。另外，变量不需要声明；它们只在使用它们时存在（并且每个变量都以空或空白开始）。

要指定一个字串给一个变量，参照这些例子：

```
MyVar1 = 123
MyVar2 = 我的字串
```

要将一个变量的内容和一个数字或字串比较，参照这些例子：

```
if MyVar2 = 我的字串
{
    MsgBox MyVar2 包含字串"我的字串"。
}
if MyVar1 >= 100
{
    MsgBox MyVar1 包含 %MyVar1%， 它是一个大于或等于 100 的数字。
}
```

在上面的 **MsgBox** 行，注意那第二个出现的 *MyVar1* 被括在百分号内。它在那里显示了 *MyVar1* 的内容。相同的手法能用来复制一个变量的内容给另一个变量。例如：

```
MyVarConcatenated = %MyVar1% %MyVar2%
```

上面这行在变量 **MyVarConcatenated** 中储存了字串 "123 我的字串"（不含引号）。

要比较一个变量和另一个变量的内容的用法，参考这个例子：

```
if (ItemCount > ItemLimit)
{
    MsgBox 在 ItemCount 里的值，是 %ItemCount%， 比 %ItemLimit% 大。
}
```

注意上面例子的首行包含圆括号。圆括号表示那 `if`-语句包含一个[表达式\(See 9.\)](#)。没有它们的话，那行将被认作一个“非表达式的 `if`-语句”，并且因此它将需要让 `ItemLimit` 以百分号围住。这样的 `IF` 语句被限制为一个单独的比较运算符；也就是说，它们不能包含数学运算符或者像 “`AND`” 和 “`OR`” 这样的联合运算符。

数学：要执行一个数学运算，使用冒号-等号运算符 `(:=)` 来指定一个[表达式\(See 9.\)](#)的结果给一个变量，像下面的例子：

```
NetPrice := Price * (1 - Discount/100)
```

请见[表达式\(See 9.\)](#)来获得一个完整的数学运算符列表。

剪贴板：名为 `Clipboard` 的变量是特殊变量，因为它包含了当前 `Windows` 剪贴板里的文本。即使如此，它也能像一个普通变量被使用。例如，下面这行将显示当前剪贴板的内容：

```
MsgBox %clipboard%
```

要改变剪贴板内容，参考下述例子，它用新的文本替换了当前剪贴板里的内容：

```
clipboard = 一行文本。`r`n第二行文本。`r`n
```

在上面这行，`r 和 `n (重音符紧跟着字母 "r" 或 "n") 被用来标明两个特殊的字符：回车和换行。这两个字符重新开始一个文本行就像用户按下了 `Enter` 键。

要追加文本到剪贴板(或任何其他变量)，参照这个例子：

```
clipboard = %clipboard% 这里是追加的文本。
```

详见[剪贴板\(See 30.6\)](#)和[变量\(See 9.\)](#)章节。

要连续地不止一次执行某物，一个 `loop`([See 17.12](#)) 就是答案。下述 `loop` 显示了三次 `MessageBox`([See 19.11](#))：

```
Loop 3
{
    MsgBox 此窗口将被显示三次。
}
```

你也能在单词 `Loop` 后面指定一个变量，它在脚本内部的某处其循环的次数被定义时是有用的：

```
Loop %RunCount%
{
    Run C:\Check Server Status.exe
    Sleep 60000 ;暂停 60 秒。
}
```

在上面例子里，`loop` 将执行指定的次数，除非 `RunCount` 是 0，在这种情况下，`loop` 整个被跳过。

一个 **loop** 也可以终止它自己，当一个或多个条件改变时。下述例子重复地点击鼠标左键，当用户按住 **F1** 键时：

```
$F1:: ;让 F1 键成为一个热键（$ 符号有助于下面 GetKeyState 的 "P" 模式）。
Loop ;由于没有指定数字给它，这是一个无限循环，除非在内部碰到 "break" 或 "return"。
{
    if not GetKeyState("F1", "P") ;如果此状态为真，则用户已经物理地释放了 F1 键。
        break ;打破循环。
    ;否则（由于上面没有“打破”），保持点击鼠标左键。
    Click ;在当前光标位置点击鼠标左键。
}
return
```

上面的例子和一个有时称为 "**while...do**" 的循环在功能上相同。短语 "**while...do**" 指的是当某个或某些条件仍然为真时，此循环重复地做某事的事实。在本例中，当 **F1** 键被按住的时候，此循环继续点击鼠标左键。当用户释放 **F1** 键时，此循环探测到此动作并通过 **break(See 17.3)** 命令终止了它自己。**Break** 引起执行跳到循环的闭合大括号的下一行。

上面给出的例子是一般用途的循环。为了更具体的需要，可参考下述的某个循环：

文件读写循环(See 16.32): 在一个文本文件内一次检索一行。此命令能用来在一行行的基础上将一个文件转换成不同的格式。它也能被用来按你的标准逐行搜索。

文件和文件夹循环(See 16.31): 一次检索一个指定的文件或文件夹。它允许每个文件或文件夹执行一个满足你的标准的操作。

分解循环(See 17.14): 从一个字串里一次检索一个子字串。它允许一个像“红、绿、蓝”这样的字串被简单地打断成它的三个组成部分。

注册表循环(See 17.16): 一次检索一个指定的注册表子键的内容。

要添加文本到一个文件的末尾（或创建一个新文件），使用在下述例子里出现的 **FileAppend(See 16.4)**。注意它在后面使用 `n (换行) 来启用一个新的文本行：

```
FileAppend, 添加一行文本。`n, C:\My Documents\My Text File.txt
```

要覆盖一个存在的文件，在 **FileAppend** 之前使用 **FileDelete(See 16.9)**。例如：

```
FileDelete, C:\My Documents\My Text File.txt
```

其他一些常用的文件和文件夹命令是：

- **FileRead(See 16.19)**: 读取一个文件的整个内容给一个变量。

- [文件读取循环](#)(See 16.32): 在一个文本文件里逐行检索。
- [IfExist](#)(See 16.27): 根据一个文件或文件夹的是否存在而做出决定。
- [FileSelectFile](#)(See 16.23) 和 [FileSelectFolder](#)(See 16.24): 为用户选择一个文件或文件夹显示一个对话框。
- [FileDelete](#)(See 16.9)/[FileRecycle](#)(See 16.20): 删除或回收一个或多个文件。使用[FileRemoveDir](#)(See 16.22) 来删除一整个文件夹。
- [FileCopy](#)(See 16.5)/[FileMove](#)(See 16.16): 复制或移动一个或多个文件。使用[FileCopyDir](#)(See 16.6)/[FileMoveDir](#)(See 16.17) 来复制或移动一整个文件夹。
- [文件和文件夹循环](#)(See 16.31): 一次检索一个被包含在一个文件夹下的文件和文件夹。
- [FileSetAttrib](#)(See 16.25) 和 [FileSetTime](#)(See 16.26): 改变一个或多个文件的属性或时间标记。
- [IniRead](#)(See 16.29), [IniWrite](#)(See 16.30) 和 [IniDelete](#)(See 16.28): 创建、访问和修改标准格式的 INI 文件。
- [RegRead](#)(See 25.3), [RegWrite](#)(See 25.4), [RegDelete](#)(See 25.2) 和 [Registry Loop](#)(See 17.16): 和 Windows 注册表一起工作。

获取每个命令的概述，请见[命令列表](#)(See 12.)。

翻译：天堂之门 menk33@163.com 2008 年 12 月 11 日

3. FAQ (常见问答)

语法

- 何时才能在命令和它们的参数中使用引号？
- 何时需要将变量名称装入百分号？
- 何时才应该将百分号和逗号转义？

常见任务

- 为什么我脚本中的某些行总是无法执行？
- 为什么 Run 命令不能启动我的游戏或程序？
- 怎样获取命令行操作的输出？
- 怎样才能让一个脚本关闭、暂停或挂起其它的脚本？
- 怎样才能不退出脚本却可以停止重复的动作？
- 如何在玩游戏或平时 CPU 高负荷运行时改善性能？
- 如何在任意编辑器中使用上下文相关的 AutoHotKey 命令帮助？
- 如何探测一个网页何时加载完毕？
- 怎样操作或比较日期和时间？
- 为什么在某些游戏中，Hotstrings, Send 以及 MouseClick 不起作用？

- 如何控制 Winamp，即使是在它未激活时？
- 如何改变 MsgBox 的按钮名称？

热键、热字符串和重映射

- 如何让我的热键和热字符串在 PC 启动时自动生效？
- 我在把鼠标按键作为热键使用时遇到问题。有什么建议吗？
- 如何将 Tab 和空格键定义为热键？
- 如何将按键或鼠标按钮重映射成其他键？
- 如何将热键或热字符串设置为在特定程序下专用？换句话说，我想让特定按键执行它原来的功能，在指定的窗口激活时除外。
- 如何让前置按键执行它原有的功能，而不是什么也不做？
- 如何修改或者禁用 Windows 内置的快捷键，比如 Win+U (辅助工具管理器) 和 Win+R (运行)？
- 我的数字小键盘上有个特殊的 000 键，能否将其设为热键？

何时才能在命令和它们的参数中使用引号？

双引号(")仅在[表达式\(See 9.\)](#)中有特殊的含义。在所有其它的地方，它们就像普通的字符一样被原义地对待。不过，当脚本启动一个程序或文档时，操作系统通常需要用引号把带空格的命令行参数括起来，比如此例：
Run, Notepad.exe "C:\My Documents\Address List.txt"

何时需要将变量名称装入百分号？

除了以下用**粗体**表示的情况外，总是用百分号将变量名括起来的：

- 1) 在参数里的输入或输出变量：[StringLen\(See 27.17\)](#), **OutputVar**, **InputVar**
- 2) 在赋值表达式的左侧：**Var** = 123abc
- 3) 在[传统的\(非表达式\) if 语句\(See 17.10\)](#)的左侧：If **Var1** < %Var2%
- 4) 在[表达式\(See 9.\)](#)的任意位置。例如：
If (Var1 <> Var2)
Var1 :=(See 21.12) Var2 + 100

何时才应该将百分号和逗号转义 (See 29.5) ?

原义的百分号必须通过在它们前面加重音符/反引号来[转义\(See 29.5\)](#)。例如：*MsgBox The current percentage is 25`%.* 原义的逗号也必须转义 (`,)，除了用在 [MsgBox\(See 19.11\)](#) 或任何命令的最后一个参数里时(这些情况下允许使用重音符，但不是必须的)。

当在[表达式\(See 9.\)](#)中，逗号或百分号被双引号括起时，允许使用重音符，但不是必须的。例如：**Var := "15%"**

为什么我脚本中的某些行总是无法执行？

任何想在脚本启动时立即执行的行都应该出现在脚本的顶部，要在首个 [hotkey](#)(See 4.), [hotstring](#)(See 5.) 或 [Return](#)(See 17.19) 之前。详见[自动执行部分](#)(See 8.)。

而且，一个有多行要执行的热键(See 4.)，必须将它的首行列在热键的下面，而不是同一行。例如：

```
#space:: ; Win+空格
Run Notepad
WinWaitActive 无标题 - 记事本
WinMaximize
return
```

为什么 Run(See 24.5) 命令不能启动我的游戏或程序？

有些程序需要在它们自己的目录下运行(如果不能确定时，通常最好这样做)。例如：

```
Run, %A_ProgramFiles%\Some Application\App.exe, %A_ProgramFiles%\Some Application
```

怎样获取命令行操作的输出？

测试显示，由于文件缓存的存在，对于较少量的输出，一个临时文件将生成得非常快。实际上，如果文件在使用后被立即删除，通常它并没有被真正地写到磁盘上。例如：

```
RunWait(%comspec% /c dir > C:\My Temp File.txt
FileRead, VarToContainContents, C:\My Temp File.txt
FileDelete, C:\My Temp File.txt
```

要避免使用临时文件的话(特别是如果输出较大时)，可以考虑使用 [CmdRet](#)。或者，也可用一个免费的小工具 [cb.zip](#) (4 KB)，它可以从一个命令或程序获取多达 512KB 的输出。文本被抓取到剪贴板，一个脚本可以通过[剪贴板变量](#)(See 30.6)来访问该文本。例如：

```
RunWait %comspec%(See 9.) /c dir | cb.exe
MsgBox %clipboard%
```

怎样才能让一个脚本关闭、暂停或者挂起其它的脚本？

首先，这有个关闭另一个脚本的例子：

```
DetectHiddenWindows On ;允许探测到一个隐藏的脚本主窗口。
SetTitleMatchMode 2 ;避免了需要给下面的文件指定完全路径。
WinClose Script's File Name.ahk - AutoHotkey ;按脚本的名称更新这里的标题(区分大小写)。
```

要挂起(See 17.23)或者暂停(See 17.18)另一个脚本，将上面脚本的最后一行替换为下列命令中的一个：

```
PostMessage, 0x111, 65305,,, Script's File Name.ahk - AutoHotkey ;挂起。
PostMessage, 0x111, 65306,,, Script's File Name.ahk - AutoHotkey ;暂停。
```

怎样才能不退出脚本却可以停止重复的动作？

要通过按键来暂停或恢复整个脚本，只要像下例那样给 [Pause\(See 17.18\)](#) 命令指定一个热键：

```
^!p::Pause ;按 Ctrl+Alt+P 来暂停。再按一次则恢复。
```

要想停止[循环\(See 17.12\)](#)内的动作，参考下例，这是一个可以启用和停止它自身重复动作的热键。换言之，按一次热键将启动循环，再按同个热键将停止循环。

```
#MaxThreadsPerHotkey 3
#z:: ; Win+Z 热键 (可根据你的设置修改这个热键)。
#MaxThreadsPerHotkey 1
if KeepWinZRunning ;这意味着一个潜在的线程(See 30.19)正在下面的循环中运行。
{
    KeepWinZRunning := false ;向那个线程的循环发出停止的信号。
    return ;结束此线程，以便下面的线程恢复并得知上一行所做的更改。
}
;否则：
KeepWinZRunning := true
Loop
{
    ;以下四行是你要重复的动作(可根据你的设置修改它们)：
    ToolTip, 再按一次 Win-Z 将停止本提示闪动。
    Sleep 1000
    ToolTip
    Sleep 1000
    ;但是不要修改下面剩下的内容。
    if not KeepWinZRunning ;用户通过再次按下 Win-Z 来对循环发出停止信号。
        break ;跳出此循环。
}
KeepWinZRunning := false ;重置，为下一次按热键做准备。
return
```

如何在玩游戏或平时 CPU 高负荷运行时改善性能？

当 CPU 高负荷运行时，如果脚本里的 [Hotkey\(See 4.\)](#), [Click\(See 23.1\)](#) 或者 [Send\(See 20.12\)](#) 命令明显比平时慢，那么提高脚本的进程优先级可能会有所帮助。要这样做的话，请在接近脚本的顶部包含下面这行命令：

[Process, Priority, , High\(See 24.4\)](#)

如何在任意编辑器中使用上下文相关的 AutoHotKey 命令帮助？

[这个脚本\(See 30.21\)](#)由 Rajat 创建。

如何探测一个网页何时加载完毕？

用 Internet Explorer 的话，或许最可靠的方法就是使用 DllCall 和 COM，如 www.autohotkey.com/forum/topic19256.html 所示。做一个相关提示，地址栏和状态栏的内容是可以被获取的，如 www.autohotkey.com/forum/topic19255.html 所示。

较早的，不太可靠的方法：下面例子中的技术对用微软的 Internet Explorer 浏览大多数的页面来说都会有效。相似的技术用在其它浏览器上也可能会奏效。

```
Run, http://ahkbbs.cn
MouseMove, 0, 0 ;防止状态栏显示鼠标悬停处的链接来代替“完成”。
WinWait,AutoHotKey 中文论坛
WinActivate

StatusBarWait(See 28.11), Done, 30
if ErrorLevel
    MsgBox 等待超时或者窗口被关闭。
else
    MsgBox 页面加载完毕。
```

怎样操作或比较日期和时间？

EnvAdd(See 21.1) 命令可以将 YYYYMMDDHH24MISS(See 16.26) 格式的时间字符串加上或减去一些天数、小时、分钟或者秒数。下面的例子将指定的时间减去了 7 天：

EnvAdd, VarContainingTimestamp, -7, days

要计算两个日期或时间之间的间隔，请看 EnvSub(See 21.4)，它给出了一个示例。此外，内置变量 A_Now(See 9.) 包含了当前本地的时间。最后，还有一些内置的日期/时间变量(See 9.)，以及 FormatTime(See 27.1) 命令，它可用来创建自定义的日期/时间字符串。

为什么在某些游戏中，Hotstrings(See 5.)，Send(See 20.12) 以及 Click(See 23.1) 命令不起作用？

有些游戏专门使用 DirectX。作为副作用，它们可能忽略所有模拟的键击和鼠标点击。要绕弯解决的话，尝试下列方法之一(或联合使用)：

- 通过：1) SendPlay 命令(See 20.12); 2) 使用 SendMode Play(See 20.13) 和/或 3) 热字符串选项 SP(See 5.) 来使用 SendPlay。
- 增加 SetKeyDelay(See 20.14)。例如：
SetKeyDelay, 0, 50
SetKeyDelay, 0, 50, Play
- 尝试用 ControlSend(See 20.6)，在其它 Send 模式失效的地方，它可能会有用。

如何控制 Winamp，即使是在它未激活时？

请看 Automating Winamp(See 30.20)。

如何改变 MsgBox (See 19.11) 的按钮名称？

这里有个[示例](#)(See 30.30)。

如何让我的热键和热字符串在 PC 启动时自动生效？

在开始菜单中有个文件夹叫做启动。如果你将脚本的快捷方式放入此文件夹，那么每次你启动 PC 的时候脚本将自动运行。要创建快捷方式的话：

1. 在资源管理器中选中脚本文件并按下 Control-C。
2. 右键点击开始按钮并选择“浏览所有用户”。
3. 定位到程序文件夹中的启动文件夹。
4. 从菜单栏选择[编辑 > 粘贴快捷方式](#)。脚本的快捷方式就会出现在启动文件夹中。

我在把鼠标按键作为热键使用时遇到问题。有什么建议吗？

注意[鼠标热键](#)(See 4.)当前在 Windows 95/98/Me 上不可用。在其它操作系统上，鼠标左键和右键应该可以正常地被指定(例如，"#LButton:::" 是 Win+左键的热键)。相似地，鼠标中键以及[鼠标滚轮](#)(See 7.)的转动应该也可以正常地被指定，除非鼠标驱动直接控制那些按键。

第四个按键(XButton1)和第五个按键(XButton2)也可以被指定，如果你的鼠标驱动允许它们的点击被系统[看见](#)(See 20.9)。如果它们不可见，或者你的鼠标除你用到的 5 个按键外还有更多的按键，你可以尝试配置鼠标附带的软件(有时在开始菜单或控制面板中)，每当你按下这些键后就使其发送一个键击。这样的键击随后可在脚本中被定义为热键。例如，假设你配置第四个按键来发送 Control+F1，随后在脚本里你可以通过使用 ^F1:: 来间接地将那个按键作为热键来配置。

如果你有一个五个键的鼠标，它的第四个和第五个按键不能被系统[看见](#)(See 20.9)，那么你可以尝试把你的鼠标驱动改变为操作系统内含的默认驱动。这里假设你特别的鼠标会有可用的驱动并且你可以容忍不使用由鼠标自定义软件所提供的特性。

如何将 Tab 和空格键定义为热键？

使用按键的名称 (Tab 和 Space) 而不是它们的字符。例如，#Space 是 Win+空格，^!Tab 是 Control+Alt+Tab。

如何将按键或鼠标按钮重映射成其他键？

这被写在[重映射](#)(See 6.)页面。

如何将热键(See 4.)或热字符串(See 5.)设置为在特定程序下专用？换句话说，我想让特定按键执行它原来的功能，在指定的窗口激活时除外。

首选的方法是 [#IfWinActive](#)(See 20.1.4)。例如：

```
#IfWinActive, ahk_class Notepad
^a::MsgBox 在记事本激活时你按下了 Control-A。
```

Windows 95/98/Me: 虽然上述方法奏效了，但是在记事本以外的窗口中按下 **Control-A** 将没有任何作用(甚至没有它本来的功能)。要绕弯解决这种情况，请用：

```
$^a::Send ^a ; 在 Windows 9x 上, 该热键必须首先列出。前缀 & 允许热键"发送它本身"。
#IfWinActive, ahk_class Notepad
^a::MsgBox 在记事本激活时你按下了 Control-A。
```

如何让前置按键执行它原有的功能，而不是什么也不做？

参考下面这个使用 **Numpad0** 作为前置按键的示例：

```
Numpad0 & Numpad1::MsgBox, 在按住 Numpad0 的同时你按下了 Numpad1。
```

现在，要让 **Numpad0** 在它没被用来触发上述热键的时候发送一个真正的 **Numpad0** 键击，可增加下面的热键：

```
$Numpad0::Send, {Numpad0}
```

需要用前缀 **\$** 阻止产生一个关于无限循环的警告对话框(因为热键“发送了它自己”)。此外，上述动作在**释放**按键的时候发生。

如何修改或者禁用 Windows 内置的快捷键，比如 Win+U (辅助工具管理器) 和 Win+R (运行)？

这里有些[示例](#)(See 30.12)。

我的数字小键盘上有个特殊的 000 键，能否将其设为热键？

可以，但仅适用于 Windows NT, 2000, XP 及以上版本。这个[示例脚本](#)(See 30.31)将 000 键设为等号键。你可以用你选择的行替换 "Send, ="

翻译：Iwjiee 修正：天堂之门 menk33@163.com 2008 年 11 月 13 日

4. 热键

- 介绍和简单的例子
- 热键前缀符号(修饰键)
- 上下文相关的热键
- 自定义组合键和其他特性

- 鼠标滚轮热键
- 热键技巧和备注

热键有时也被称作快捷键，因为它们能轻易地触发一个动作(例如启动一个程序或[键盘宏\(See 30.11\)](#))。在下面的例子中，热键 Win+N 被设定为启动记事本。pound sign [#] 表示 Windows 键，其被称作修饰键：

```
#n::  
Run Notepad  
return
```

在上面的最后一行，“[return\(See 17.19\)](#)”用来结束热键。不过，如果一个热键仅仅需要执行一行，那么这行可以列在双冒号的右边。换句话说，[return\(See 17.19\)](#) 可以省略：

```
#n::Run Notepad
```

要对一个热键使用多个修饰键，把它们连续地列出来(顺序没有关系)。下面的例子用 ^!s 来表示 Control+Alt+S：

```
^!s::  
Send(See 20.12) Sincerely,{enter}John Smith ;这行发送键击到激活的(最前面的)窗口。  
return
```

你可以使用下列的修饰键符号来定义热键：

符号	描述
#	Win (Windows 标识键)
!	Alt
^	Control
+	Shift
&	一个连接符可以用在任何两个键或者鼠标按键的中间，从而将它们组合成一个自定义热键。详见 下面 。这样的热键在 Windows 95/98/Me 上被忽略(未被激活)。
<	使用成对按键中左边的按键。例如 <!a 和 !a 一样，只是仅有左边的 Alt 键会触发它。这个符号在 Windows 95/98/ME 上被忽略。
>	使用成对按键中右边的按键。这个符号在 Windows 95/98/ME 上被忽略。
<^>!	AltGr (alternate graving)(译注：传说此键出现在大部分欧洲键盘上)。如果你的键盘布局中有一个代替右 Alt 键的 AltGr 键，这一系列的符号一般能用来表示 AltGr (需要 Windows NT/2k/XP 或之后版本)。例如： <^>!m::MsgBox 你按下了 AltGr+m. <^<!m::MsgBox 你按下了 LeftControl+LeftAlt+m.

	<p>或者, 要使 AltGr 它自己成为一个热键, 使用下面的热键(不用像上面出现的任何一个热键):</p> <pre>LControl & RAlt::MsgBox 你按了 AltGr 它本身。</pre>
*	<p>通配符: 即使附加的修饰键被按住也激发热键。这常被用来协同重映射(See 6.)按键或按钮。例如:</p> <pre>*#c::Run Calc.exe ; Win+C, Shift+Win+C, Ctrl+Win+C 等全部会触发这个热键。 *ScrollLock::Run Notepad ;即使当修饰键按住时, 按下 Scrolllock 键也将触发这个热键。</pre> <p>这个符号在 Windows 95/98/ME 上被忽略。</p>
~	<p>当激发热键时, 按键的原来的功能不会被屏蔽(被操作系统隐藏)。在下面的两个例子中, 用户的鼠标按钮点击将被发送到激活的窗口:</p> <pre>~RButton::MsgBox 你点击了鼠标右键。 ~RButton & C::MsgBox 你在按住鼠标右键的同时按下了 C 键。</pre> <p>注意: 1) 和其他前缀符号不同, 波浪符前缀被允许在一个热键的某些变体(See 20.1.4)上出现但在其他部分不存在; 2) 那些代替 alt-tab 的特殊热键永远忽略波浪符前缀; 3) 波浪符前缀在 Windows 95/98/ME 上被忽略。</p>
\$	<p>这符号通常仅仅在脚本使用 Send(See 20.12) 命令发送包含了热键自身的按键时才有必要使用, 否则这可能导致发送热键触发它自己。\$ 前缀确切的特性变化取决于操作系统:</p> <p>在 Windows NT4/2k/XP 或之后版本上: \$ 前缀强制使用键盘钩子(See 20.2)来执行这个热键, 作为一个副作用它防止了 Send(See 20.12) 命令触发热键。\$ 前缀等效于在此热键定义的上面某处指定了 #UseHook(See 20.1.9)。</p> <p>在 Windows 95/98/Me 上: 热键在它的线程(See 30.19)执行期间被禁用, 之后重新启用。作为一个副作用, 如果 #MaxThreadsPerHotkey(See 20.1.8) 设成高于 1, 它也将表现得像被这个热键设为 1 一样。</p>
UP	<p>单词 UP 可以跟在一个热键名称后面来促使热键在松开按键时激发而不是在按下按键时。下面的例子将 LWin 重映射(See 6.)成 LControl:</p> <pre>*LWin::Send {LControl Down} *LWin Up::Send {LControl Up}</pre> <p>"Up" 也可以和一般的热键一起使用比如在这个例子中: ^!r Up::MsgBox 你按下并松开了 Ctrl+Alt+R 。它也可以和 组合热键 一起用(例如 F1 & e Up::)</p> <p>限制: 1) "Up" 不能和游戏杆按钮(See 7.)一起用; 2) "Up" 需要 Windows NT4/2000/XP 或之后版本; 3) 不和一个普通/按下的配对热键一起用的 "Up" 热键将完全接管那个热键从而防止它卡在接下的状态。唯一能防止这种情况发生的就是添加一个波浪符前缀 (例如 ~LControl up::)</p> <p>作个相关的提示, 和上面相似的一个技术是让一个热键成为一个前缀键。尽管热键会在松开时激发是</p>

个有利情况，但热键仅在你如果按住它的同时不去按其他任何键的时候才会激发。例如：

`LControl & F1::return ;通过让左 control 键在 "&" 前面至少使用一次来把它变成一个前缀。`

`LControl::MsgBox` 你在没有用 `LControl` 去修饰其他任何键的情况下松开了它。

(请看[按键列表\(See 7.\)](#)获取一个完整的键盘按键和鼠标/操纵杆按钮列表)

多个热键可以被垂直地堆放起来让它们执行同样的动作。例如：

`^Numpad0::`

`^Numpad1::`

`MsgBox` 按了 `Control+Numpad0` 或 `Control+Numpad1` 都会显示这个消息。

`return`

通过让热键不作任何动作可以在整个操作系统中禁用一个键或者组合键。下面的例子禁用了右边的 Windows 键：

`RWin::return`

`#IfWinActive/Exist`([See 20.1.4](#)) 指令可以被用来让一个热键视激活或存在的窗口的类型而执行一个不同的动作(或根本无动静)。例如：

`#IfWinActive, ahk_class Notepad`

`^a::MsgBox` 你在记事本激活时按了 `Ctrl-A`。在其他任何窗口按 `Ctrl-A` 将只传递 `Ctrl-A` 键击到那个窗口。

`#c::MsgBox` 你在记事本激活时按了 `Win-C`。

`#IfWinActive`

`#c::MsgBox` 你在除了记事本外的任意窗口激活时按了 `Win-C`。

你可以通过在两个键之间使用 " & " 来自定义一个组合键(除了操纵杆按钮)。在下面的例子中，你能够按住 `Numpad0` 之后再按第二个键来触发热键：

`Numpad0 & Numpad1::MsgBox` 你在按住 `Numpad0` 的同时按下了 `Numpad1`。

`Numpad0 & Numpad2::Run Notepad`

在上面的例子中，`Numpad0` 变成一个**前缀键**；而且也导致了通过它去按 `Numpad0` 它自己的时候失去了它原本/天生的功能。要避免这种情况，脚本可以像下面中的一种那样来配置 `Numpad0` 去执行一个新的动作：

`Numpad0::WinMaximize A ;最大化激活的/前台的窗口。`

`Numpad0::Send {Numpad0} ;让 Numpad0 松开后去再生一个 Numpad0 键击。请见下面的内容。`

上面的热键之一的存在都促使 Numpad0 松开后去执行指示的动作，但仅在 Numpad0 被按住时如果你没有去按其他任何键的情况下。

Numlock, Capslock 和 Scrolllock: 这些按键可以被强制变成 "AlwaysOn" 或 "AlwaysOff"。例如：
`SetNumlockState(See 20.15) AlwaysOn`

覆盖 Explorer 的热键: Windows 的内置热键例如 Win-E (#e) 和 Win-R (#r) 可以通过在脚本里将它们指定为一个动作来简单地单独覆盖。详见[覆盖页面\(See 30.12\)](#)。

替换 Alt-Tab: 热键能通过 alt-tab 提供一种交替的效果。例如，下面的两个热键可让你用右手来 alt-tab：

`RControl & RShift::AltTab ;按住右-control 然后反复地按右-shift 来向前移动。`

`RControl & Enter::ShiftAltTab ;甚至不需要释放右-control，直接按 Enter 键来反向移动。`

详见[Alt-Tab](#)。

通过键名 WheelDown 和 WheelUp 可以支持在调节鼠标滚轮的时候激发热键。WheelLeft 和 WheelRight 需要 v1.0.48+ 以上支持，但是在 Windows Vista 或以上没影响。以下是 鼠标滚轮热键的例子：

`MButton & WheelDown::MsgBox 你在按住鼠标中键的时候向后滚动了滚轮。`

`^!WheelUp::MsgBox 你在按住 Control+Alt 的时候向前旋转了滚轮。`

在 v1.0.43.03+，内置变量 **A_EventInfo** 包含了通过调节滚轮得到的触点次数，它一般情况是 1。然而，在下面的情况下，A_EventInfo 可能大于或小于 1：如果鼠标硬件报告的距离不足一个刻痕(你滚动滚轮感觉到的一下)，A_EventInfo 可能包含 0；当滚轮被快速地滚动(取决于鼠标类型)，A_EventInfo 可能大于 1。例子用法：`~WheelDown::ToolTip %A_EventInfo%`

鼠标滚轮的一些最有用的热键涉及交替窗口的文本滚动模式。例如，下面的一对热键在你按住左边的 Control 键并调节滚轮时用水平地滚动代替了垂直地滚动：

`~LControl & WheelUp:: ;向左滚动。`

`ControlGetFocus, fcontrol, A`

`Loop 2 ;<-- 调大这个数值来快速滚动。`

`SendMessage, 0x114, 0, 0, %fcontrol%, A ;0x114 是 WM_HSCROLL, 它之后的 0 是 SB_LINELEFT。`

`return`

`~LControl & WheelDown:: ;向右滚动。`

`ControlGetFocus, fcontrol, A`

`Loop 2 ;<-- 调大这个数值来快速滚动。`

`SendMessage, 0x114, 1, 0, %fcontrol%, A ;0x114 是 WM_HSCROLL, 它之后的 1 是`

```
SB_LINERIGHT.
```

```
return
```

最后提下，由于鼠标滚轮热键只产生按下事件(从来没有弹起事件)，它们不能被用作弹起的按键热键。

依靠 `Numlock` 的状态，每个数字键区的按键能启动两种不同的热键子程序。或者，一个数字键区按键能不管 `Numlock` 的状态而启动相同的子程序。例如：

```
NumpadEnd::  
Numpad1::  
MsgBox, 不管 Numlock 是否打开，这个热键都将被启用。  
return
```

如果波浪操作符 (~) 和一个前缀键即使一起使用了一次，前缀键也总是会被发送到激活的窗口。例如，下面两个热键中，激活的窗口会接收到所有的右键点击即使只有一个热键定义包含了波浪符：

```
~RButton & LButton::MsgBox 你在按住鼠标右键的同时按了鼠标左键。  
RButton & WheelUp::MsgBox 你在按住鼠标右键的同时向前推动了滚轮。
```

`Suspend`(See 17.23) 命令能临时禁用所有热键除了你要免除的那些外。要得到更多的选择性，使用 `#IfWinActive/Exist`(See 20.1.4)。

通过使用 `Hotkey`(See 20.1.10) 命令，热键能在脚本运行过程中被动态地创建。`Hotkey` 命令也能单独地修改、禁用或启用已存在的脚本热键。

操纵杆热键目前不支持修饰键前缀例如 ^ (Control) 和 # (Win)。不过，你可以使用 `GetKeyState`(See 20.7) 模仿下面例子中展示的这个效果：

```
Joy2::  
if not GetKeyState("Control");左边或右边的 Control 都没有按下。  
return ;也就是什么也不做。  
MsgBox 你在按住 Control 键的同时按了首个操纵杆的第二个按钮。  
return
```

当一个热键在继续前要等它的修饰键被松开的时候可能需要一些时间。参考下面的例子：

```
^!s::Send {Delete}
```

按下 Control-Alt-S 会导致系统表现得像你按了 Control-Alt-Delete 一样(由于系统的侵略性的 Ctrl-Alt-Delete 探测)。要绕弯解决这种情况，使用 `KeyWait`(See 20.10) 来等待那些键被松开；例如：

```
^!s::  
KeyWait Control  
KeyWait Alt  
Send {Delete}  
return
```

一个热键标记能被用作 `Gosub`(See 17.8) 或 `Goto`(See 17.9) 要跳转的目标。例如： `Gosub ^!s`

一个热键常用来开始和结束一个重复的动作，比如一系列的键击或鼠标点击。要得到这样一个例子，请看这个 [FAQ 主题\(See 3.\)](#)。

最后，每个脚本都是类似的[多线程\(See 30.19\)](#)，其允许在前一个热键子程序还在运行时，一个新的热键也能被启动。例如，即使当一个 [MsgBox\(See 19.11\)](#) 被当前热键调用显示时，新的热键也能被启动。

每个 **Alt-Tab** 热键必须是一个双键组合，其往往通过连接符 (&) 实现。在下面的例子中，你将按住右边的 Alt 键并按下 J 或 K 来浏览 alt-tab 菜单：

```
RAlt & j::AltTab  
RAlt & k::ShiftAltTab
```

AltTab 和 **ShiftAltTab** 是两个特殊的命令，它们仅当和热键使用在同一行时才被识别。这里有一个完整的列表：

AltTab: 如果 alt-tab 菜单可见，在菜单中前移。否则，显示菜单(仅在热键是一个用 "&" 连接的双键组合才行；否则，它什么也不做)。

ShiftAltTab: 和上面的一样，除了在菜单中是向后移动。

AltTabAndMenu: 如果 alt-tab 菜单可见，在菜单中前移。否则，显示菜单。

AltTabMenuDismiss: 关闭 Alt-tab 菜单。

要阐明上面的命令，鼠标滚轮可以来整个代替 Alt-tab。让下面的热键生效时，点击鼠标中键显示菜单并且调节滚轮可在它里面来导航：

```
MButton::AltTabMenu  
WheelDown::AltTab  
WheelUp::ShiftAltTab
```

要取消一个热键调用的 Alt-tab 菜单而不激活选中的窗口，使用一个像下面那样的热键。它可能需要依赖后面的条件来调整：1) 借助原本显示的 alt-tab 菜单；2) 脚本是否安装了[键盘钩子\(See 20.2\)](#)。

```
LCtrl & CapsLock::AltTab  
!MButton:: ;鼠标中键。前缀 ! 使它在按住 Alt 键时激发(如果 alt-tab 菜单可见，Alt 键就是按住的)。  
IfWinExist ahk_class #32771 ;指示 alt-tab 菜单当前是否出现在屏幕上。  
    Send !{Escape}{Alt up}  
return
```

目前，所有特殊的 Alt-tab 动作必须像上面的例子一样直接指定到一个热键(也就是它们不能像命令那样使用)。另外，alt-tab 菜单的存在能通过 [IfWinExist ahk_class #32771](#) 探测到。

自定义 alt-tab 动作也能通过热键来创建。在下面的例子中，你将按 F1 来显示菜单并原先在里面前移。然后你将按 F2 来激活选中的窗口(或按 Escape 来取消)：

```
*F1::Send {Alt down}{tab} ;在这里需要星号。译注：因为按了后 Alt 处于 Down 状态，再次激发热键只能靠星号匹配 Alt 这种修饰键。
```

```

!F2::Send {Alt up} ;松开 Alt 键激活选中的窗口。
~*Escape:::
IfWinExist ahk_class #32771
    Send {Escape}{Alt up} ;取消菜单而不激活选中的窗口。
return

```

翻译：天堂之门 menk33@163.com 2008 年 11 月 22 日

5. 热字串 和 自动替换

注意：热字符串需要 Windows NT/2000/XP 或更高版本的支持。（[译注：关于中文支持，请看本页末尾部分。](#)）

虽然热字符串主要被用来在你输入它们的时候扩展缩写(自动替换)，但它们也可以用来启动任何脚本化的动作。在这方面它们与[热键\(See 4.\)](#)很相似，只不过它们一般由多个字符组成(也就是字符串)。

要定义一个热字符串，只需要把触发缩写放在两个冒号之间即可，像这样：

```
::btw::by the way
```

上例中，当你输入缩写 `btw` 时，就会被自动地替换成 "by the way"(不过默认情况下，你必须在输入 `btw` 之后再输入一个[结束字符](#)，比如一个空格、句点或者回车)。

因为输入的文本被自动地清除并被替换为第二个双冒号后面指定的字符串，所以上面的 "by the way" 例子被称为一个自动替换的热字符串。相比起来，热字符串也能被定义成执行任何自定义的动作，如下所示。注意命令必须在热字符串的下面：

```

::btw::

MsgBox 你输入了 "btw"。

return

::*;

*:]:d:: ;这个热字符串通过下面的命令，把 "]d" 替换成了当前的日期和时间。

FormatTime(See 27.1), CurrentDateTime,, M/d/yyyy h:mm tt ;看起来将是 9/1/2005
3:53 PM 这样

SendInput %CurrentDateTime% ;译注：输出的字符串末尾的 AM 或 PM 在中文操作系统下
会用“上午”或“下午”来代替。但这两个中文字符会输出为乱码。关于中文支持，请看文末。

return

```

尽管上面的两个例子不是自动替换的热字符串，但默认情况下，你输入的缩写也将被清除。这是通过自动地退格来实现的，但可以通过 [b0 选项](#)来禁用。

除非星号选项已经生效，要不然你必须在一个热字符串的缩写后面输入一个结束字符才能触发它。结束字符初始由下列字符组成: `-(){}';"/\,.?!`n `t` (注意 `n 是回车, `t 是 Tab, 在 `n 和 `t 之间有个空格)。可以通过编辑下例来更改这组字符, 这将为所有的热字符串设置结束字符, 而不只是影响此指令下面的热字符串:

```
#Hotstring EndChars -(){}';"/\,.?!`n `t
```

可以使用以下两种方式改变热字符串的默认行为:

1. `#Hotstring`(See 20.1.3) 指令, 可以影响脚本中它那个点下面的所有热字符串。下例将使 C 和 R 选项生效:

```
#Hotstring c r
```

2. 把选项放在第一个双冒号之间。下例使 C 和 * 选项在当前热字符串中生效:

```
:c*:j@::john@somedomain.com ;区分大小写并且“不需要使用结束字符”。
```

下面列出了每个选项的描述。当用上面的方法指定多个选项时, 可以在它们之间选择性地包含空格。(译注: 我会将下列字母选项对应的英文关键词用绿色标注, 以帮助记忆!)

***** (星号): 不需要结束字符(比如空格、句点或回车)来触发热字符串。例如:

```
:*:j@::jsmith@somedomain.com
```

上例中, 当你一输入 @ 字符时, 它就会送出它的替换字串。如果在 `#Hotstring` 指令(See 20.1.3)中使用了它, 可以用 ***0** 来关闭这个选项。

? (问号): 就算热字符串是在另一个词里, 也可以被触发; 就是说, 在字符串被混排前立即输入字符。例如, 如果 `:?:al::airline` 是一个热字符串, 输入 "practial" 时将会产生 "practiairline" (译注: 热字符串的缩写必须在另一个词的结尾)。可用 **?0** 关闭此选项。

B0 (B 后跟一个零。**Backspace 0**): 不使用自动退格来清除你输入的缩写。之后可以使用一个 **B** 再启用之前被关掉的退格功能。脚本也可以通过 `{bs 5}` 发送 5 个退格键来自行退格。类似地, 能通过 `{left 5}` 发送左方向键击。例如, 下面的热字符串制造出 `` 并把光标向左移动 5 个位置(因此它处在标签之间了):

```
:*b0:<em>::</em>{left 5}
```

C (**Case sensitive**): 区分大小写: 当你输入缩写的时候, 它必须和脚本中定义的大小写严格匹配。可用 **C0** 来关闭大小写敏感性。

C1: 不遵守输入的大小写。使用此选项可以让**自动替换热字符串**不区分大小写并防止热字符串遵守你实际输入的字符的大小写。如果你输入了全部大写的缩写, 遵守大小写的热字符串(这是默认的)将用全部大写的方式制造它们的替换文本。如果你只输入了首个大写的字母, 替换字串的首个字母(如果是字母的话)也将是大写的。如果你以任何其它方式输入大小写, 替换字串将只严格地按脚本定义的那样发送。当使用 `#Hotstring` 指令(See 20.1.3)时, **C0** 可被用来关闭此选项, 这能使热字符串重新遵守大小写。

Kn (Key-delay n): 按键延迟：这个罕用的选项用来设置自动退格或者[自动替换](#)产生的键击之间的延迟。给 n 设定新的延迟；例如，指定 k10 将产生 10 毫秒的延迟，k-1 则没有延迟。本选项实际的表现取决于当前生效的[发送模式](#)是哪个：

- **SI (SendInput):** 因为这种模式是没有延迟的，所以按键延迟将被忽略。在 [SendInput](#) 模式[无效](#)(See 20.12)的情况下例外，这时热字符串转而采用到下面的 [SendPlay](#) 模式(它会遵从按键延迟)。
- **SP (SendPlay):** 默认延迟为零，与 [SendPlay](#) 模式下的 -1(无延迟) 相同。这种模式下，延迟实际上是一个[按键时长](#)(See 20.14)，而非键击之间的延迟。
- **SE (SendEvent):** 默认延迟为零。多数情况下建议值为零，因为它快速并且与其它进程配合得较好(由于内部会做一个 [Sleep 0](#)(See 17.22))。指定 k-1 彻底禁止延迟，如果你的 CPU 经常高负载运作，让自动替换执行的更快这样会有所帮助。当设为 -1 时，脚本的进程优先级就成为击键发送速度的重要因素。要提高脚本的优先级，请用 [Process](#)(See 24.4), *Priority,, High*。

O (Omit): 在生成替换时，省略掉[自动替换热字符串](#)的结束字符。这在你仍需要一个结束字符来明确一个热字符串，但实际上又不想让结束字符显示在屏幕上的时候会很有用。例如，假设 :o:ar:::aristocrat 是个热字符串，在输入 "ar" 后跟空格时将会生成末尾不带空格的 "aristocrat"，这就使你在输入一个单词的复数或所有格时，不用再退格了。可用 **O0** (字母 O 后跟一个零)关闭此选项。

Pn (Priority n): 热字符串的[优先级](#)(See 30.19)(比如 P1)。这个罕用的选项对[自动替换热字符串](#)无效。

R (Raw): 按照原样发送替换文本；就是说，完全按它显示的那样，而不将 {Enter} 转换为一个 ENTER 键击，也不把 ^c 转换为 Control-C，等等。对有[连续部分](#)的热字符串，此选项会自动生效。可用 **RO** 关闭此选项。

SI 或 SP 或 SE [1.0.43 及之后版本]: 设定[自动替换热字符串](#)发送它们键击的方法。这些选项互斥：每次仅有一个会生效。每个选项描述如下：

- SI 代表 [SendInput](#)(See 20.12)，由于优越的速度和可靠性，它在 1.0.43 及之后版本成为默认项。另一个好处是和下面的 [SendPlay](#) 一样，[SendInput](#) 在热字符串[自动替换文本](#)时，会延迟你输入的任何内容。这能防止替换文本被你的键击穿插。当 [SendInput](#) [无效](#)(See 20.12)时，热字符串将自动使用 [SendPlay](#) 来代替。
- SP 代表 [SendPlay](#)(See 20.12)，它能使热字符串用在很多类型的游戏中。
- SE 代表 [SendEvent](#)(See 20.12)，在 1.0.43 之前的版本中它是默认项。

Z: 这个罕用的选项将在热字符串每次触发后重置热字符串识别器。换句话说，脚本将开始等待一个全新的热字符串，不考虑你之前输入的任何内容。这能避免不必要的热字符串触发。要理解它，请看下面的热字符串：

```
:b0*?:11::
SendInput xx
return
```

因为上面没有 Z 选项，在输入 111(三个连续的 1)时将会触发热字符串两次，因为中间的 1 既是首次触发的末尾字符，又是第二次触发的开头字符。通过在 b0 前面添加字母 Z，你就要输入四个 1 来代替三个 1，从而触发两次热字符串。可用 **Z0** 关闭此选项。

使用一个[连续部分](#)(See 8.)可以让产生大量替换文本的热字符串变得更具可读性和可维护性。例如：

```
::text1:: ;译注：由于不能直接发送中文字符，故下面三行英文保留，只在每行下面通过注解形式翻译。
```

```
(
```

Any text between the top and bottom parentheses is treated literally, including commas and percent signs.

;在顶部和底部圆括号之间的任何文本都作为原义来对待，包括逗号和百分号。

By default, the hard carriage return (Enter) between the previous line and this one is also preserved.

;默认，上一行与本行之间的硬回车(Enter)也将被保留。

By default, the indentation (tab) to the left of this line is preserved.

;默认，本行左侧的缩进(tab)将被保留。

See [continuation section](#)(See 8.) for how to change these default behaviors.

;如何修改这些默认的特性，请看[连续部分](#)(See 8.)。

```
)
```

连续部分的出现也会使热字符串默认使用 [raw](#) 模式。要改变这种特殊默认模式的唯一方法，就是给每个使用连续部分的热字符串指定 [r0 选项](#)(比如 :r0:text1::)。

[#IfWinActive/Exist](#)(See 20.1.4) 指令可以使选定的热字符串产生上下文相关性。根据激活或存在的窗口类型，这样的热字符串会发送不同的替换文本，执行不同的操作或者干脆无效。例如：

```
#IfWinActive ahk_class Notepad
```

```
::btw::This replacement text will appear only in Notepad. ;这行替换文本将只出现在记事本中。
```

```
#IfWinActive
```

```
::btw::This replacement text appears in windows other than Notepad. ;这行替换文本将出现在任何不是记事本的窗口中。
```

下面这个脚本使用了热字符串来即时校正大约 4700 个常见的英文拼写错误。它还包括一个 Win+H 热键，可以方便地添加更多拼写错误：

下载: [AutoCorrect.ahk](#) (127 KB)

作者: [Jim Biancolo](#) 以及维基百科的常见拼写错误列表

当前在替换文本中不支持 %MyVar% 这样的变量引用。解决方法是，不要让这样的热字符串自动替换，而是在缩写后面使用 [SendInput\(See 20.12\)](#) 命令，和一个仅包含单词 Return 的行。

要在替换文本后发送一个额外的空格或 tab，可以把它们放在替换文本后面，但是要以一个重音符/反引号(`)作为结尾字符。例如：

```
:*:btw::By the way `
```

鼠标左键或右键的任何点击都将重置热字符串的识别器。换句话说，脚本将开始等待一个全新的热字符串，不考虑你之前输入的任何内容(如果不想这样，可以在脚本的任意位置指定一行 [#Hotstring NoMouse\(See 20.1.3\)](#))。由于每次点击通常都会移动文本插入点(光标)或者把键盘的焦点置于一个新的控件/区域。所以这种情况下，通常需要：1) 触发热字符串，即使它没有问号选项；2) 防止你在意外地鼠标点击后输入的内容与你之前输入的那些形成一个有效的缩写从而触发。因此这种“点击后重新识别”的特性是默认的。

内置变量 **A_EndChar** 里包含了你用来触发最近的非自动替换型热字符串的结束字符。如果不使用结束字符(由于使用了 * 选项)，它将为空。在让热字符串使用 Send 命令或者那些会根据你输入的结束字符而变化的行为的时候，A_EndChar 变量将很有用。要发送结束字符它自身，可用 SendRaw %A_EndChar% (之所以使用 [SendRaw\(See 20.12\)](#)，是因为普通的 Send 命令不能正确发送比如 !{} 这些字符)。

虽然在热字符串定义中的逗号、百分号和单冒号无需转义([See 29.5](#))，但那些跟在空格或 tab 后面的反引号和分号是需要的。请看[转义顺序\(See 29.5\)](#)中的完整列表。

虽然在自动替换型文本中支持 [Send 命令\(See 20.12\)](#)的特殊字符比如 {Enter} (如果未使用 raw 选项的话)，但热字符串的缩写它们自身却不使用。而是为 ENTER 键指定 `n，为 TAB 指定 `t (或一个原义的 tab，请看[转义顺序\(See 29.5\)](#)中的完整列表)。例如：当你输入 "ab" 后跟一个 tab 时，将触发热字符串 :*:ab`t::。

在热字符串定义中会原义地对待空格和 tab。例如，下面两行的结果是不同的：

```
::btw::by the way
```

```
::btw:: by the way
```

每个热字符串缩写的长度都不能超过 40 个字符。如果超出此长度，程序将会弹出警告对话框。而当[发送模式](#)是默认的 [SendInput](#) 时，热字符串的替换文本长度限制约为 5000 个字符。这个限制可以通过切换到其它[发送模式](#)而增加到 16,383 个字符。此外，在热字符串内使用 [SendPlay %MyVariable%\(See 20.12\)](#) 则可以发送一个不限制数量的文本。

热字符串的定义顺序决定了它们彼此的优先级。换句话说，如果有多个热字符串可以匹配你输入的内容，那么只有脚本中首个列出的热字符串会起作用。相关主题：[上下文相关的热字符串](#)。

为了识别热字符串，你输入的所有退格都会被算进来。不过在编辑器中使用方向键, `PageUp`, `PageDown`, `Home` 以及 `End` 来导航的话，会导致热字符串识别过程被重置。换句话说，它会开始等待一个全新的热字符串。

即使激活窗口忽略你的键击时，也能输入热字符串。换句话说，就算触发的缩写不可见，热字符串也仍然会被触发。另外，你依然可以使用退格键来撤销最近输入的键击(尽管你看不见效果)。

将首对冒号(以及其中的任何选项符号)包含在热字符串名称的前面，可以 [Gosub\(See 17.8\)](#) 或 [Goto\(See 17.9\)](#) 到一个热字符串标签。例如: `Gosub ::xyz`。不过，跳转到一个[单行\(自动替换型\)热字符串](#)的话，将什么也不会做，只会执行一个 [return\(See 17.19\)](#)。

虽然不会监视热字符串，在一个不可见的 [Input\(See 20.11\)](#) 命令过程中也不会被触发，但是可见的 `Input` 却会触发它们。

任何 `AutoHotkey` 脚本产生的键击永远都不会触发热字符串。这避免了热字符串彼此反复触发而进入无限循环的可能性。

对于某些特定的目的，[Input\(See 20.11\)](#) 命令会比热字符串更灵活。比如，它允许你的键击在激活窗口中不可见(比如一个游戏)。它还支持非字符型的结束按键，比如 `Escape`。

任何包含热字符串的脚本都会自动使用 [keyboard hook\(See 20.2\)](#)。

在下列情况中热字符串与热键具有相同的特性：

- 它们都会受到 [Suspend\(See 17.23\)](#) 命令的影响。
- 它们都遵守 [#MaxThreads\(See 20.1.6\)](#) 和 [#MaxThreadsPerHotkey\(See 20.1.8\)](#) (但不遵守 [#MaxThreadsBuffer\(See 20.1.7\)](#)) 的设置。
- 包含热字符串的脚本将会自动地[持续运行\(See 29.21\)](#)。
- 非自动替换型热字符串在启动时将会创建一个新的[线程\(See 30.19\)](#)。另外，它们会更新内置的热键变量，比如 [A_ThisHotkey\(See 9.\)](#)。

已知限制：在 `Java` 应用程序的某些系统中，热字符串可能会干扰用户键入 `diacritical` 字母(通过 `dead` 键)。解决方法是，可以临时打开 [Suspend\(See 17.23\)](#) (这会禁用所有的热字符串)。

Andreas Borutta 推荐了下面的脚本，如果你是一个重度热字符串用户，它将会很有用。通过按 `Win+H` (或你自定义的其它热键)，当前选中的文本可以被转化成一个热字符串。例如，假设你在一个字处理软件中选择了 "by the way"，那么按下 `Win+H` 将会提示你输入一个缩写(比如 `btw`)，之后就会在脚本中添加新的热字符串。随后它会重载脚本来激活热字符串。

```
#h:: ; Win+H 热键
; 获取当前选中的文本。使用剪贴板代替 "ControlGet Selected"
; 是因为它能在更多种类的编辑器(也就是字处理软件)中起作用。
; 当前剪贴板中的内容将被保存，以便之后恢复。尽管只能恢复纯文本，但总比没有的好：
AutoTrim Off ; 保留剪贴板中首尾的空白字符。
```

```

ClipboardOld = %ClipboardAll%
Clipboard = ;为检验正确，需要清空。
Send ^c
ClipWait 1
if ErrorLevel ; ClipWait 超时
    return
;将回车换行和/或换行符替换为 `n 以便使用在一个 "send-raw" 的热字符串中：
;对任何在 raw 模式下可能出问题的其他字符应用同样的操作：
StringReplace, Hotstring, Clipboard, ``, ````, All ;为避免与下列替换冲突，先作这个替换。
StringReplace, Hotstring, Hotstring, `r`n, ``r, All ;在微软 Word 等等软件中 `r 比 `n 表现得更好。
StringReplace, Hotstring, Hotstring, `n, ``r, All
StringReplace, Hotstring, Hotstring, %A_Tab%, ``t, All
StringReplace, Hotstring, Hotstring, `;, ``;, All
Clipboard = %ClipboardOld% ;恢复记事本之前的内容
;这将把输入对话框中的光标移动到更人性化的位置：
SetTimer, MoveCaret, 10
;显示提供了缺省热字符串的输入对话框：
InputBox, Hotstring, 新建热字符串，在光标处输入你的缩写。需要的话，你也可以编辑替换文本。
`n`n 示例::R:btw`::by the way,..... :R:`::%Hotstring%
if ErrorLevel ;用户选择了取消
    return
IfInString, Hotstring, :R`:::
{
    MsgBox 你没有输入缩写。热字符串不会被添加。
    return
}
;否则，添加热字符串并重新加载脚本：
FileAppend, `n%Hotstring%, %A_ScriptFullPath% ;在字符串开头放一个 `n 以防文件在它的末尾没有空行。

```

Reload

Sleep 200 ;如果重新加载成功，那么在 **Sleep** 期间就会关闭这个实例，这样就永远不会运行到下一行。

MsgBox, 4,, 刚刚添加的热字符串格式化不正确。你要打开脚本编辑么？注意，有问题的热字符串在脚本的最后一行。

```
IfMsgBox, Yes, Edit
```

```
return
```

MoveCaret:

```
IfWinNotActive, New Hotstring
```

```
return
```

;否则，将输入对话框中的光标移动到用户输入缩写的位置。

```
Send {Home}{Right 3}
```

```
SetTimer, MoveCaret, Off
```

```
return
```

注意：这部分内容是译者自己添加的。

由于 **AutoHotkey** 不能直接发送中文字符，所以可以尝试将 **Send** 命令和其他方法组合使用，从而实现发送中文的目的。

热字符串支持缩写触发命令，所以只要把下面这一系列的命令写在缩写下面就可以实现热字符串发送中文了。

方法 1：把要发送的中文赋值给剪贴板，然后发送一个粘贴动作即可自动转换字符编码：

```
ClipboardOld = %ClipboardAll% ;保留剪贴板中原来的内容
```

```
Clipboard = 需要发送的中文内容
```

```
Send ^v
```

```
Clipboard = %ClipboardOld% ;恢复剪贴板初始的内容
```

```
Return
```

方法 2：Yonken 在他的博客里有写过如何[在脚本中使用 Send 发送中文](#)。他给出的方案不依赖剪贴板，直接转换字符编码，最后通过函数的形式来调用。感兴趣的朋友可以去看看。不过转换的速度会相对慢一点。

注意，热字符串的缩写还不能使用中文，因为中文输入法输出的字符还不能被热字符串的识别器识别，解决方法未知。

翻译: wanggang999 修正: 天堂之门 menk33@163.com 2008 年 12 月 8 日

6. 鼠标与键盘的映射

本部分介绍的方法不适用于 Windows 95/98/Me 环境下的鼠标键盘映射，也不适用于对游戏操作杆进行映射。这两部分的映射方法，可参看下面的链接：

- [Windows 9x remapping](#)
- [Joystick remapping\(See 30.15\)](#)

限制: 其实可以直接通过修改 Windows 注册表来进行鼠标键盘映射，而且在多数情况下，这要比用 AutoHotKey 来映射鼠标键盘更直接和高效。当然两种方法还是各有其优缺点的，可参看 [registry remapping](#)。

映射鼠标键盘的语法为：*OriginKey:::DestinationKey*。例如，包含下面代码的脚本(See 8.)可将 'a' 键映射成 'b' 键(点击 'a' 键将产生点击 'b' 键的效果)。

```
a::b
```

上述代码不会改变 'b' 键，按下 'b' 键时仍然发送 'b' 键被按下的消息。当然，你也可以对 'b' 键进行映射(如下所示)。

```
a::b
```

```
b::a
```

上述的代码中使用的都是小写字母，但也会同时映射相应的大些字母(因此，按下 'a' 则输出 'b'，按下 'A' 则输出 'B')。如果 "::" 右边用大写字母，那么就只输出大写字母。例如，如果用下面的代码，不管你输入 'a' 还是 'A' 都只会输出 'B'(但是要保证 Capslock 没有锁住，否则输出会变成 'b')：

```
a::B
```

映射鼠标: 映射鼠标，方法一样。例如：

MButton::Shift	将鼠标中键映射成 Shift 键。(注意，这里是指按下鼠标中键而不是拨动鼠标滚轮)
XButton1::LButton	将鼠标的第四个键映射成鼠标左键。
RAlt::RButton	将右 Alt 键映射成鼠标右键。

其他常用(有用)的映射:

Capslock::Ctrl	将 Capslock 键映射成 Ctrl 键。但这样会丢失 Capslock 键，可以添加映射 +Capslock::Capslock。(这样，当你按下 Shift 键再点击 Capslock 就可以开关大写锁定了)
XButton2::^LButton	将鼠标的第五个键(XButton2)映射成 Control 加鼠标左键。
RAlt::AppsKey	将右 Alt 键映射成 Apps 键(就是打开右键菜单的键)。
RCtrl::RWin	将右 Ctrl 键映射成右 Win 键。
Ctrl::Alt	将左右 Ctrl 键映射成 Alt 键。但是这样会产生一些问题，参看 alt-tab issues 。
^x::^c	将 Ctrl-X 映射成 Ctrl-C。同时会附带将 Ctrl-Alt-X 映射成 Ctrl-Alt-C 的效果。
RWin::Return	禁用右 Win 键(在 ":" 右边返回 return (See 17.19) 就可以禁用左边的键)。

你可以试验上面所有的例子。方法是将代码拷贝到一个文本文件中(如 "Remap.ahk")，然后运行该文件。

可以在 [Key List](#)(See 7.) 查看键盘键和鼠标键的完整列表。

使用 [#IfWinActive/Exist](#)(See 20.1.4) 命令可以让映射仅在指定的窗口内生效。例如:

```
#IfWinActive ahk_class Notepad
a:::b ; 将 'a' 键映射成 'b' 键，但仅在记事本窗口内有效。
#IfWinActive ; 这个命令后面的映射和热键将会在所有的窗口内都有效。
```

任何键盘和鼠标的映射都有如下性质:

- 如果按下修饰键(就是 Ctrl 、 Alt 、 Shift 这些键)再按原始键(origin key)，会将修饰键的效果带入目标键(destination key)。例如映射 b:::a ，按下 Ctrl-B 也会产生 Ctrl-A 的效果。
- 通常 Capslock 对映射键和普通键的影响相同。
- 持续按住原始键(origin key)，目标键(destination key)也处于持续按下的状态。(一些游戏不支持键的映射，在这些游戏中所有的映射都起作用)
- 被映射的键(原始键)被按下时也会自动重复发送击键消息。(将键盘键映射成鼠标键时除外)

被映射的键(原始键)可以触发含有目标键的普通热键，但不能触发含有目标键的鼠标热键(mouse hotkey)和 [钩子热键](#)(See 20.1.9)(可以使用 [ListHotkeys](#)(See 20.1.11) 命令查看那些热键被“勾住”了)。例如，如果映射 a:::b 有效，那么按下 Ctrl-Alt-A 会触发 ^!b 热键(只要 ^!b 不是一个钩子热键)。如果 ^!b 是一个钩子热键，而你又希望按下 Ctrl-Alt-A 能够触发它，你可以将 ^!a 定义成和 ^!b 一样的热键，这样按下 Ctrl-Alt-A 和按下 Ctrl-Alt-B 会执行相同动作，从而达到“触发” ^!b 热键的效果。如下例所示:

```
a:::b
^!a:::
^!b:::
```

```
ToolTip You pressed %A_ThisHotkey%.
```

```
return
```

如果在自动运行区(auto-execute section, 就是从脚本的第一行到出现第一个 Return, Exit, hotkey/hotstring label 之前的区域)使用了 [SendMode](#)(See 20.13), 所有的映射将会受其影响。另外, 由于映射使用的是 [Send {Blind}](#)(See 20.12) 模式, 而 [SendPlay mode](#)(See 20.13) 不完全支持 {Blind} 模式, 一些映射(尤其是包含 Control, Shift, Alt, Win 键的映射)在 SendPlay mode 下会出现一些不正常的现象。所以当脚本中有映射时, 应避免在自动运行区使用 SendPlay 模式; 然后在脚本其他需要的地方使用 [SendPlay](#)(See 20.12) 命令来代替 Send 命令。或者, 另一种解决这个问题的方法是, 将你的脚本用相应的热键代替(下面有详细的介绍), 这些热键使用 SendEvent 代替 Send 命令。

当脚本加载时, 其中的每一个映射会被翻译成一对 [热键\(hotkeys\)](#)(See 4.)。例如, 映射 `a::b` 会被翻译成如下两个热键:

```
*a::
```

```
SetKeyDelay -1 ; 如果目标键是鼠标键, 则使用 SetMouseDelay 命令。
```

```
Send {Blind}(See 20.12){b DownTemp} ; DownTemp 和 Down 相似, 只是如果使用 DownTemp, 脚本运行到下一个 Send 命令时会不会认为 'b' 键仍处于按下的状态。
```

```
return
```

```
*a up::
```

```
SetKeyDelay -1 ; 代码中两处 SetKeyDelay 命令都没有指定按键时长(press-duration, SetKeyDelay 命令的第二个参数), 其原因请看下面的解释。
```

```
Send {Blind}{b Up}
```

```
return
```

在下面的情况下, 将映射翻译成热键时会作一些调整:

1. 如果原始键(source key)是 LCtrl 键, 目标键(destination key)是 Alt 键, `Send {Blind}{LAlt DownTemp}` 会被替换成 `Send {Blind}{LCtrl Up}{LAlt DownTemp}`。当原始键是 RCtrl 时也会作类似的调整。
2. 如果是把键盘键映射成鼠标键(如 `RCtrl::RButton`), 则会用 SetMouseDelay 命令代替 SetKeyDelay 命令。同时, 上面的第一个热键也会被替换成下面的热键, 这样就可以防止由于键盘键按下后会重复发送键击消息而产生鼠标重复点击的消息:

```
3.*RCtrl::
```

```
4. SetMouseDelay -1
```

```
5.if not GetKeyState("RButton") ; 换言之, 鼠标右键尚未按下。
```

```
6. Send {Blind}{RButton DownTemp}
```

```
return
```

注意 `SetKeyDelay` 命令的第二个参数 **按键时长(press duration)**(See 20.14) 在上述的热键中被略去了。这是因为按键时长不适用于键按下和键弹起这样的事件(如 `{b down}` 和 `{b up}`)。但是按键时长却适用于涉及到 `Shift/Ctrl/Alt/Win` 键的事件，例如在 `a::B` 和 `a::^b` 中。因此，脚本中在 **自动运行区(auto-execute section)**(See 8.) 发生作用的按键时长参数会对所有这样的映射产生影响。

虽然不能将一对键直接映射成单个键(例如，`a & c::b` 这样的映射是无效的)，但只需修改上述的代码中两个热键(分别是键按下和键弹起时的热键)就可以实现：将 `*a::` 替换成 `a & c::`，并将 `*a up::` 替换成 `a & c up::` 即可。

由于映射最终会被翻译成热键，**Suspend**(See 17.23) 命令也会对映射产生影响。同样，也可以用 **Hotkey**(See 20.1.10) 命令来禁用或修改一个映射。例如，下面两行代码可以禁用映射 `a::b`。

```
Hotkey, *a, off
```

```
Hotkey, *a up, off
```

与 `Alt-tab` 有关的问题：如果你将一个键盘键或鼠标键映射成 `Alt` 键，这个键很有可能实现不了 `alt-tab` 的功能。在多数情况下，添加 `*Tab::Send {Blind}{Tab}` 这个热键解决这个问题——但是注意这很可能会影响原始 `Alt` 键实现 `alt-tab` 的功能。因此，只有当你仅用映射键或 **alt-tab hotkeys**(See 4.) 来实现 `alt-tab` 的功能时才用这个方法。

除了 **Key List**(See 7.) 页面所列出来的鼠标键和键盘键外，原始键(**source key**)也可以是一个虚拟键(**virtual key** 或 `VKnn`)或扫描码(`scan code` 或 `SCnnn`)，在 **special keys**(See 7.) 页面中有对虚拟键和扫描码的介绍。同样，目标按键也可以使用虚拟键或扫描码，不过它还可以同时使用虚拟键和扫描码(在虚拟按键后面再指定一个扫描码)。例如，在大多数键盘布局下 `sc01e::vk42sc030` 与 `a::b` 等效。

如果想禁用一个键，而不是映射这个键，将其定义成直接 **returns**(See 17.19) 的热键。例如，热键 `F1::return` 可以禁用 `F1` 键。

以下的键不支持映射：

- 鼠标滚轮。(虽然写着不支持，不过确实支持啊)
- `Pause` 键和 `Break` 键不能作为目标键(**destination keys**)。(这是因为它们在代码中会被当做命令执行)。
- 花括号 `{}` 不能作为目标键。但是可以用 **虚拟键/扫描码的方法(VK/SC method)**(See 20.12) 来实现。如 `x::+sc01A` 和 `y::+sc01B`
- 百分号 `%` 不能作为目标键。同样可以用 **虚拟键/扫描码的方法(VK/SC method)**(See 20.12) 来实现。
- 回车符 "Return" 不能作为目标键。用 "Enter" 代替。

通过映射(或者热键)，键盘可以用来移动鼠标的光标，如 **键盘-鼠标脚本(Keyboard-To-Mouse script)**(See 30.32) 所示。由于这个脚本能提供鼠标平滑移动、加速和其他的功能，如果你需要用键盘来实现鼠标的功能，推荐你使用这个脚本。下面是一个简化的键盘-鼠标脚本实例：

```
*#up::MouseMove, 0, -10, 0, R ; Win+UpArrow hotkey => 光标上移
```

```
*#Down::MouseMove, 0, 10, 0, R ; Win+DownArrow => 光标下移
```

```
*#Left::MouseMove, -10, 0, 0, R ; Win+LeftArrow => 光标左移
*#Right::MouseMove, 10, 0, 0, R ; Win+RightArrow => 光标右移

*<#RCtrl:: ; LeftWin + RightControl => 鼠标左击。
SendEvent {Blind}{LButton down}
KeyWait RCtrl ; 防止键盘自动重复导致鼠标重复点击。
SendEvent {Blind}{LButton up}
return

*<#AppsKey:: ; LeftWin + AppsKey => 鼠标右击。
SendEvent {Blind}{RButton down}
KeyWait AppsKey ; 防止键盘自动重复导致鼠标重复点击。
SendEvent {Blind}{RButton up}
return
```

优点:

- 注册表映射在大多数情况下比 [AutoHotkey](#) 的映射 更加直接和有效。如，注册表映射能在更多的游戏中起作用，而且不会存在 [alt-tab 问题](#)；另外也能够触发 [AutoHotkey](#) 中的钩子热键([hook hotkey](#))。(而 [AutoHotkey](#) 中的映射需要使用 [间接的方法\(workaround\)](#) 才能“触发”勾住的热键)。
- 如果你是手动地修改注册表项(下面会详细描述)，那么不需要使用任何外部软件就可以实现映射。即使你使用 [KeyTweak](#) 来修改注册表项，修改完后也不需要 [KeyTweak](#) 运行就可以让映射生效(而 [AutoHotkey](#) 则需要一直运行才能使映射生效)。

缺点:

- 缺少灵活性：每一次修改映射，需要重启计算机后才能让修改生效。
- 不能让映射仅对特定的用户、程序有效。
- 不能发送包含 [Shift](#), [Control](#), [Alt](#), [AltGr](#) 键的组合键击消息。也不能将一个小写字母映射成大写字母。
- 在 Windows 95/98/Me 下不被支持。而 [AutoHotkey](#) 能实现 [部分 Win9x 映射\(limited Win9x remapping\)](#)。

- 仅支持对键盘的映射。而 AutoHotkey 还支持 [鼠标映射](#) 和 [部分操作杆映射\(limited joystick remapping\)](#)(See 30.15)。

如何修改注册表: 至少有两种方法:

- 通过软件, 如 [KeyTweak](#) (免费软件)来映射键盘。它会自动修改注册表。
- 手动创建 .reg 文件(纯文本文件)并将它载入注册表。详见
www.autohotkey.com/forum/post-56216.html#56216

方法是直接使用热键。推荐在编写热键时使用 [Send\(See 20.12\)](#) 和 [KeyWait\(See 20.10\)](#) 这两个命令。例如, 下面的热键可以将 'A' 键映射成左方向键:

```
a::  
    Send {Left down} ; 按下左方向键。  
    KeyWait a ; 等待用户释放 'a' 键。  
    Send {Left up} ; 释放左方向键。  
return
```

[List of keys and mouse buttons\(See 7.\)](#)

[GetKeyState\(See 20.7\)](#)

[Remapping a joystick\(See 30.15\)](#)

7. 按键列表 (键盘、鼠标与操纵杆)

LButton - 鼠标左键

RButton - 鼠标右键

MButton - 鼠标中键或滚轮

WheelDown - 这相当于将鼠标滚轮向后旋转(朝向你自己)

WheelUp - 上面这种的反向。

WheelLeft 和 WheelRight [v1.0.48+] - 这两个要求鼠标有左右滚动的能力, 而且它们在 Windows Vista 以前的操作系统中没有效果。

(参见 [mouse wheel hotkeys\(See 4.\)](#) 来检测鼠标滚轮被滚动了多少。)

仅支持 Windows 2000/XP 或之后版本:

XButton1 - 仅出现在某些鼠标上的一个按键
 XButton2 - 同上

注意: 字母和数字按键的名称和单个字母或数字是一样的。例如: **b** 表示 "b" 键, **5** 表示 "5" 键。

Space - 空格键

Tab

Enter (或 Return)

Escape (或 Esc)

Backspace (或 BS)

Delete (或 Del)

Insert (或 Ins)

Home

End

PgUp

PgDn

Up

Down

Left

Right

ScrollLock

CapsLock

NumLock

Numlock 启用 **Numlock 关闭**

Numpad0 NumpadIns

Numpad1 NumpadEnd

Numpad2 NumpadDown

Numpad3 NumpadPgDn

Numpad4 NumpadLeft

Numpad5 NumpadClear

Numpad6 NumpadRight

Numpad7 NumpadHome

Numpad8 NumpadUp

Numpad9 NumpadPgUp

NumpadDot (.) NumpadDel

NumpadDiv (/) NumpadDiv (/)

NumpadMult (*) NumpadMult (*)

NumpadAdd (+) NumpadAdd (+)
 NumpadSub (-) NumpadSub (-)
 NumpadEnter NumpadEnter

F1 到 F24 - 在大多数键盘顶端的 12 个或更多的功能键。

AppsKey - 这是个调用右键上下文菜单的按键。 (译注：一般在右 Win 键和右 Ctrl 键中间)

LWin - 左边的 windows 标志 键

RWin - 右边的 windows 标志 键。注意：和 Control/Alt/Shift 不同，这里没有一般的/中性的 "Win" 键因为操作系统不支持。

Control (或 Ctrl)

Alt

Shift

注意：Shift::、Alt:: 和 Control:: 热键在按键松开时激发除非它们有波浪符前缀例如 ~Alt::。相比之下，一个明确了左或右的热键例如 LAlt:: 在按下时激发。

注意：多半，下面的 6 个键不被 Windows 95/98/Me 支持。使用上面的代替：

LControl (或 LCtrl) - 左边的 control 键

RControl (或 RCtrl) - 右边的 control 键

LShift - 左边的 shift 键

RShift - 右边的 shift 键

LAlt - 左边的 Alt 键

RAlt - 注意：如果你的键盘布局用 AltGr 代替了 RAlt，你也许可以通过 <^>! 让它像[这里\(See 4.\)](#)描述的那样作为一个热键前缀来使用。此外，"LControl & RAlt::" 将让 AltGr 它自身成为一个热键。

PrintScreen

CtrlBreak

Pause

Break -- 由于此键和 Pause 是一起的，在热键中不能使用 ^Pause 或 ^Break，而用 ^CtrlBreak 来代替。

Help - 此键也许在大多数键盘上不存在。它通常和 F1 不一样。

Sleep - 注意 sleep 键在一些键盘上可能不是休眠功能。 (译注：电源选项>高级 标签中可以调节。)

下面的仅存在那些有额外按键或按钮的多媒体或 Internet 键盘上：

Browser_Back

Browser_Forward

Browser_Refresh

Browser_Stop

Browser_Search

Browser_Favorites

Browser_Home

Volume_Mute

Volume_Down

Volume_Up

Media_Next
Media_Prev
Media_Stop
Media_Play_Pause
Launch_Mail
Launch_Media
Launch_App1
Launch_App2

SCnnn (这里 nnn 是一个按键的扫描代码) - 识别上面没有提到的特殊的按键。详见[特殊按键](#)。

VKn (这里 nn 是一个按键的十六进制虚拟按键代码) - 这种极少用的方法也阻止了某些类型的热键([See 4.](#))去依赖[键盘钩子](#)([See 20.2](#))。例如，后面的热键不使用键盘钩子，但作为一个副作用，通过按 **Home** 或 **NumpadHome** 都能触发它: ^VK24::MsgBox 你在按住 **Control** 的同时按下了 **Home** 或 **NumpadHome** 键。详见[特殊按键](#)。

Joy1 到 Joy32: 操纵杆的按钮。要帮忙确定你的操纵杆的按钮编号，请用这个[测试脚本](#)([See 30.26](#))。注意[热键前缀符号](#)([See 4.](#))例如 ^ (control) 和 + (shift) 不被支持 (虽然 [GetKeyState](#)([See 20.7](#)) 能用来代替)。同样注意如果激活的窗口被设计成探测操纵杆按钮的按下，操纵杆的按钮按下将总会“传递”给它。

虽然下面的操纵杆控制器名称不能被用作热键，但它们能和 [GetKeyState](#)([See 20.7](#))一起使用:

JoyX, JoyY 和 JoyZ: 操纵杆的 X (水平的)、Y (垂直的) 和 Z (高度/深度) 轴。

JoyR: 方向舵或操纵杆的第 4 个轴。

JoyU 和 JoyV: 操纵杆的第 5 个和第 6 个轴。

JoyPOV: point-of-view (hat) 控制器。

JoyName: 操纵杆的名称或它的驱动。

JoyButtons: 操纵杆支持的按钮总数(不会总是精确的)。

JoyAxes: 操纵杆支持的轴的总数。

JoyInfo: 提供一个包含零或下面多个字母组成的字串来显示操纵杆的功能: **Z** (有 Z 轴), **R** (有 R 轴), **U** (有 U 轴), **V** (有 V 轴), **P** (有 POV 控制器), **D** (POV 控制器有少数不连续的/不同的设置), **C** (POV 控制器是连续的/精细的)。字串示例: ZRUVPD

多个操纵杆: 如果计算机有多个操纵杆并且你想在第一个旁边使用另一个，那么在控制器名称前面包含操纵杆编号。例如，**2joy1** 是第二个操纵杆的第一个按钮。

注意: 如果你获得的脚本在识别你的操纵杆时有问题，有一个人报告说需要指定一个操纵杆的编号为除 1 以外的数字，即使只有单独一个操纵杆存在。不清楚这种情况如何发生的或者它是否正常，但将操纵杆编号在[操纵杆测试脚本](#)([See 30.26](#))中试验能帮助确定这是否适用于你的系统。

也可以看:

[操纵杆重映射](#)([See 30.15](#)): 用一个操纵杆发送键击和鼠标点击的方法。

[操纵杆到鼠标脚本](#)([See 30.25](#)): 将操纵杆作为鼠标使用。

响应手持遥控通过 [WinLIRC 客户端脚本](#)(See 30.36)发送的信号。

如果你的键盘或鼠标有按键没有在上面被列出来,你也许通过采用下面的步骤仍有可能使它成为一个热键(需要 Windows XP/2000/NT 或之后版本):

1. 确保至少有一个使用[键盘钩子](#)(See 20.2)的脚本正在运行。你只要通过打开一个脚本的主窗口并从菜单栏选择 "View->Key history" 就能知道它是否使用了键盘钩子。
2. 双击那个脚本的托盘图标来打开它的主窗口。
3. 按下你键盘上的某个“神秘按键”。
4. 选择菜单项 "View->Key history"
5. 向下滚动到页面的底部。在底部附近的某处就有你的按键的按下和弹起事件记录。注意: 某些按键不生成事件记录, 因此这里将看不到记录。如果是这种情况, 你不能直接让那个特殊按键成为一个热键, 因为你的键盘驱动或硬件在一个 AutoHotkey 访问不到的很低的层面处理它。要得到可能的解决方案, 再往下看。
6. 如果你的按键可被探测到, 记下在列表第二列的 3 位十六进制数值 (例如 **159**)。
7. 要定义此键为一个热键, 参照此例:

8. SC159:: ;用你那按键的数值替换 159。

9. MsgBox, %A_Hotkey% 被按下。

return

反向: 要让其他某些键重映射为一个“神秘按键”, 参照此例:

;用上面发现的数值替换 159。用按键的虚拟键值替换 FF (如果需要的话), 它能在按键历史页面的第一列找到。

#c::Send {vkFFsc159}(See 20.12)

替换的解决方案: 如果你的按键或鼠标按键在按键历史页面没有被探测到, 下面的某种方法也许有帮助:

1. 重新配置你的鼠标或键盘附带的软件(大多数时候能在控制面板或开始菜单进行访问) 来使“神秘按键”发送其他一些键击。然后这样的一个键击能在脚本里被定义为一个热键。例如, 如果你设置一个神秘按键来发送 **Control+F1**, 之后你能通过在脚本中用 **^F1::** 直接让它成为一个热键。
2. 试试 [DII Call: Support for Human Interface Devices](#)。你也可以在 [forum](#) (论坛) 中搜索类似 **RawInput*** 的关键字。
3. 下面是一个最后的手段并且通常应该仅在绝望时尝试。这是因为成功的机会很小并且它可能导致不想要的副作用而且难以撤销:
禁用或移除你的键盘或鼠标附带的额外的软件或者将它的驱动改变为一个更加标准的例如建立在操作系统上。这里假设的是有那么一种驱动给你特殊的键盘或鼠标, 而你能不用它自定义的驱动和软件提供的特性而继续使用下去。

翻译: 天堂之门 menk33@163.com 2008 年 12 月 31 日

8. 脚本

- [介绍](#)
- [脚本顶部\(自动执行部分\)](#): 当脚本启动时，这部分会自动地执行。
- [转义顺序](#): 何时要使用 `%` 和 `，来表示原义的百分号或逗号。
- [脚本中的注解](#): 使用分号和 `/*...*/` 符号给脚本添加备注。
- [把一个过长的行分割成一系列短行](#): 这样可以改善脚本的可读性和可维护性。
- [把脚本转换成 EXE 文件\(ahk2exe\)](#): 把一个 .ahk 脚本转换成一个能在任何 PC 上运行的 .exe 文件。
- [向脚本传递命令行参数](#): %1%, %2% 等等包含了输入参数的变量。
- [调试脚本](#): 如何在不能正常工作的脚本中找到的问题。
- [AutoHotkey.exe 的可移植性](#): 拥有一个 AutoHotkey.exe 的备份即可执行任何 .ahk 文件。
- [安装程序的选项](#): 如何进行无人值守/静默的安装或卸载。

每个脚本都是一个纯文本文件，其中包含了要被程序(AutoHotkey.exe)执行的行。一个脚本也可以包含[热键](#)(See 4.)和[热字符串](#)(See 5.)，或者完全由它们构成。不过，在没有热键和热字符串的情况下，脚本会在启动的那刻起，从上到下顺序地执行其中的命令。

程序会将脚本逐行的读取到内存中，每行最多可以包含 16,383 个字符。在读取过程中，脚本将被[优化](#)(See 30.13)并确认。任何语法错误都将被列出，它们必须被纠正后脚本才能运行。

在脚本被读取完毕后，它会从第一行开始执行，直至遇到 [Return](#)(See 17.19), [Exit](#)(See 17.6), [热键/热字符串标签](#)(See 4.)或者脚本的底部(无论最先遇到哪个)。脚本的这个顶部被称为[自动执行部分](#)。

一个非持续的(See 29.21)并且没有[热键](#)(See 4.)、[热字符串](#)(See 5.)、[OnMessage](#)(See 18.11)和[GUI](#)(See 19.3)的脚本，将会在自动执行部分结束后终止。否则，它会以空闲状态继续运行，从而对例如热键、热字符串、[GUI 事件](#)(See 19.3)、[自定义菜单项](#)(See 22.13)和[定时器](#)(See 17.21)这些事件作出反应。

如果在自动执行部分设置了下列属性，那么由[热键](#)(See 4.)、[热字符串](#)(See 5.)、[菜单项](#)(See 22.13)、[GUI 事件](#)(See 19.3)或[定时器](#)(See 17.21)启动的每个[线程](#)(See 30.19)都将以它们的默认值重新开始执行。如果未设置，则应用标准的默认值(如下列每个页面中说明的那样): [DetectHiddenWindows](#)(See 28.5), [DetectHiddenText](#)(See 28.4), [SetTitleMatchMode](#)(See 28.8), [SetBatchLines](#)(See 17.20), [SendMode](#)(See 20.13), [SetKeyDelay](#)(See 20.14), [SetMouseDelay](#)(See 23.8), [SetWinDelay](#)(See 28.9), [SetControlDelay](#)(See 28.1.13), [SetDefaultMouseSpeed](#)(See 23.7), [CoordMode](#)(See 22.6), [SetStoreCapslockMode](#)(See 20.16), [AutoTrim](#)(See 22.3), [SetFormat](#)(See 21.10), [StringCaseSense](#)(See 27.13), [Thread](#)(See 22.22) 以及 [Critical](#)(See 22.7)。

如果自动执行部分需要较长时间才能完成(或者无法完成), 那么上面这些设置的默认值将会在 100 毫秒之后生效。当自动执行部分最终完成(如果能的话), 那些在自动执行部分结尾处已生效的默认值才被再次更新。因此, 通常最好在包含[热键\(See 4.\)](#)、[热字符串\(See 5.\)](#)、[定时器\(See 17.21\)](#)或者[自定义菜单项\(See 22.13\)](#)的脚本的顶部对默认值做出想要的更改。还要注意, 每个[线程\(See 30.19\)](#)会保持它自己的上述设置的集合。对那些设置所做的更改不会影响到其他[线程\(See 30.19\)](#)。

AutoHotkey 的默认转义字符([See 29.5](#))是重音符/反引号(`), 它位于大多数英文键盘的左上角。使用这个字符而不是反斜线, 可以避免在文件路径中对双反斜线的需要问题。

由于在 **AutoHotkey** 语言中逗号和百分号有着特殊的意义, 因此可以用 ` 来指定一个原义的逗号, 用 `% 来指定一个原义的百分号。[MsgBox\(See 19.11\)](#) 是一个例外, 它不需要将逗号转义。另一个例外是, 任何命令的最后一个参数中的逗号: 它们不需要被转义。有关转义顺序的完整列表, 请看 [#EscapeChar\(See 29.5\)](#)。

某些特殊字符也需要通过转义顺序来生成。最常见的有 `t ([tab](#)), `n (换行) 和 `r (回车)。

提示: 任何命令的第一个逗号都可以省略(当第一个参数为空或者命令单独处于[连续部分](#)的首行这两种情况除外)。例如:

MsgBox 这样可以。

MsgBox, 这样也可以 (它有一个明显的逗号)。

在行首使用一个分号可以注解脚本。例如:

; 这整行是一个注解。

注解也可以添加到一个命令的末尾, 这种情况下分号左侧必须至少有一个空格或 [tab](#)。例如:

Run Notepad ;这是一个和命令在同一行的注解。

另外, /* 和 */ 符号可以被用来注解掉整个部分, 但是只有在符号出现在行首时才行, 如下所示:

/*

MsgBox, 这一行被注解掉了(禁用)。

MsgBox, 这行也是。

*/

由于在脚本启动时将忽略注解, 所以他们不会影响性能或者内存利用率。

通过 [#CommenFlag\(See 29.3\)](#) 可以把默认的注解符号(分号)更改为其它字符或字符串。

过长的行可以被分割成一系列较短的行来改善可读性和可维护性。这样做不会降低脚本的执行速度，因为在脚本启动的时候这些短行会在内存中被合并。

方法 #1: 以 "and"、"or"、"||"、"&&"、逗号或句号(See 9.)开头的行会自动合并到它的上一行(在 1.0.46 及之后版本，除了 ++ 和 -- 以外的所有其他表达式运算符(See 9.)都会如此)。下例中，由于第二行以一个逗号开头，所以它被追加到第一行：

```
FileAppend, 这是要追加的文本`n ;这里允许使用注解。
, %A_ProgramFiles%\SomeApplication\LogFile.txt ;注解。
```

类似地，以下几行将被合并为一行，因为最后两行是以 "and" 和 "or" 开头的：

```
if (Color = "Red" or Color = "Green" or Color = "Blue" ;注解。
    or Color = "Black" or Color = "Gray" or Color = "White") ;注解。
    and ProductIsAvailableInColor(Product, Color) ;注解。
```

三元运算符(See 9.)也是一个很好的候选：

```
ProductIsAvailable := (Color = "Red")
? false ;我们没有任何红色的产品，因此不用麻烦调用函数了。
: ProductIsAvailableInColor(Product, Color)
```

尽管用在上面例子中的缩进不是必须的，但它可以指示哪行从属于上一行，这样可以改善代码的清晰度。此外，没有必要将额外的空格包括在以单词 "AND" 和 "OR" 开头的行里；程序会自动处理这些。最后，可以在上面例子中的任何行与行之间或者行尾添加空行或注解。

方法 #2: 这个方法可以用来合并大量的行或者是方法 #1 不适用的行。尽管本方法对自动替换热字符串(See 5.)特别有用，但它同样可以用于任何命令或表达式(See 9.)。例如：

```
;例子 #1:
Var =
(
第一行文本。
```

第二行文本。默认情况下，将在两行中插入一个换行符(`n)。

```
)
```

```
;例子 #2:
FileAppend, ;这种情况下需要逗号。
(
一行文本。
```

默认情况下，上一行和本行之间的硬回车(**Enter**)将作为换行符(`n)写到文件中。

默认情况下，本行左侧的 **tab** 也将写到文件中(空格也会一样)。

默认情况下，例如 **%Var%** 这样的变量引用将被解析为变量的内容。

```
), C:\My File.txt
```

在上面的例子中，数行内容在头尾被一对圆括号围起来。这被称作**连续部分**。注意，最下面一行的右圆括号后包含了 **FileAppend**(See 16.4) 的最后一个参数。这个习惯是可选的；此时这样做是以便将逗号视为一个参数分隔符而非一个原义的逗号。

通过在左括号的右侧包含一个或多个下列选项，可以覆盖连续部分的默认特性。如果存在多个选项，用它们彼此用一个空格分隔开。比如：(**LTrim Join| %**

Join: 指定行与行应该如何连接。如果省略此选项，除最后一行外，每一行后面都会跟一个换行符(`n)。如果单独指定单词 **Join**，行与行之间不添加任何字符而直接连接起来。否则，单词 **Join** 后面应该最多紧跟 15 个字符。例如，**Join`\$** 将会在除最后一行外的每行末尾添加一个空格(`\$ 代表一个原义的空格，这是一个只能被 **Join** 识别的特殊的转义顺序)。再比如 **Join`r`n**，将在行与行之间插入 **CR+LF**(回车+换行)。类似地，**Join|** 将在行与行之间插入一个竖线。要让连续部分的最后一行也以一个 **join** 字符串结尾，只要在连续部分右括号的上一行包含一个空行即可。

LTrim: 省略每行头部的空格和 **tab**。主要被用来允许连续部分使用缩进。另外，通过在一行上单独指定 **#LTrim** 可以对多个连续部分开启此选项。**#LTrim** 是位置相关的：它会影响它下面所有的连续部分。可以通过 **#LTrim Off** 来关闭此选项。

RTrim0 (RTrim 后跟一个零): 将忽略每行末尾的空格和 **tab** 的设置关闭。

Comments (或 Comment 或 Com 或 C) [1.0.45.03 及之后版本]: 在连续部分内允许**分号注解**(但 **/*..*/** 无效)。这种注解(连同它们左侧的任何空格和 **tab**)将从合并结果内完全忽略，而不会被当作原义的文本处理。每个注解都可以出现在一行的右侧或者单独另起一行。

% (百分号): 把百分号作为原义处理而非变量引用。这样就无需为每个百分号**转义**(See 29.5)而使其成为原义。在百分号已经是原义的地方就不需要此选项了，比如[自动替换热字符串](#)(See 5.)。

, (逗号): 将逗号作为分隔符处理，而非其原义。这个罕用的选项只有在命令的参数之间才用得到，因为在[函数调用](#)(See 10.)中，逗号的类型不重要。并且，此选项只转换那些真正分隔参数的逗号。换句话说，一旦到达命令的最后一个参数(或者根本就没有参数)，随后的逗号就将作为原义逗号来对待从而忽略此选项。

` (重音符): 把每个反引号都作为原义来对待，而不是**转义字符**(See 29.5)。这也防止了逗号和百分号被明确单独地转义。此外，它会防止任何明显被指定的转义顺序比如：`r 和 `t 的转换。

备注

除了**重音符`**选项被指定的时候外，在连续部分中是支持**转义顺序**(See 29.5)的比如 `n(换行)和 `t (tab)。

当未指定 **comment** 选项时，在连续部分内部是不支持分号和 **/*..*/** 注解的，因为它们被视为原义的文本。不过，在连续部分的顶部和底部行中可以包含注解，例如：

```
FileAppend, ;注解。
```

```
;注解。
( LTrim Join ;注解。
; 这不是一个注解；它是原义的。在上一行添加单词 Comments 就可以使这行成为注解。
), C:\File.txt ;注解。
```

由于上面的原因，在连续部分中分号不再需要被[转义](#)(See 29.5)。

连续部分无法创建总长度超过 16,383 个字符的行(如果尝试创建，程序会在脚本启动时警告你)。解决方案是把一系列内容串联到一个变量中。例如：

```
Var =
(
...
)
Var = %Var%`n ;通过另一个连续部分向变量添加更多文本。
(
...
)
FileAppend, %Var%, C:\My File.txt
```

因为一个右括号标志着一个连续部分的结束，要让某一行以原义的右括号开头，可以在它前面用重音符/反引号开头：`)。

一个连续部分后面可以紧跟某个包含了另一个连续部分左括号的行。这就使得上面提及的那些选项在创建单行内容的过程中可以多样化。

不支持通过 [#Include](#)(See 17.1) 的方式将一个个连续部分组合起来。

AutoHotkey 的程序中包括一个脚本编译器(受 Jonathan Bennett 的 AutoIt v3 源代码的恩惠)。不支持 AutoIt v2 的脚本，所以如果有必要的话，可以先将你的 .aut 文件[自动转换](#)(See 11.)成 .ahk 文件。

一旦脚本被编译，它就生成一个独立的可执行文件；就是说即便在没有安装 AutoHotkey 的机器上也可以运行(这样的 EXE 文件可以随意发布或出售)。编译过程中，下列所有文件将被压缩并加密：脚本、脚本 [include](#)(See 17.1) 的任何文件，以及任何通过 [FileInstall](#)(See 16.15) 命令合并的文件。

编译并不会改善脚本的性能。事实上已编译的脚本启动时会略慢，因为它必须先解密和解压缩到内存，这之后它就会像一个普通脚本那样被[优化](#)(See 30.13)。

可以用下列方法使用 Ahk2Exe：

1. **图形界面:** 在开始菜单中运行 "Convert .ahk to .exe"。在 1.0.46.10 及之后版本, 可以在图形界面的密码框指定 N/A 来避免被 [exe2ahk](#) 反编译成脚本。此方法仅适用于图形界面; 对于命令行方式, 可用 `/NoDecompile` 开关来实现。
2. **右键点击:** 在打开的资源管理器窗口内, 你可以右键点击任何 .ahk 文件, 选择 "Compile Script"(只有安装 AutoHotkey 时选择了 `script compiler` 选项才会有效)。稍后在同个目录下会产生一个和脚本同名的 EXE 文件。注意: 会用和方法 1 中最后一次使用的自定义图标和压缩等級相同的设置来生成 EXE 文件, 并且没有反编译密码。
3. **命令行:** 编译器可以使用下列参数以命令行方式运行:

```
Ahk2exe.exe /in MyScript.ahk [/out MyScript.exe][/icon MyIcon.ico][/pass password][/NoDecompile]
```

例如:

```
Ahk2exe.exe /in "MyScript.ahk" /icon "MyIcon.ico" /pass "CustomPassword" /NoDecompile
```

用法:

- 包含空格的参数应该置于双引号之间。
- 如果省略 "out" 的文件名, EXE 文件会和脚本同名。
- 在 1.0.46.10 及之后版本, 即便给出了正确的密码, `/NoDecompile` 开关(如果有的话)也会阻止 [exe2ahk](#) 反编译脚本。这样就能在一个长而复杂的密码之外再提供一层额外的保护。

注意:

- 如果你打算发布自己的 EXE 文件, 并且不想让任何人查看你的脚本源代码, 你可以使用带 `/NoDecompile` 开关的命令行方式进行编译并指定一个长而复杂的密码, 从而获得最大限度的保护。密码的最大长度为 64 个字符。
- [#NoTrayIcon\(See 22.1\)](#) 和 "[Menu, Tray, ShowMainWindow\(See 22.13\)](#)" 命令将会影响编译后脚本的行为。
- 下载 [Exe2Ahk](#) (此工具应该在命令提示符中运行)就可以反编译 EXE 文件, 从而提取原脚本。不过, 原来脚本中存在的注解(分号或 `/**/`)都将丢失。
- 将你的 AutoHotkey\Compiler 文件夹中的 AutoHotkeySC.bin 文件替换为[这个更小的版本](#)(覆盖已存在的同名文件), 就可以使编译的脚本减少大概 20 KB 的大小。以这种方式产生的任何已编译脚本都将依赖 MSVCRT.dll。尽管这个 DLL 在 Windows 2000/XP 或之后的系统中总是存在, 但较早的操作系统却不一定有。
- 使用 Resource Hacker(免费软件)这样的工具来编辑 "AutoHotkeySC.bin" 文件, 就可以将自定义的版本信息(在资源管理器的文件属性对话框中看到的那些)添加到你编译的脚本中。该文件被包含在 AutoHotkey 安装目录的 "Compiler" 子文件夹中。注意: Resource Hacker 会破坏已编译的脚本, 这就是为什么应该只编辑 AutoHotkeySC.bin 文件的原因。
- 上面的方法也可以用来改变所有已编译脚本中已存在的图标或者添加新的图标。
- 如果脚本是用已编译的形式运行, 那么内置变量 `A_IsCompiled` 包含 1, 否则为空。
- 如果你不希望你编译的脚本被压缩, 只要删除或者重命名 AutoHotkey 的 "Compiler" 文件夹中的 "upx.exe" 文件即可。

- 当传递参数给 `Ahk2Exe` 时，一条关于编译过程成功与否的消息将会写入到标准输出。尽管消息不会显示在命令提示符中，但可以使用诸如重定向输出到一个文件的方式来“获取”。[1.0.43 及之后版本]

脚本支持命令行参数。格式为：

```
AutoHotkey.exe [Switches] [Script Filename] [Script Parameters]
```

对于编译过的脚本，格式为：

```
CompiledScript.exe [Switches] [Script Parameters]
```

Switches: 零个或多个下列开关：

`/f` 或 `/force` -- 无条件运行，跳过任何警告对话框。
`/r` 或 `/restart` -- 表明脚本将被重新加载(在内部，[Reload](#)(See 20.1.13) 命令也使用这个开关)。
`/ErrorStdOut` -- 把妨碍脚本启动的语法错误发送到标准输出而不是显示一个对话框。详见
[#ErrorStdOut](#)(See 29.4)。

Script Filename: 如果没有 `Script Parameters` 的话可以省略这个参数。如果省略了，将会运行(或者提示你创建)[我的文档](#)(See 9.)文件夹中的 `AutoHotkey.ahk` 文件。唯一的例外是，在当前的工作目录存在 `AutoHotkey.ini` 文件，这样的话就会运行那个文件。

Script Parameters: 你想要传递给脚本的字符串，彼此间用一个空格分隔。任何包含有空格的参数应该置于一对引号之间。要使用原义的引号，可以通过在它前面加一个反斜线(\")来传递。因此，任何引号内的参数末尾是反斜线的(例如："C:\My Documents\"将被作为一个原义的引号处理(就是说，脚本将会收到 `C:\My Documents"` 这样的字符串))。要移除这样的引号，请用 [StringReplace, 1, 1, "", All](#)(See 27.20)

脚本把传入的参数视为 `%1%`, `%2%` 等等这样的[变量](#)(See 9.)。另外，`%0%` 包含了传入参数的数量(如果没有则为 0)。在下面的例子中，如果传入的参数数量不够，脚本将会退出：

```
if 0 < 3 ;非表达式 if 语句(See 17.10)的左侧总是变量名。
{
    MsgBox 此脚本需要至少三个传入参数，但只接收到 %0% 个。
    ExitApp
}
```

如果传送给脚本的参数数量不定(可能是由于用户拖放了一组文件到脚本上)，下面的例子可以用来将它们逐个提取：

```
Loop, %0% ;对每个参数执行一次:
{
    param := %A_Index% ;提取 A_Index 包含的变量名称里的内容。
    MsgBox, 4,, 第 %A_Index% 个参数是 %param%。继续?
```

```
IfMsgBox, No
    break
}
```

如果参数是文件名，下面的例子可以用来把他们转换为大小写正确的长文件名(像在文件系统中存储的那样)，包括完全/绝对路径：

```
Loop %0% ;对每个参数(或者拖放到脚本上的文件)执行一次:
{
    GivenPath := %A_Index% ;提取 A_Index 包含的变量名称里的内容。
    Loop %GivenPath%, 1
        LongPath = %A_LoopFileLongPath%
    MsgBox 文件`n%GivenPath%`n 的大小写正确的长路径名是: `n%LongPath%
}
```

已知限制：如果 NTFS 文件系统已经关闭了的 8.3(短)文件名，那么拖放到 .ahk 脚本上的文件将无法正常工作。一个解决方案是[编译](#)脚本，然后向编译产生的 EXE 拖放文件。

[ListVars\(See 22.12\)](#) 和 [Pause\(See 17.18\)](#) 等命令可以帮助你调试一个脚本。例如，下面这两行，当临时插入到精心选择的位置时，脚本会创建一个“断点”：

[ListVars\(See 22.12\)](#)

Pause

当脚本运行到这两行的时候，将会显示你需要检视的所有变量的当前内容。当你准备恢复的时候，只要通过 File 或者托盘菜单反 pause 脚本即可。然后脚本将继续运行直到遇到下一个“断点”(如果有的话)。

通常最好把这些“断点”插入到激活的窗口和脚本没什么关系的位置，比如 WinActivate 命令的上一行。这样就允许你反 pause 脚本时，它能妥当地恢复操作。

以下几个命令对调试脚本也很有用：[ListLines\(See 22.11\)](#), [KeyHistory\(See 20.9\)](#) 和 [OutputDebug\(See 22.14\)](#)。

运行任何 .ahk 脚本就只需要 AutoHotkey.exe 这一个文件。唯一的例外是 Windows NT4，任何在这个系统上使用 [Process 命令\(See 24.4\)](#)的脚本都需要复制一份 psapi.dll(从 AutoHotkey 文件夹)。

要静默安装 AutoHotkey 到默认目录(这也是非静默模式所显示的同一个目录), 只要把参数 /S 传递给安装程序即可(/S 必须是大写的)。例如:

```
AutoHotkey104307_Install.exe /S
```

可以通过参数 /D 来指定默认路径以外的路径(没有参数 /S 的情况下, 这将改变通过安装程序显示的默认路径)。例如:

```
AutoHotkey104307_Install.exe /S /D=C:\Program Files\AutoHotkey
```

要静默地卸载 AutoHotkey, 只要把参数 /S 传递给反安装程序即可。例如:

```
"C:\Program Files\AutoHotkey\uninst.exe" /S
```

这个页面列出了一些有用的脚本。

翻译: wanggang999 修正: 天堂之门 menk33@163.com 2008 年 11 月 27 日

9. 变量和表达式

- 变量介绍
- 表达式
- 表达式中的运算符
- 内置变量
- 环境变量对普通变量
- 变量的容量和内存

变量类型: AutoHotkey 没有明确定义变量类型; 所有的变量都是以字符串的形式存储。不过, 当一个只包含数字(可以还有小数点)的变量进行数学运算或比较时, 它将被自动地转换为数值(译注: 整型数或浮点型数等)。反过来, 当数学运算的结果需要存储在一个变量里时, 它将被转换回字符串。

变量的范围和声明: 除了函数中的**局部变量**(See 10.), 其他所有变量都是全局的; 也就是说, 它们的内容可以在脚本的任何位置被读取或改变。此外, 变量是无需声明的; 只要在使用它们的时候它们就产生了(每个变量初始为空/空白)。

变量名: 变量名不区分大小写(例如, *CurrentDate* 等同于 *currentdate*)。变量名最长可以有 254 个字符并且可以由字母、数字和下列标点符号: # _ @ \$? [] 组成。

由于格式上的惯例, 一般来说最好仅使用字母、数字和下划线来命名你的变量(例如: *CursorPosition*、*Total_Items* 和 *entry_is_valid*)。这种命名风格使熟悉其他计算机语言的人能够更容易地理解你的脚本。

而且，如果你在 **AutoHotkey** 中使用的命名风格与你在其他语言中使用的风格一致，你会发现能更容易地重读自己的脚本。

虽然一个变量名可以完全由数字组成，但是通常仅给传入的命令行参数([See 8.](#))使用。这种数字名称不能在表达式中使用，因为它们会被视为数字而不是变量。

因为单词 **AND**、**OR**、**NOT** 在表达式中被当作[运算符](#)使用，所以通常它们不应该作为变量名使用。在表达式中使用这样的名称将阻碍正确的计算。

给变量赋值：要在在一个变量里储存字符串或数值，有两种方法：传统方法和表达式方法。传统方法使用[等号运算符\(=\)](#)([See 22.19](#))来指定未引用的原义字符串或者围在百分号里的变量。例如：

```
MyNumber = 123
MyString = This is a literal string.
CopyOfVar = %Var% ;和 = 运算符一起用时，需要使用百分号来获得变量的内容。
```

相比之下，表达式方法使用[冒号-等号运算符 \(:=\)](#)([See 21.12](#))来存储数字、引用的字符串和其他类型的表达式。下面的例子在功能上与上面的例子相同：

```
MyNumber := 123
MyString := "This is a literal string."
CopyOfVar := Var ;和上面的部分里与它相似的那个不同，百分号不和 := 运算符一起使用。
```

第二种方法由于其更清晰以及与其他语言几乎一致的[表达式语法](#)而被大多数人所喜爱。

从上面的例子来看，你可能已经猜到了有两种方法来清除一个变量的内容(就是说，使变量为空)：

```
MyVar =
MyVar := ""
```

上面这对空双引号只能和 **:=** 运算符一起使用，因为假如它与 **=** 运算符一起用，它将在变量里储存两个原义的引号字符。

获取变量的内容：类似于赋值的两种方法，获取变量的值也有两种方法：传统方法和表达式方法。传统方法要将每个变量名围在百分号里以取得它的内容。例如：

```
CopyOfVar = %Var%
MsgBox(See 19.11) 变量 Var 的值为 %Var%。
```

相比之下，表达式方法不用在变量名两边加上百分号，但要将原义的字符串括入双引号。所以，下列表达式方法是与上面的例子相等的：

```
CopyOfVar := Var
MsgBox % "变量 Var 的值为" . Var . "。
```

在上面 **MsgBox** 这行，一个百分号一个空格用来将参数由传统方法变为表达式方法(译注：后面的一个点两个空格表示连接，具体请看下面的表达式中的运算符那段)。因为所有命令默认使用传统方法(除了那些另外说明的)，所以这是必须的。不过，一些命令的特定参数已说明过接受表达式，在这种情况下，带头的百分号允许使用但不是必须的。例如，下面所有命令实际上是一样的，因为 **Sleep**([See 17.22](#)) 的首个参数可以是表达式：

```
Sleep MillisecondsToWait
Sleep %MillisecondsToWait%
Sleep % MillisecondsToWait
```

比较变量: 请读下面的[表达式这节](#)中关于各种比较的注意事项，特别是关于何时使用圆括号的部分。

表达式被用来在一系列变量、原义的字符串和/或原义的数字上执行一个或者多个操作。

表达式中的变量名不用围在百分号内 (除了数组和其他的[双重引用](#))。所以，为了与变量区别开来，原义的字符串必须用双引号围住。例如：

```
if (CurrentSetting > 100 or FoundColor <> "Blue")
    MsgBox 设置太高或者呈现了错误的颜色。
```

在上面的例子里，“Blue”出现在双引号内，因为它是一个原义的字符串。要在一个原义的字符串内包含一个**真的**引号字符，请指定两个连续的引号，如在这个例子中出现了两回：“She said, """An apple a day."""”

重要的: 一个包含有表达式的 if-语句与[传统的 if-语句](#)(See 17.10)例如 *If FoundColor <> Blue* 可以通过 “if” 后面是否有左括号来区分。虽然一般将整个表达式放在括号里，但也可以像这样写 *if(x > 0) and (y > 0)*。此外，如果 “if” 后面的第一项是一个[函数调用](#)(See 10.)或者一个像 “not” 或 “!” 这样的运算符，括号也可以完全地省略。

空字符串: 要在表达式中指定一个空字符串，可以使用一对空引号。例如，语句 *if(MyVar <> "")* 中如果 *MyVar* 不为空，那么将返回 *true*。不过，在[传统的-if](#)(See 17.10) 中，一对空引号将被当作原义的字符串。例如，语句 *if MyVar = ""* 只在 *MyVar* 包含一对真的引号时才为 *true*。因此，要检查在一个传统的-if 语句中变量是否为空，可以像这个例子一样让 = 或 <> 右侧为空：*if Var =*

作个相关提示，任何无效的表达式例如 *(x +* 3)* 都将产生一个空字符串。

存储表达式的结果: 要指定一个表达式的结果给变量，可以使用 [:= 运算符](#)(See 21.12)。例如：

```
NetPrice := Price * (1 - Discount/100)
```

因为上面的例子产生了一个浮点型数(由于有除法)，那么存在变量 *NetPrice* 中的小数位数就由 [SetFormat](#)(See 21.10) 决定。[SetFormat](#) 还可以改变表达式的结果的其他特性。相比之下，[AutoTrim](#)(See 22.3) 不会对表达式的结果产生影响。

布尔值: 当一个表达式需要计算出是 *true* 还是 *false* 时(例如一个 IF-语句)，表达式结果为空或为零将视为 *false*，而其他所有结果视为 *true*。例如，语句 *"if ItemCount"* 只在 *ItemCount* 为空或为零时才是 *false*。类似地，语句 *"if not ItemCount"* 将得出相反的结果。

运算符例如 NOT/AND/OR/>/=/< 将自动生成一个 *true* 或 *false* 的值：它们是 *true* 得出 1，而 *false* 得出 0。例如，在下面的表达式中，如果任何一个条件为 *true*，变量 *Done* 会指定为 1：

```
Done := A_Index > 5 or FoundIt
```

如上面提示的那样，一个变量可以简单地通过留空或指定为 0 使其一直是 *false* 值。利用这个，速写的语句 *"if Done"* 可以用来检查变量 *Done* 是 *true* 还是 *false*。

单词 `true` 和 `false` 是包含 `1` 和 `0` 的内置变量。它们可以被用来使脚本像下面的例子那样更具可读性：

```
CaseSensitive := false
ContinueSearch := true
```

整型数和浮点型数：在一个表达式里，原义的数字如果含有一个小数点将被视为浮点型数；反之，就是整数。对于绝大多数运算符 -- 例如加和乘 -- 只要输入的任何一个是浮点型数，结果将也是一个浮点型数。

不论是在表达式里还是在表达式外，整型数都可以被写为十六进制数形式或十进制数形式。十六进制数以前缀 `0x` 开头。比如，`Sleep 0xFF` 等同于 `Sleep 255`。在 v1.0.46.11 及之后版本中，能识别以科学计数法表示的浮点型数；不过只有在它们包含小数点时才行(例如 `1.0e4` 和 `-2.1E-4`)。

强制写入表达式：通过在表达式的前面加上一个百分号以及空格或 `tab`，能让表达式用在一个不直接支持它的参数里(除了像 [StringLen](#)(See 27.17) 的那些 `OutputVar` 或 `InputVar` 参数外)。这种技巧常被用来访问[数组](#)(See 30.5)。例如：

```
FileAppend(See 16.4), % MyArray%i%, My File.txt
MsgBox(See 19.11) % "变量 MyVar 包含" MyVar "."
Loop(See 17.12) % Iterations + 1
WinSet(See 28.29), Transparent, % X + 100
Control(See 28.29), Choose, % CurrentSelection - 1
```

除非在下面另有规定外，优先级相同的运算符例如乘(*)和除(/)按从左到右的顺序计算。相比之下，优先级低的运算符在优先级高的之后执行，比如加(+)在乘(*)后执行。例如，`3 + 2 * 2` 等价于 `3 + (2 * 2)`。
括号在这个例子里可以用来超越优先级：`(3 + 2) * 2`

表达式运算符(以优先级降序排列)

%Var%	在表达式中如果一个变量被百分号围住(例如 <code>%Var%</code>)，不管此变量包含什么都将被当作另一个变量的名称或者部分名称(如果不在此变量， <code>%Var%</code> 就解析为一个空字符串)。常用来像下例这样引用 数组 (See 30.5)元素： <code>Var := MyArray%A_Index% + 100</code> 被引用的元素不应该是环境变量、剪切板或任何保留的/只读的变量。如果是，它将被当做一个空字符串。
++ --	前置和后置递增/递减。 将变量增加或减少 1(不过在 v1.0.46 之前的版本里，只有将它们放在一行才能用；不能有其他的运算符出现)。此运算符可以放在变量名称的前面或后面。如果出现在变量的前面，此运算马上执行并且它的结果可以给之后的运算使用。例如 <code>Var := ++X</code> 马上递增 <code>X</code> 然后将其值传给 <code>Var</code> 。相反地，如果运算符出现在变量的后面，变量被之后的运算使用过后才执行此运算。例如 <code>Var := X++</code> 仅在把当前 <code>X</code> 的值给 <code>Var</code> 之后才递增 <code>X</code> 。
**	幂。 底和指数都可以包含小数点。如果指数是负数，即使底和指数都为整数，结果也将格式为浮点数。因为 <code>**</code> 的优先级比负号更高， <code>-2**2</code> 计算成 <code>-(2**2)</code> 求得 <code>-4</code> 。因此，要让负数能够做幂运算的底，可以将这个负数用小括号括住，比如 <code>(-2)**2</code> 。注意：一个负底数配上一个分数

	指数是不被支持的，例如: $(-2)^{**}0.5$ 它将得出一个空字符串。不过像 $(-2)^{**}2$ 和 $(-2)^{**}2.0$ 这样都是支持的。
- ! ~ & *	<p>负号(-):虽然它和减运算符使用一样的符号，不过负号仅适用于单个元素或子表达式，就像这个例子里出现的两种情况一样: $-(3 / -x)$。顺便说一下，表达式中任何正号(+)都被忽略。</p> <p>逻辑-非(!):如果运算对象为空或 0，逻辑非的结果将是 1，也就是 "true"。否则，结果将为 0 (false)。例如: $!x \text{ or } !(y \text{ and } z)$。注意：运算符 NOT 除了没有 ! 的优先级高外，含义与 ! 相同。在 v1.0.46 及之后版本，连续的一元运算符，例如 $!!Var$ 是被允许的，因为它们运算的顺序是从右到左。</p> <p>按位-非(~):此运算符反转其运算对象的每一位。如果运算对象是一个浮点数，在运算前将被截成整数。如果运算对象是 0 到 4294967295 (0xffffffff) 之间的数，它将被视为一个<u>无符号的</u>32 位数值；否则，它将被视为一个<u>有符号的</u>64 位数值，例如 $\sim 0xf0f$ 得出 0xfffff0f0 (4294963440)。</p> <p>地址(&)和 Dereference(解引用) (*): $\&MyVar$ 返回变量 MyVar 的内容在内存里的地址。相反地，$*Expression$ 假设 Expression 已解析为一个数字的内存地址；它将取得数字 0 到 255 之间的地址里的字节(如果地址为 0，值将始终为 0；但必须避免是其它任何非法的地址，因为它可能使脚本崩溃；此外，NumGet()(See 10.) 获取数字通常会更快一些)。这些很少使用的运算符可以帮助我们使用 DII Call 结构体(See 18.3)以及处理含有二进制零的字符串。</p>
*	<p>乘(*):如果输入的都是整数的话，结果也是整数；反之，结果为浮点数。</p> <p>True divide(真除) (/):与 EnvDiv(See 21.2) 不同，即使输入的都是整数，真除运算也将得出浮点数的结果。例如，$3/2$ 得出 1.5 而不是 1，$4/2$ 得出 2.0 而不是 2。</p> <p>Floor divide(向下舍除) (//):如果输入的两个都是整数，双斜线运算符将使用高效的整数除法。例如，$5//3$ 得出 1，$5//-3$ 得出 -1。如果输入的任何一个浮点数，将执行浮点型除法并且截取数轴向左最接近整数的结果。例如，$5//3.0$ 得出 1.0 而 $5.0//3$ 得出 -2.0。虽然浮点型除法的结果是一个整数，但它以浮点数的格式存储，以便其它使用者能使用浮点格式。计算模数，请看 mod()(See 10.)。</p> <p>$==(See 21.3)$ 和 $/=(See 21.2)$: 运算符是用变量的值乘或除另一个值的一种简写方式。例如，$Var*=$ 产生的结果同 $Var:=$ 一样(不过前者执行得更好)。</p> <p>除以 0 将得到一个空结果(空字符串)。</p>
+	<p>加(+)和减(-)。 注意:在表达式中，任何在数学运算中出现的空值(空字符串)不假设为零，而被当作是错误，其导致那部分的表达式得出一个空字符串。例如，如果变量 X 为空，则表达式 $X+1$ 得到一个空值而不是 1。</p> <p>$+=(See 21.1)$ 和 $-(See 21.4)$: 运算符是递增或递减一个变量的简写方式。例如，$Var+=$ 产生的结果同 $Var:=$ 一样(不过前者执行得更好)。类似地，一个变量想要增加或减少 1 的话还可以使用 Var++, Var--, ++Var 或 --Var。</p>
<<	左移位(<<)和右移位(>>). 使用范例: $Value1 << Value2$ 。任何浮点数先被截为整数再参

>>	加计算。左位移(<<) 等价于 <code>Value1 * (2 ** Value2)</code> 。右位移(>>) 等价于 <code>Value1 / (2 ** Value2)</code> 并且取数轴向左最接近整数的结果；例如， <code>-3>>1</code> 结果为 <code>-2</code> 。
& ^ 	位与(&), bitwise-exclusive-or(位异或) (^)和位或()。三个中, & 的优先级最高而 优先级最低。任何浮点数在运算前都要先转换成整数。
.	<p>连接。点运算符用于将两个元素连接成一个字符串(应确保点的两边至少有一个空格)。你也可以忽略句点, 得到同样的结果(除非那有不明确的例如 <code>x -y</code> 或当右边的元素以 <code>++</code> 或 <code>--</code> 开头时)。当忽略句点时, 合并的两个元素之间应该至少有一个空格。</p> <p>示例(表达式方法): <code>Var := "The color is " . FoundColor</code></p> <p>示例(传统方法): <code>Var = The color is %FoundColor%</code></p> <p>子表达式同样可以被连接。例如: <code>Var := "The net price is " . Price * (1 - Discount/100)</code></p> <p>以句点(或其他任何运算符)开头的行, 自动添加到(See 8.)上一行。</p>
> < >= <=	大于(>), 小于(<), 大于等于(>=)和小于等于(<=)如果输入的任何一个都不是数字, 两边将按字母的顺序比较(一个由双引号括住的字符串例如 "55", 在此情况下总视为非数值)。只有在 StringCaseSense(See 27.13) 开启时, 比较才区分大小写。同时请看: Sort(See 27.12)
= = <> !=	等于 (=, 区分大小写的等于(=)和不等于(<> 或 !=)运算符 != 和 <> 在功能上相同。= 运算符和 = 一样, 除了当输入的任何一个就都不是数字时, 这种情况下, == 总是区分大小写, 而 = 却不区分(不区分大小写的方法由 StringCaseSense(See 27.13) 决定)。相比之下, <> 和 != 都服从于 StringCaseSense(See 27.13))。注意:一个由双引号括住的字符串例如 "55", 在此情况下总视为非数值。
NOT	逻辑非。除了它的优先级比 ! 低外, 其它的同运算符 ! 一样。例如, <code>not (x = 3 or y = 3)</code> 等同于 <code>!(x = 3 or y = 3)</code>
AND &&	两个都是逻辑与。例如: <code>x > 3 and x < 10</code> 。要提高性能, 可以应用 最少运算(See 10.) 。此外, 以 AND/OR/&&/ (或任何其他运算符) 开头的行将被自动添加到(See 8.)上一行。
OR 	两个都是逻辑或。例如: <code>x <= 3 or x >= 10</code> 。要提高性能, 可以应用 最少运算(See 10.) 。
?:	三元运算符[v1.0.46+]。这个运算符是替代 if-else 语句(See 17.11) 的简写。它计算它左边的条件来确定两个分支中的哪一个应该成为它最终的结果。例如, <code>var := x>y ? 2 : 3</code> 如果 x 大于 y, 将在 var 里存储 2; 反之, 其存储 3。要提高性能, 可以只计算胜出的分支(请看 最少运算(See 10.))。注意:由于兼容性的原因, 问号两边都至少要有一个空格才能被认出来(此问题可能会在未来的版本解决)。
:= += -= *= /= //= .= =	<p>赋值。对变量的内容进行操作再把结果存回同个变量(不过在 1.0.46 之前的版本, 这些只能用在一行最左边的运算符并且是前 5 个运算符才被支持)。最简单的赋值运算符就是冒号-等号(:=)(See 21.12), 用于将一个表达式的结果存在一个变量里。例如, <code>Var//= 2</code> 执行向下舍除, <code>Var</code> 除以 2, 接着将结果存回到 <code>Var</code>。类似地, <code>Var.= "abc"</code> 是 <code>Var := Var. "abc"</code> 的简写。</p> <p>不像大多数运算符, 赋值是从右到左开始运算的。所以, 像这样的一行: <code>Var1 := Var2 := 0</code> 首先将 0 赋值给 <code>Var2</code> 再将 <code>Var2</code> 赋值给 <code>Var1</code>。</p> <p>如果一个赋值运算被用作其它运算符的输入部分, 输入部分的值就是变量本身的价值。例如, 表达</p>

&= ^= >>= <<=	<p>式 <code>(Var+=2) > 50</code> 如果 <code>Var</code> 新增加了值后大于 50, 那么表达式为 <code>true</code>。而且也允许赋值操作被传递给 ByRef(See 10.), 或取得它的地址; 例如: <code>&(x:="abc")</code>。</p> <p>当赋值运算符为了避免语法错误或者要提供更直观的操作时, 它的优先级可以被自动提升。例如: <code>not x:=</code> 等价于 <code>not (x:=)</code>。类似地, <code>++Var := X</code> 等价于 <code>++(Var := X)</code>; 而 <code>Z>0 ? X:=2 : Y:=</code> 等价于 <code>Z>0 ? (X:=2) : (Y:=)</code></p> <p>已知限制(可能会在以后的版本中解决): 1)为了向后兼容, 当 <code>/=</code> 是表达式最左边的运算符且它不是多语句表达式的一部分时, 它执行向下舍除(See 9.)除非输入中有一个是浮点数(所有其他的情况下, <code>/=</code> 执行真除); 2)只有 <code>+= /-=</code> 是一行中最左边的运算符时, 日期/时间的算术(See 21.1)才被它们支持; 3)当运算符 <code>+=, -=</code> 和 <code>*=</code> 不是多语句表达式的一部分时才会把空白值当作零(否则, 这样的运算产生的结果还是空白)。</p>
<code>,</code>	<p>逗号(多语句)[v1.0.46+]。逗号被用来在一行中写入多个子表达式。它最常用来将多个赋值语句或函数调用放在一起。例如: <code>x:=, y+=, ++index, func()</code> [不过请看逗号的性能]。这些语句以从左到右的顺序执行。注意:以逗号(或任何其他运算符)开头的行将被自动添加到(See 8.)上一行。</p> <p>在 v1.0.46.01 及之后版本, 当逗号紧跟着一个变量和一个等号, 那个等号将被自动作为赋值(<code>:=</code>)(See 21.12)。例如, 下列所有的都是赋值: <code>x:=1, y=2, a=</code></p>
mod() round() abs()	<p>这几个以及其它的内置数学函数在这里(See 10.)有阐述。</p>

性能: 表达式赋值运算符 (`:=`)(See 21.12) 已被优化, 以便在执行像下面这样简单的赋值时能和非表达式运算符(`=`)(See 22.19))一样快:

```
x := y ; 和 x = %y% 的性能一样
x := 5 ; 和 x = 5 的性能一样
x := "literal string" ; 和 x = literal string 的性能一样
```

逗号运算符(`,`)强行将简单的赋值语句当作表达式来运算而不是在速度上进行优化。例如, 下列各自放一行的话, 速度将至少提高 20% (这个性能上的差异可能在未来的版本中被减少): `Var++, Var-=, Var:=,` `Var:="literal string", Var:=`。相反地, 如果表达式含有的是非平凡的比如函数调用, 逗号通常能提高性能(虽然一般不到 5%), 因为组合的表达式可以被一次全部运算而不是分开运算。

下列变量都内置在程序中, 并且可以被任何脚本引用。除了 [Clipboard](#), (See 30.6)[ErrorLevel](#)(See 30.8) 和[命令行参数](#), 这些变量都是只读的; 也就是说, 它们的内容不能被脚本直接改变。

目录

- 特殊字符: [A_Space](#), [A_Tab](#)
- 脚本属性: [command line parameters](#), [A_WorkingDir](#), [A_ScriptDir](#), [A_ScriptName](#), (...更多...)
- 日期和时间: [A_YYYY](#), [A_MM](#), [A_DD](#), [A_Hour](#), [A_Min](#), [A_Sec](#), (...更多...)
- 脚本设置: [A_IsSuspended](#), [A_BatchLines](#), [A_TitleMatchMode](#), (...更多...)
- 用户闲置时间: [A_TimeIdle](#), [A_TimeIdlePhysical](#)
- GUI 窗口和菜单栏: [A_Gui](#), [A_GuiControl](#), [A_GuiEvent](#), [A_EventInfo](#)
- 热键、热字符串和自定义菜单项: [A_ThisHotkey](#), [A_EndChar](#), [A_ThisMenuItem](#), (...更多...)
- 操作系统和用户信息: [A_OsVersion](#), [A_ScreenWidth](#), [A_ScreenHeight](#), (...更多...)
- 杂项: [A_Cursor](#), [A_CaretX](#), [A_CaretY](#), [Clipboard](#)(See 30.6), [ClipboardAll](#)(See 30.6), [ErrorLevel](#)(See 30.8)
- 循环: [A_Index](#), (...更多...)

特殊字符

A_Space	此变量包含了一个空格字符。详见 AutoTrim (See 22.3)。
A_Tab	此变量包含了一个 tab 字符。详见 AutoTrim (See 22.3)。

脚本属性

1, 2, 3 等等	当一个脚本与命令行参数一起被启动时，这些变量就被自动创建。它们可以像正常变量那样被引用和修改(例如: %1%)。而变量 %0% 表示传递的参数的个数(如果没有，值为 0)。详见 命令行参数 (See 8.)。
A_WorkingDir	脚本的当前工作目录，是访问文件的默认路径。除非是根目录，否则目录结尾不包括反斜线。两个例子: C:\ 和 C:\My Documents。可以使用 SetWorkingDir (See 16.33)来改变工作目录。
A_ScriptDir	当前脚本所在目录的完全路径。为了向后兼容 AutoIt v2，只为 .aut 脚本包含路径末尾的反斜线(甚至根分区也一样)。一个对 .aut 脚本的例子: C:\My Documents\
A_ScriptName	当前脚本的文件名称，不带路径，例如: MyScript.ahk。
A_ScriptFullPath	结合了上面两个变量的脚本的完全路径，例如: C:\My Documents\My Script.ahk
A_LineNumber	当前脚本中正在执行的那一行的行号(或是它的某个 #Include 文件 (See 17.1))。这个行号同 ListLines (See 22.11) 显示的一致；它对错误报告会比较有用，比如这个例子: MsgBox 不能写入记录文件(行号 %A_LineNumber%)。 由于 已编译的脚本 (See 8.)将它所有的 #Include 文件 (See 17.1)合 并进一个大脚本，它的行号可能和它以非编译模式运行时不一样。
A_LineFile	A_LineNumber 所属的文件名称及其完全路径，除非当前行属于未编译脚本的某个 #Include 文件 (See 17.1)，否则它会和 A_ScriptFullPath 一样。
A_ThisFunc [v1.0.46.16+]	当前正在执行的 自定义的函数 (See 10.)名称(如果没有就是空白)；例如: MyFunction
A_ThisLabel	当前正在执行的标签(子程序)的名称(如果没有就是空白)；例如: MyLabel。当脚本执行

[v1.0.46.16+]	Gosub (See 17.8)/ Return (See 17.19) 或 Goto (See 17.9) 时它的值会更新。此变量也会被像 timers (See 17.21), GUI threads (See 19.3), menu items (See 22.13), hotkeys (See 4.), hotstrings (See 5.), OnClipboardChange (See 30.6) 和 OnExit (See 17.17) 那样自动调用标签的命令更新。不过，在执行过程中从上面的语句“掉落”到一个标签时， A_ThisLabel 不会更新，还是保留之前的值。同时请看： A_ThisHotkey
A_AhkVersion	在 1.0.22 之前的版本里，此变量为空白。之后它包含了当前运行脚本的 AutoHotkey 的版本号，比如 1.0.22。对于已编译的脚本(See 8.)，会报上原先用来编译它的版本号。格式化的版本号使得脚本可以使用 > 或 >= 来检查 A_AhkVersion 是否大于某些低版本号，例如：if A_AhkVersion >= 1.0.25.07
A_AhkPath	对于未编译的脚本而言：是运行当前脚本的 EXE 文件的完全路径和文件名称。例如： C:\Program Files\AutoHotkey\AutoHotkey.exe 对于已编译的脚本(See 8.)而言：除了通过注册表项 HKEY_LOCAL_MACHINE\SOFTWARE\AutoHotkey\InstallDir 来得到 AutoHotkey 的目录外，其它和上面的一样。如果没有那个注册表项， A_AhkPath 将为空白。
A_IsCompiled	如果脚本作为一个已编译的 EXE(See 8.) 运行，它就包含 1，如果不是，则为空。
A_ExitReason	最近一次脚本被要求退出的原因。除非脚本有个 OnExit (See 17.17) 子程序并且这个子程序正在运行或已被一个退出的尝试调用过至少一次，否则此变量为空。详见 OnExit (See 17.17)。

日期和时间

A_YYYY	当前 4 位数的年份(例如 2004)。与 A_Year 同义。注意：要取得一个适用于你的区域和语言设置的格式化的时间或日期，可以使用 " FormatTime (See 27.1), OutputVar " (时间和长日期格式) 或者 " FormatTime (See 27.1), OutputVar,, LongDate " (获取长日期格式)。
A_MM	当前 2 位数的月份(01-12)。与 A_Mon 同义。
A_DD	当前月份的 2 位数的日期(01-31)。与 A_MDay 同义。
A MMMM	在当前的用户语言里当前月份的全称，例如 July
A MMM	在当前的用户语言里当前月份的缩写，例如 Jul
A_DDDD	在当前的用户语言里当前星期几的全称，例如 Sunday
A_DDD	在当前的用户语言里当前星期几的 3 个字母的缩写，例如 Sun
A_WDay	当前星期几的 1 位数(1-7)。1 表示星期天。
A_YDay	当前年份的天数(1-366)。变量的值是没有填充零的，例如取得 9，而不是 009。要取得一个用零填充的值，可使用： FormatTime (See 27.1), OutputVar, , YDay0
A_YWeek	当前的年份和星期数(例如 200453)按照 ISO 8601。要将年份和星期数分离开来，可以使用语句 StringLeft (See 27.15), Year , A_YWeek , 4 和 StringRight (See 27.15), Week , A_YWeek , 2。 A_YWeek 的精确定义：如果 Week 包含了新年的一月一日并有四

	天或更多在内，那么它被称为第一个星期。否则，它是前一年的最后一周，而且下一周才是新年的第一周。
A_Hour	当前 24 小时制中的 2 位小时数(00-23) (例如, 17 表示 5pm)。要获得 12 小时制以及 AM/PM 指示，参此例: FormatTime (See 27.1), OutputVar, , h:mm:ss tt
A_Min	当前的 2 位分钟数(00-59)。
A_Sec	当前的 2 位秒数(00-59)。
A_MSec	当前的 3 位毫秒数(000-999)。要移除前置的零，参此例: Milliseconds := A_MSec + 0
A_Now	YYYYMMDDHH24MISS (See 16.26) 格式的当前本地时间。注意：时间和日期的算术可以用 EnvAdd (See 21.1) 和 EnvSub (See 21.4) 来完成。此外， FormatTime (See 27.1) 可以按照你的区域设置或首选项来格式化日期和/或时间。
A_NowUTC	YYYYMMDDHH24MISS (See 16.26) 格式的当前世界协调时间(UTC)。UTC 本质上和格林尼治标准时间(GMT)是一样的。
A_TickCount	<p>计算机重启后经过的毫秒数。通过将 <code>A_TickCount</code> 存到一个变量里，然后把最新的 <code>A_TickCount</code> 的值减去那个变量，就能测得消逝的时间。例如：</p> <pre>StartTime := A_TickCount Sleep, 1000 Elapsed := A_TickCount - StartTime MsgBox, 已经消逝了 %Elapsed% 毫秒。</pre> <p>如果你需要比 <code>A_TickCount</code> 的 10ms 更高的精确度，请用 QueryPerformanceCounter()(See 18.3)。</p>

脚本设置

A_IsSuspended	如果脚本挂起(See 17.23)则包含 1; 否则为 0。
A_BatchLines	(与 <code>A_NumBatchLines</code> 同义)由 SetBatchLines (See 17.20) 来设置当前的值。例如: 200 或 10ms (根据格式)。
A_TitleMatchMode	由 SetTitleMatchMode (See 28.8) 设置当前的模式: 1, 2, 3 或 RegEx。
A_TitleMatchModeSpeed	由 SetTitleMatchMode (See 28.8) 设置当前的匹配速度(fast 或 slow)。
A_DetectHiddenWindows	由 DetectHiddenWindows (See 28.5) 设置当前的模式(On 或 Off)。
A_DetectHiddenText	由 DetectHiddenText (See 28.4) 设置当前的模式(On 或 Off)。
A_AutoTrim	由 AutoTrim (See 22.3) 设置当前的模式(On 或 Off)。
A_StringCaseSense	由 StringCaseSense (See 27.13) 设置当前的模式(On, Off 或 Locale)。
A_FormatInteger	由 SetFormat (See 21.10) 设置当前的整数格式(H 或 D)。
A_FormatFloat	由 SetFormat (See 21.10) 设置当前的浮点数格式。
A_KeyDelay	由 SetKeyDelay (See 20.14) 设置当前的延迟(总是十进制数，不是十六进制)。此延迟是针对传统的 <code>SendEvent</code> 模式，不是 SendPlay (See 20.12)。

A_WinDelay	由 SetWinDelay(See 28.9) 设置当前的延迟(总是十进制数, 不是十六进制)。
A_ControlDelay	由 SetControlDelay(See 28.1.13) 设置当前的延迟(总是十进制数, 不是十六进制)。
A_MouseDelay	由 SetMouseDelay(See 23.8) 设置当前的延迟(总是十进制数, 不是十六进制)。此延迟是针对传统的 <code>SendEvent</code> 模式, 不是 <code>SendPlay</code> (See 20.12)。
A_DefaultMouseSpeed	由 SetDefaultMouseSpeed(See 23.7) 设置当前的速度(总是十进制数, 不是十六进制)。
A_IconHidden	如果托盘图标当前被隐藏则为 1; 否则为 0。可以用 #NoTrayIcon(See 22.1) 或 Menu(See 22.13) 命令来隐藏图标。
A_IconTip	除非用 Menu(See 22.13) , <code>tray, tip</code> 已经为托盘图标指定了自定义提示--这种情况下它就是此提示的文本; 否则为空。
A_IconFile	除非用 Menu(See 22.13) , <code>tray, icon</code> 已经指定了自定义托盘图标--这种情况下它就是此图标文件的完全路径和名称; 否则为空。
A_IconNumber	如果 <code>A_IconFile</code> 为空, 则它也为空。否则, 它是 <code>A_IconFile</code> 里的图标编号(一般是 1)。

用户闲置时间

A_TimeIdle	从系统最后一次接收键盘、鼠标或其它输入后所消逝的毫秒数。这用来判断用户是否离开是有用的。除了操作系统是 Windows 2000, XP 或之后的版本外, 此变量将为空。用户物理的输入和由 任何程序或脚本 (例如 Send(See 20.12) 或 MouseMove(See 23.6) 命令)产生的模拟输入都将重置此变量为零。由于这个值倾向于以 10 的增量增加, 所以判断它是否等于另一个值便毫无意义了。取而代之, 可以检查它是否大于或小于另一个值。例如: <code>IfGreater, A_TimeIdle, 600000, MsgBox</code> , 上一次键盘或鼠标的活动至少是 10 分钟前的事了。
A_TimeIdlePhysical	和上面的一样, 不过在安装了相应的钩子(keyboard(See 20.2) 或 mouse(See 20.3))时会忽略模拟的键击和/或鼠标点击。如果没有安装任何钩子, 此变量就等于 <code>A_TimeIdle</code> 。如果只有一个钩子, 那么将只忽略它模拟的输入。在判断用户是否真的出现方面, <code>A_TimeIdlePhysical</code> 可能比 <code>A_TimeIdle</code> 更有用。

GUI 窗口和菜单栏

A_Gui	启动了 当前线程(See 30.19) 的 GUI(See 19.3) 窗口的编号。除非启动当前线程的是一个 Gui 控件、菜单栏项目或者比如 <code>GuiClose/GuiEscape</code> 事件, 那么此变量为空。
A_GuiControl	启动了 当前线程(See 30.19) 的与 GUI 控件相关连的变量名称。如果那个控件缺少一个 相关的变量(See 19.3) , <code>A_GuiControl</code> 代替它包含控件的文本/标题的前 63 个字符(这常用来避免给每个按钮一个变量名称)。 <code>A_GuiControl</code> 为空, 每当: 1) <code>A_Gui</code> 为空; 2) 一个 GUI 菜单栏项目或比如 <code>GuiClose/GuiEscape</code> 这样的事件启动了当前线程; 3) 控件缺少一个相关联的变量并且没有标题; 或者 4) 最初启动当前线程的控件已不存

	在 (可能导致 Gui Destroy (See 19.3))。
A_GuiWidth A_GuiHeight	当在一个 GuiSize 子程序 (See 19.3)引用时, 这些变量包含 GUI 窗口的宽和高。它们适用于窗口的客户端区域, 此区域包括标题栏、菜单栏和边框。
A_GuiX A_GuiY	这些变量包含了 GuiContextMenu (See 19.3) 和 GuiDropFiles (See 19.3) 事件的 X 和 Y 坐标。坐标是相对于窗口的左上角的。
A_GuiEvent 或 A_GuiControlEvent	<p>启动了当前线程(See 30.19)的事件类型。如果线程不是通过 GUI 动作(See 19.3)启动的, 此变量为空。反之, 它会包含以下字符串之一:</p> <p>Normal: 通过一个鼠标左键单击或者通过键击(箭头按键、TAB 键、空格、带下划线的快捷键, 等等)触发的事件。这个变量的值也可以给菜单栏项目和像 GuiClose 和 GuiEscape 这样的特殊事件来使用。</p> <p>DoubleClick: 此事件可由双击来触发。注意: 双击中的首次点击仍会引起 <i>Normal</i> 事件先被获取。换句话说, 子程序被启动了两次: 一次是首次点击, 然后是第二次。</p> <p>RightClick: 仅发生在 GuiContextMenu(See 19.3), ListViews(See 19.7) 和 TreeViews(See 19.8)。</p> <p>Context-sensitive values: 详见 GuiContextMenu(See 19.3), GuiDropFiles(See 19.3), Slider(See 19.4), MonthCal(See 19.4), ListView(See 19.7) 和 TreeView(See 19.8)。</p>
A_EventInfo	<p>包含下列事件的额外信息:</p> <ul style="list-style-type: none"> • OnClipboardChange 标签(See 30.6) • 鼠标滚轮热键(See 4.) (上滚/下滚) • RegisterCallback()(See 18.14) • GUI 事件(See 19.3), 也就是 GuiContextMenu(See 19.3), GuiDropFiles(See 19.3), ListBox(See 19.4), ListView(See 19.7), TreeView(See 19.8) 和 StatusBar(See 19.4)。如果没有事件的额外信息, A_EventInfo 包含 0。

注意: 与变量比如 A_ThisHotkey 不同, 每个 [thread](#)(See 30.19) 为 A_Gui, A_GuiControl, A_GuiX/Y, A_GuiEvent 和 A_EventInfo 包含了它自己的值。因此, 如果一个变量被另一个变量打断, 在恢复时仍可以在那些变量里看到它原来/正确的值。

热键、热字符串和自定义菜单项

A_ThisMenuItem	最近选择的 自定义菜单项 (See 22.13)的名称(如果没有就为空)。
A_ThisMenu	A_ThisMenuItem 所在的菜单的名称。
A_ThisMenuItemPos	指示 A_ThisMenu 中 A_ThisMenuItem 当前位置的编号。菜单的第一项就是 1, 第二项则为 2, 以此类推。菜单分隔符行也计算在内。如果 A_ThisMenuItem 为空或不再存在 A_ThisMenu 里, 此变量为空。如果 A_ThisMenu 本身已不再存在, 它也为。

A_ThisHotkey	最近执行的热键(See 4.)和热字符串(See 5.)的按键名称(如果没有则为空),例如 #z。如果当前线程(See 30.19)被其它热键中断,这个值将会改变,所以如果你之后需要在一个子程序中使用原来的值,确保马上将其复制到另一个变量。 当一个热键被第一次创建时--无论是通过脚本中的 Hotkey 命令(See 20.1.10)还是双冒号标签(See 4.)--它的键名和修饰键符号的顺序成为那个热键的固定名称。同时请看: A_ThisLabel
A_PriorHotkey	除了存放的是前一个热键的键名外,其它的同上。如果没有它将为空。
A_TimeSinceThisHotkey	从 A_ThisHotkey 被按下后消逝的毫秒数。如果 A_ThisHotkey 为空,其值为-1。
A_TimeSincePriorHotkey	从 A_PriorHotkey 被按下后消逝的毫秒数。如果 A_PriorHotkey 为空,其值为 -1。
A_EndChar	用户最近按下的用于触发非自动替换型热字符串(See 5.)的结束字符。如果不需 要结束符(因为 * 选项),此变量将为空。

操作系统和用户信息

ComSpec [v1.0.43.08+]	此变量同系统环境变量 ComSpec 的值一样(例如 C:\Windows\system32\cmd.exe)。常和 Run/RunWait(See 24.5)一起使用。注意:此变量没有前缀 A_。
A_Temp [v1.0.43.09+]	存放临时文件的文件夹的完全路径和名称(例如 C:\DOCUME~1\用户名\LOCALS~1\Temp)。它从下列的几个地方获得(按顺序): 1) 环境变量 TMP, TEMP 或 USERPROFILE; 2) Windows 目录。在 Windows 9x 上, 如果没有 TMP 和 TEMP 的存在, 将使用 A_WorkingDir 的值。
A_OSType	正在运行的操作系统的类型。值为 WIN32_WINDOWS (即 Windows 95/98/ME)或 WIN32_NT (即 Windows NT4/2000/XP/2003/Vista)。
A_OSVersions	下列字符串之一: WIN_VISTA [需要 v1.0.44.13+], WIN_2003, WIN_XP, WIN_2000, WIN_NT4, WIN_95, WIN_98, WIN_ME。例如: <pre>if A_OSVersions in WIN_NT4,WIN_95,WIN_98,WIN_ME ;注意: 逗号周围没空格。 { MsgBox 此脚本需要 Windows 2000/XP 及之后版本。 ExitApp }</pre>
A_Language	系统的默认语言,值为这些 4 位数字代码(See 30.10)中的一个。
A_ComputerName	网络上可查看到的计算机名称。
A_UserName	当前用户的登录名称。
A_WinDir	Windows 目录。例如: C:\Windows

A_ProgramFiles 或 ProgramFiles	Program Files 目录(例如 C:\Program Files)。在 v1.0.43.08 及之后版本, 前缀 A_ 可省略, 它用来帮助自然过渡到 #NoEnv (See 29.19)。
A_AppData [v1.0.43.09+]	当前用户的应用程序数据文件夹的完全路径和名称。例如: C:\Documents and Settings\Username\Application Data
A_AppDataCommon [v1.0.43.09+]	所有用户的应用程序数据文件夹的完全路径和名称。
A_Desktop	当前用户桌面文件的文件夹的完全路径和名称。
A/DesktopCommon	所有用户桌面文件的文件夹的完全路径和名称。
A_StartMenu	当前用户开始菜单文件夹的完全路径和名称。
A_StartMenuCommon	所有用户开始菜单文件夹的完全路径和名称。
A_Programs	当前用户开始菜单中 Programs 文件夹的完全路径和名称。
A_ProgramsCommon	所有用户开始菜单中 Programs 文件夹的完全路径和名称。
A_Startup	当前用户开始菜单中启动文件夹的完全路径和名称。
A_StartupCommon	所有用户开始菜单中启动文件夹的完全路径和名称。
A_MyDocuments	当前用户“我的文档”文件夹的完全路径和名称。与大多数变量不同, 如果此文件夹是一个驱动器的根目录, 末尾处不包括反斜线。例如, 它包含 M: 而不是 M:\
A_IsAdmin	<p>如果当前用户有管理员权限, 此变量值为 1。否则为 0。不过在 Windows 95/98/Me 下此变量始终为 1。</p> <p>在 Windows Vista 下, 一些脚本可能需要管理员权限才能正常的运行(比如一个会影响进程或窗口的脚本需要管理员权限来运行)。要实现此目的, 请在脚本的顶部添加下列语句(对已编译的脚本需要一些调整, 也许还需要传递参数):</p> <pre>if not A_IsAdmin { DllCall("shell32\ShellExecuteA", uint, 0, str, "RunAs", str, A_AhkPath , str, """ . A_ScriptFullPath . """ , str, A_WorkingDir, int, 1) ExitApp }</pre>
A_ScreenWidth A_ScreenHeight	<p>主显示器的宽度和高度, 以像素为单位(例如 1024 和 768)。</p> <p>要得到多显示器系统中其他显示器的尺寸, 请用 SysGet(See 22.21)。</p> <p>要得到整个桌面(即使它跨越了多个显示器)的宽和高, 可用下面的例子 (不过在 Windows 95/NT 下, 下面的两个变量都将被设置为 0):</p> <p>SysGet(See 22.21), VirtualWidth, 78 SysGet(See 22.21), VirtualHeight, 79</p> <p>此外, 可用 SysGet(See 22.21) 来获得显示器工作区域, 它会比显示器的整个</p>

	区域要小，因为没有包括任务栏和其他注册的桌面工具栏。
A_IPAddress1 到 4	计算机中前 4 个网卡的 IP 地址。

杂项

A_Cursor	<p>当前被显示的鼠标指针类型。它的值会是下列单词之一: AppStarting, Arrow, Cross, Help, IBeam, Icon, No, Size, SizeAll, SizeNESW, SizeNS, SizeNWSE, SizeWE, UpArrow, Wait, Unknown。那些和 size 类的指针写在一起的单词首字母缩写指示了方向，例如 NESW = NorthEast+SouthWest。手形的指针(点击和抓取)被分进 Unknown 类型里。</p> <p>在 Windows 95 中的已知限制: 如果此变量的内容被高频率(即每 500 ms 或更快)反复读取，就会干扰使用者双击鼠标的能力。没有已知的解决方案。</p>
A_CaretX A_CaretY	<p>当前文本插入符的 X 和 Y 坐标。除非使用了 CoordMode(See 22.6) 让坐标相对于整个显示器，否则它们相对于激活的窗口。如果没有激活的窗口或者无法确定插入符的位置，这两个变量将为空。</p> <p>下面这个脚本允许你四处移动插入符时通过自动更新的提示来显示它当前的位置。注意某些窗口(例如某些版本的 MicroSoft Word)将报告同个插入符位置而不管它真正的位置。</p> <pre>#Persistent SetTimer, WatchCaret, 100 return WatchCaret: ToolTip, X%A_CaretX% Y%A_CaretY%, A_CaretX, A_CaretY - 20 return</pre> <p>如果这些变量的内容被高频率(即每 500 ms 或更快)反复读取，就会干扰使用者双击鼠标的能力。没有已知的解决方案。</p>
Clipboard	操作系统的剪贴板内容，它能被读取或者写入。请看 Clipboard (See 30.6) 章节。
ClipboardAll	整个剪贴板的内容(例如格式设置和文本)。请看 ClipboardAll (See 30.6)。
ErrorLevel	请看 ErrorLevel (See 30.8)。
A_LastError	操作系统的函数 GetLastError() 返回的结果。详见 DIICall() (See 18.3) 和 Run/RunWait (See 24.5)。

循环

A_Index	当前循环重复的次数(64 位的整型数)。例如，脚本第一次执行循环体时，此变量的值为 1。详见 Loop (See 17.12)。
A_LoopFileName 等	此变量和其它相关的变量仅在 文件-循环 (See 16.31)中有效。
A_LoopRegName 等	此变量和其它相关的变量仅在 注册表-循环 (See 17.16)中有效。

A_LoopReadLine	请看 文件-读取循环(See 16.32) 。
A_LoopField	请看 分解循环(See 17.14) 。

环境变量由操作系统维护。在命令提示符中输入 `SET` 并回车，你就可以看到一个它们的列表。

一个脚本可以用 [EnvSet\(See 15.3\)](#) 创建一个新的环境变量或者改变一个已存在的环境变量的值。然而，这些添加和改变都是私有的；它们不会被系统其它的部分看到。不过还有一个例外，当一个脚本使用 [Run\(See 24.5\)](#) 或 [RunWait\(See 24.5\)](#) 来运行一个程序(甚至另一个脚本)：这个程序将继承一个复制自父脚本的环境变量，包括那些私有的环境变量。

在 v1.0.43.08 及之后版本，推荐在所有新脚本中通过下面的方式获得像 `Path` 这样的环境变量：

[EnvGet, OutputVar, Path\(See 15.2\)](#) ;解释请看 [#NoEnv\(See 29.19\)](#)。

- 每个变量可以包含多达 64 MB 的文本(这个限制可以通过 [#MaxMem\(See 29.15\)](#) 来增加)。
- 当给变量赋的新值比当前的更长时，将自动分配额外的系统内存。
- 占据巨大内存空间的变量可将其设置为空来释放内存，例如：`var := ""`
- 没有对一个脚本可以创建的变量个数做限制。程序被设计来支持至少几百万变量而不会出现一个性能上的巨大落差。
- 命令、函数和表达式通常能够接受 15 位精度的浮点数值；64 位带正负号的整数值，范围从 -9223372036854775808 (-0x8000000000000000) 到 9223372036854775807 (0x7FFFFFFFFFFFFFFF)。任何在范围之外的整数常量不被支持并可能产生不一致的结果。相反，整数的算术运算却可能产生溢出(例如：`0x7FFFFFFFFFFFFFFF + 1 = -0x8000000000000000`)。

翻译：第八单元 8thunit@gmail.com 修正：天堂之门 menk33@163.com 2008 年 9 月 27 日

10. 函数

- 介绍以及简单的例子
- 参数
- 可选参数
- 局部变量
- 优化布尔求值
- 在函数中使用子程序
- [Return, Exit](#) 和一般说明
- 用 [#Include](#) 在多个脚本中共享函数

- 函数库: 标准库和用户库
- 内置函数

函数和子程序([Gosub\(See 17.8\)](#))相似, 只不过它能从它的调用者那里接受参数(输入)。此外, 函数能随意地向它的调用者返回一个值。参考下面这个简单的函数, 它接受了两个数字并返回它们的和:

```
Add(x, y)
{
    return x + y ; "Return(See 17.19)" 需要一个表达式(See 9.)
}
```

上面的被称为函数**定义**, 因为它创建了一个名为 "Add"(不区分大小写)的函数, 并且确定无论谁要调用它都必须确切地提供两个参数(x 和 y)。要调用函数并将它的结果赋值给变量, 用 [:= \(See 21.12\)](#)**运算符**(See 21.12)。例如:

```
Var := Add(2, 3) ;数字 5 将存储在变量 var 中。
```

而且, 可以调用函数而不存储它的返回值:

```
Add(2, 3)
```

但是如果这样, 任何通过函数返回的值都将被抛弃; 因此除非函数除了它的返回值外还会产生其他一些效果, 不然此调用将无济于事。

一旦函数在[表达式\(See 9.\)](#)中被调用, 任何在它参数列表中的变量名都不能被附上百分号。相反, 原文字符串却要加上双引号。例如:

```
if InStr(MyVar, "fox")
    MsgBox MyVar 变量包含单词 fox。
```

在 v1.0.47.06 及之后版本里, 一个函数(甚至是[内置函数](#))可以通过百分号被动态地调用。例如, `%Var%(x, "fox")` 将调用一个名称存储在变量 Var 里的函数。相似地, `Func%A_Index%()` 将调用名称存储在指定数组元素([See 30.5](#))中的函数。所调用的函数必须在脚本中明确地定义, 也可通过 [#Include\(See 17.1\)](#) 或者非动态调用一个[函数库](#)。如果被调用的函数不存在, 或者给函数参数传递的值或类型不对, 那么包含此调用的表达式将产生一个空字符串。

最后, 函数可以被任何命令的参数调用 (除了像 [StringLen\(See 27.17\)](#) 命令中的那些 OutputVar 和 InputVar 参数)。不过, 那些不支持[表达式\(See 9.\)](#)的参数必须使用 "%" 作为前缀, 例如下面这个例子:

```
MsgBox % "答案是: " . Add(3, 2)
```

"%" 前缀也允许用在本来就支持表达式的参数里, 但它将被简单地忽略。

当一个函数被定义的时候, 它的参数都列在它的名称后面的括号中(它的名称和左括号之间不能有空格)。如果一个函数不接收任何参数, 就让括号空着; 例如: `GetCurrentTimestamp()`。

ByRef 参数: 从函数的观点来看, 参数本质上和[局部变量\(See 10.\)](#)一样, 除非它们像在下面这个例子中那样被定义为 **ByRef**:

```
Swap(ByRef Left, ByRef Right)
{
    temp := Left
    Left := Right
    Right := temp
}
```

在上面的例子中, **ByRef** 的使用导致各个参数变成了从调用者传递进来的变量的一个别名, 换句话说, 参数和调用者的变量在内存中都引用了相同的内容。这就允许了 **Swap** 函数通过移动 *Left* 的内容给 *Right* 来改变调用者的变量, 反过来也一样。

相比之下, 如果在上面的例子中没有使用 **ByRef** 的话, *Left* 和 *Right* 将是调用者的变量的复制, 因此 **Swap** 函数也将没有效果。

由于 [return\(See 17.19\)](#) 只能送回一个值给函数的调用者, 所以 **ByRef** 可以用来送回更多的结果。这是由函数向调用者传递进来的变量(通常为空)储存一个值来实现的。

当向函数传递大量字符串的时候, **ByRef** 可以提高性能并且通过回避复制字符串的需求来节约内存。同样, 使用 **ByRef** 送回一个长字符串给调用者往往比例如 **Return** 巨型字符串执行得更好。

已知限制:

- 不能向函数的 **ByRef** 参数传递[剪贴板\(See 30.6\)](#), [内置变量\(See 9.\)](#)或者[环境变量\(See 9.\)](#), 甚至当 [#NoEnv\(See 29.19\)](#) 在脚本里不存在的时候。传递一个内置变量给一个 **ByRef** 参数将导致显示一个错误对话框。
- 尽管一个函数能递归地调用自身, 但如果它传递它自己的一个[局部变量\(See 10.\)](#)或者非 **ByRef** 参数给它自身的 **ByRef**, 新一层的 **ByRef** 参数将引用它自身那个局部变量的名称而不是之前的层的。不过, 这个矛盾在一个函数传递一个[全局变量](#), [静态变量](#)或者 **ByRef** 参数给它自身的时候不会发生。
- 如果一个参数在函数调用里被解析为一个变量(比如 *Var* 或 *++Var* 或 *Var*=*), 它左边或右边的其他参数能在它被传递给函数前改变那个变量。比如, 当 *Var* 最初为 0 时, 即使当函数的首个参数不是 **ByRef** 时, *func(Var, Var++)* 仍会意外地传递 1 和 0。由于这种行为是违反常规的, 所以可能在将来放出的版本中改变。

在定义一个函数时, 它的一个或多个参数可以被标记为可选。这可以通过给它们添加一个等号和一个默认值来实现。下面的函数的参数 *Z* 已标记为可选:

```
Add(X, Y, Z = 0)
{
    return X + Y + Z
}
```

当调用者传递满三个参数给上面的函数时, *Z* 的默认值将被忽略。但是当调用者仅传递了两个参数时, *Z* 自动地获取默认值 0。

不能将可选参数孤立在参数列表的中间。换句话说，位于首个可选参数右边的所有参数也必须标记为可选。

在 v1.0.46.13 及之后版本中，[ByRef 参数](#)也可支持默认值；例如：`Func(ByRef p1 = "")`。每当调用者省略了这样一个参数，函数会创建一个局部变量并赋上默认值；换句话说，函数会表现得像没有 [ByRef](#) 这个关键词一样。

一个参数的默认值必须是下列形式之一：`true`, `false`, 一个原义的整数，一个原义的浮点数，或一个引用的/原义的字符串例如 "fox" 或 "" (但在 1.0.46.13+ 之前的版本只允许 "")。

局部变量

所有在函数内部引用或创建的变量默认都是**局部的**（除了 [Clipboard\(See 30.6\)](#), [ErrorLevel\(See 30.8\)](#) 和 [A_TimeIdle\(See 9.\)](#) 这些内置变量）。每个局部变量的内容都只能在函数内可见。所以，一个局部变量可以和一个全局变量有着相同的名字却有着不同的内容。最后，所有的局部变量每次在函数被调用时都以空值开始。

全局变量

要在一个函数里引用一个存在的全局变量(或创建一个新的)，需要在使用它之前先声明此变量为 `global`。例如：

```
LogFile(TextToLog)
{
    global LogFileName ;这个全局变量先前已经在此函数外赋过值了。
    FileAppend, %TextToLog%`n, %LogFileName%
}
```

如果一个函数需要引用或创建大量的全局变量，可以通过将它的首行设为单词"`global`"或者声明一个局部变量来假设它所有的变量都是全局的(它的参数除外)。例如：

```
SetDefaults()
{
    global ;如果在此函数的首行有比如"local MyVar"这样的词，那么这个单词可以被省略。
    Var := 33 ;将 33 赋值给一个全局变量，如果需要，首次可创建变量。
    local x, y:=0, z ;局部变量必须用这种形式来声明，不然它们会被假定为全局变量。
    ; ...等等。
}
```

这种假设的全局模式也可以被函数用来创建一个全局数组([See 30.5](#))，例如一个赋值给 `Array%A_Index%` 的循环。

静态变量

一个变量可以被声明为 `static` 来使它的值在多次调用期间被记住。例如：

```

LogFile(TextToLog)
{
    static LineCount = 0
    LineCount += 1 ;保持自身的累加(它的值在多次调用期间能被记住)。
    global LogFileName
    FileAppend, %LineCount%: %TextToLog%`n, %LogFileName%
}

```

静态变量一般都是隐式的局部变量。在 1.0.46 之前的版本，所有的静态变量都以空值开始；所以要检查一个静态变量首次被使用的唯一办法就是检查它是否为空值。在 v1.0.46 及之后的版本，一个静态变量可以初始化为除了 "" 外的东西，通过其后跟 := 或 = 以及下列之一：true, false, 一个原义的整数，一个原义的浮点数，或一个引用的/原义的字符串比如 "fox"。例如：static X:=0, Y:="fox"。每个静态变量都只初始化一次(在脚本执行之前)。

关于局部和全局的更多信息：

在下面例子中，通过逗号将多种变量分隔开从而在同一行声明：

```

global LogFileName, MaxRetries := 5
static TotalAttempts = 0, PrevResult

```

在 v1.0.46 及之后版本，一个局部或者全局变量可以在同一行被初始化，通过在它的声明后加上 := 或 = 以及任何[表达式\(See 9.\)](#)(在声明中运算符 = 和 := 作用相同)。在一个特定行有多个声明时，由于[性能原因\(See 9.\)](#)(不像普通的[逗号分隔语句\(See 9.\)](#))每个有初始化设定的声明都被作为单独的行执行。与[静态变量初始化设定](#)不同，局部变量和全局变量的初始化设定在每次函数被调用时都会执行，但是仅当控制流实际能到达它们这时。换句话说，在一行中写下 local x = 0 和在两个单独的行写下 local x 以及 x = 0 的作用是一样的。

因为单词 *local*, *global* 和 *static* 在脚本启动时会被立即处理，所以不能用 [IF 语句\(See 17.11\)](#)来有条件地声明变量。换句话说，在一个 IF 或 ELSE [区块\(See 17.2\)](#)里的声明，声明与函数末尾大括号之间的所有行都会无条件地生效。同时注意目前还不可能声明一个动态变量比如 *global Array%i%*。

在一个函数里，任何动态变量引用比如 *Array%i%* 常常会解析为一个局部变量，除非那个名称的变量不存在，在这时才会使用全局变量如果它存在的话。如果两者都不存在，那么需要变量先被创建才能使用，它会被创建为一个局部变量，除非[假设全局模式](#)已生效。因此，仅当函数被定义为[假设全局](#)函数时，它才能够手动地创建一个全局[数组\(See 30.5\)](#)(比如使用 *Array%i% := A_Index* 这样的方法)。

对于创建[数组\(See 30.5\)](#)的命令(例如 [StringSplit\(See 27.21\)](#))，如果[假设全局模式](#)未生效或者数组的第一个元素已声明为局部变量(将函数的一个参数传递给它也可以 -- 即使那个参数是 [ByRef\(See 10.\)](#) 的 -- 因为参数和局部变量很相似)，那么得出的数组就是局部的。相反的，如果首个元素已被[声明为全局](#)，那么创建的是全局数组。[StringSplit\(See 27.21\)](#) 的首个元素是 *ArrayName0*。对于其他创建数组的命令比如 [WinGet List\(See 28.15\)](#)，首个元素是 *ArrayName*(即没有数字)。

常见疑点：在脚本启动时对变量的任何**非**动态引用都将创建那个变量。例如：在脚本启动的时候，在函数外使用 *MsgBox %Array1%* 将创建全局的 *Array1*。相反的，在脚本启动时，在函数内部使用 *MsgBox %Array1%* 将创建 *Array1* 作为函数局部变量之一(除非[假设全局](#)已生效)。

当在[表达式](#)(See 9.)中使用 *AND*, *OR* 和[三元运算符](#)(See 9.)时，他们为提高性能而优化(不管当前是否有函数调用)。通过拒绝计算一个表达式里那些不能影响它的最终结果的部分来实行优化操作。要阐明此观点，请看下例：

```
if (ColorName <> "" AND not FindColor(ColorName))
    MsgBox 没找到 %ColorName%.
```

在上例中，如果 *ColorName* 变量为空，*FindColor()* 函数永远不会被调用。这是因为 *AND* 左侧的结果将为 *false*，因此它的右侧不可能让最终结果输出为 *true*。

由于此特性，所以必须明白，如果在 *AND* 或 *OR* 右侧调用函数，函数可能永远会产生任何副作用(例如改变一个全局变量的内容)。

同时也要注意在嵌套的 *AND* 和 *OR* 中串联的求值优化。例如，在下面的表达式里，每当 *ColorName* 为空，就只需最左侧的比较就能执行了。这是因为左侧的表达式已经能确保最终的结果：

```
if (ColorName = "" OR FindColor(ColorName, Region1) OR FindColor(ColorName,
Region2))
    break ;搜索内容为空，或者已经有一个匹配。
```

从上面例子来看，任何耗时的函数一般都应该在 *AND* 或 *OR* 的右侧调用从而提高性能。这个技术还能用来阻止函数的某个参数在传递一个不恰当的值比如一个空字符串时函数被调用。

在 v1.0.46 及之后的版本，[三重条件运算符 \(?:\)](#)(See 9.) 也通过不计算丢失的分支来优化求值。

尽管一个函数不能包含其他函数的[定义](#)，但它可以包含子程序。与其他子程序一样，可使用 [Gosub](#)(See 17.8) 来启动它们，[Return](#)(See 17.19) 来返回结果(这时候 *Return* 属于 *Gosub* 而不是函数)。

已知限制：当前，在整个脚本中每个子程序的名称(标签)必须是独一无二的。如果存在重复的标签，程序将会通知你。

如果一个函数使用 [Gosub](#)(See 17.8) 跳转到一个公共子程序(在函数括号以外的子程序)，那么在外部的所有变量都是全局变量，而函数自己的[局部变量](#)在子程序返回之前将不可用。

虽然一般不鼓励使用 [Goto](#)(See 17.9) 命令，但是它能用来在同个函数内跳转到函数的其他位置。这能帮助我们简化很多返回点的复杂函数，所有那些返回点需要在返回之前做一些清理。

尽管 [Goto](#)(See 17.9) 那些函数外部的目标被忽略，但是对函数来说可以用 [Gosub](#)(See 17.8) 进入一个外部的/公用的子程序然后从那里使用 [Goto](#)。

一个函数可以包含从外部调用的子程序，比如[定时器](#)(See 17.21), [GUI](#)(See 19.3) 和[菜单项](#)(See 22.13)。通常将它们封装在不同的文件中供 [#Include](#)(See 17.1) 使用，这将防止它们干扰脚本的[自动执行部分](#)(See 8.)。不过，还有如下限制：

- 如果它们的函数是正常地调用，这样的子程序只能用[静态变量](#)和[全局变量](#)(不能是[局部变量](#))。这是因为一个子程序[线程](#)(See 30.19)打断一个函数调用线程(或者反过来也一样)将能够改变被打断的

线程看到的局部变量的值。此外，每当函数返回它的调用者时，它所有的局部变量都被清空从而释放它们的内存。

- 这样的子程序应该只将全局变量(See 10.)(不是静态变量(See 10.))用作 GUI 控件变量(See 19.3)。
- 当一个子程序线程(See 30.19)进入一个函数时，任何由此线程引用的动态变量(See 30.5)将被作为全局变量对待(包括创建数组的命令)。

如果一个函数内部的执行流在到达函数的结束大括号之前遇上一个 Return(See 17.19)，那么函数将结束并返回一个空值(空字符串)给它的调用者。每当函数明确地省略 Return(See 17.19) 的参数时也将返回一个空值。

当一个函数使用 Exit (See 17.6)命令来结束当前线程(See 30.19)的时候，它的调用者根本不会收到返回值。例如：在语句 `Var := Add(2, 3)` 中如果 `Add()` 退出，那么 `Var` 将不做改变。如果函数遇上运行错误比如运行(See 24.5)了一个不存在的文件(当 UseErrorLevel(See 24.5) 无效时)，那么也将发生同样的事情。

一个函数可以改变 ErrorLevel(See 30.8) 的值为了返回一个更容易记忆的额外值。

要在调用函数时使用一个或多个空值(空字符串)，可以像这个例子一样使用一对空的双引号：

```
FindColor(ColorName, "")
```

因为调用函数不会启用新的线程(See 30.19)，所以函数做出的任何设置比如 SendMode(See 20.13) 和 SetTitleMatchMode(See 28.8) 的改变也都会影响到它的调用者。

一个函数的调用者可能传递一个不存在的变量或者数组(See 30.5)元素给它，这在函数期望这个相应的参数被 ByRef 时将变得很有用。例如：调用 `GetNextLine(BlankArray%i%)` 将自动地创建一个局部或全局变量 `BlankArray%i%` (根据调用者是否在函数内并且是否有假设的全局模式在起作用)。

当在一个函数里使用时，ListVars(See 22.12) 会显示一个函数的局部变量和它们的内容。这能帮助我们调试脚本。

你会发现如果给复杂的函数的特定变量加一个独特的前缀，将使函数更易于阅读和维护。例如，用 "p" 或 "p_" 开头来命名函数的每个参数能让你一眼就辨别出它们的特性，特别是当函数还有很多局部变量来争着吸引你眼球的时候。相似地，"r" 或 "r_" 可以用在 ByRef 参数的前面，"s" 或 "s_" 可以用在静态变量的前面。

单个正确的大括号(OTB) 类型(See 17.2)也可以用来定义函数。例如：

```
Add(x, y) {
    return x + y
}
```

可以用 [#Include\(See 17.1\)](#) 指令(甚至在脚本的顶部)从外部文件加载函数。

说明：当脚本的执行流遇到函数定义时，它会跳过函数(用一种瞬间完成的方法)，并在它的结束大括号的下一行恢复执行。所以执行流不会从上面掉进一个函数，也不会因为在脚本的顶部有一个或多个函数而影响到[自动执行部分\(See 8.\)](#)。

一个脚本不一定非要使用 [#Include\(See 17.1\)](#) 来调用外部文件中的函数。要达到这个目的，一个与函数同名的文件必须存在于下列的库目录之一中：

`%A_MyDocuments%(See 9.)\AutoHotkey\Lib\ ;用户库。此目录是可选的；也可以完全不存在。`

当前运行的 `AutoHotkey.exe` 的路径`\Lib\ ;标准库。同样是可选的。`

例如，假设一个脚本调用一个不存在的函数 `MyFunc()`，程序将在用户库里搜索名为 `MyFunc.ahk` 的文件。如果找不到，它将在标准库里继续搜索。如果仍未找到并且函数的名字里有一个下划线(比如 `MyPrefix_MyFunc`)，那么程序将在两个库里搜索名为 `MyPrefix.ahk` 的文件，如果存在的话会加载它。这使得 `MyPrefix.ahk` 可以包含函数 `MyPrefix_MyFunc` 以及其他相关的函数名以 `MyPrefix_` 开头的函数。

虽然一个库文件通常只包含一个和它的文件名同名的函数，但它也可以包含仅被同名函数调用的私有函数和子程序。然而，这些函数应该具有相当独特的名称，因为它们仍会在全局命名空间里；也就是说，可以从脚本的任意位置调用它们。

如果一个库函数使用 [#Include\(See 17.1\)](#)，那么 `#Include` 的工作目录就是库函数自身所在的目录。这能被用来创建一个重定向到一个包含此函数和其他与之相关的函数的巨大的库文件。

[脚本编译器 \(ahk2exe\)\(See 8.\)](#) 同样支持库函数。不过，它需要编译器目录的上层目录里存在一个复制的 `AutoHotkey.exe`(通常已是如此)。如果 `AutoHotkey.exe` 不存在，编译器仍可以运行，不过无法自动地包含库函数。

包含一个库函数也会执行地同其他函数一样好，因为它们在脚本开始执行之前已经被预加载了。

在一个内置函数的参数列表末尾的任何可选的参数是可以被完全省略的。例如，`WinExist("Untitled - Notepad")` 是有效的，因为它的另外三个参数将被视为空。

如果脚本定义了一个和内置函数同名的函数，内置函数将被覆盖。例如：一个脚本将调用它自定义的 `WinExist()` 函数来代替标准的那个。

存在于 DLL 文件里的外部函数可以通过 [DllCall\(\)\(See 18.3\)](#) 来调用。

经常使用的函数

FileExist(FilePattern): 如果 `FilePattern` 不存在(如果未指定绝对路径，程序将假设 `FilePattern` 在 `A_WorkingDir(See 9.)`)就返回一个空值(空字符串)。否则，它将返回首个匹配的文件或文件夹的属性字

串(See 16.10)("RASHNDOCT"的子集)。如果文件没有属性(罕见), 将返回 "X"。*FilePattern* 可以是文件或文件夹的准确的名称, 或者包含通配符 (*) 或 (?). 因为空字符串被视为 "false", 所以函数的返回值总是可以作为一个准布尔值来使用。例如: 如果文件存在, 语句 *if FileExist("C:\My File.txt")* 将为 true, 反之为 false。相似地, 只有当文档存在并且是一个目录, 语句 *if InStr(FileExist("C:\My Folder"), "D")* 才为 true。相关命令: [IfExist\(See 16.27\)](#) 和 [FileGetAttrib\(See 16.10\)](#).

GetKeyState(KeyName(See 7.) [, "P" or "T"]): 与 [GetKeyState](#) 命令(See 20.7)不同, 后者是按下的返回 D, 弹起的返回 U; 而这个函数是如果键是按下的就返回 true (1), 弹起的返回 false (0)。如果参数 *KeyName(See 7.)* 是无效的, 将返回空字符串。请看 [GetKeyState\(See 20.7\)](#) 来了解其他返回值和用法。

InStr(Haystack, Needle [, CaseSensitive = false, StartingPos = 1]): 返回 *Haystack* 字符串中首个匹配的 *Needle* 字符串的位置。与 [StringGetPos\(See 27.14\)](#) 不同, 第一个字符的位置是 1; 这是因为 0 同义于 "false", 会使其成为一个直观的"未找到"的指示。如果参数 *CaseSensitive* 被省略或者为 false, 则搜索将不区分大小写(不区分的模式取决于 [StringCaseSense\(See 27.13\)](#) 命令); 否则就得精确地匹配大小写。如果省略 *StartingPos*, 其默认为 1(*Haystack* 字符串的起点)。否则, 指定 2 从 *Haystack* 的第二个字符开始搜索, 3 从第三个开始, 等等。如果 *StartingPos* 超出 *Haystack* 的长度, 那么函数将返回 0。如果 *StartingPos* 为 0, 那么搜索将逆序执行(从右到左)以便找到在最右边匹配的结果。不管 *StartingPos* 的值是多少, 返回的位置将始终相对于 *Haystack* 的首个字符而言。例如: 在 "123abc789" 中 "abc" 的位置永远是 4。相关链接: [RegExMatch\(\)](#)(See 18.12), [IfInString\(See 27.3\)](#) 和 [StringGetPos\(See 27.14\)](#)。(译注: 例如 `InStr("123abc789","abc",false,1)`)

RegExMatch(Haystack, NeedleRegEx [, UnquotedOutputVar = "", StartingPos = 1]):
请看 [RegExMatch\(See 18.12\)\(\)](#)(See 18.12)。

RegExReplace(Haystack, NeedleRegEx [, Replacement = "", OutputVarCount = "", Limit = -1, StartingPos = 1]): 请看 [RegExReplace\(See 18.13\)\(\)](#)(See 18.13)。

SubStr(String, StartingPos [, Length]) [v1.0.46+]: 在 *String* 字符串中从 *StartingPos* 起始点开始向右复制不超过 *Length* 长度的字符的子字符串(如果参数 *Length* 省略, 就默认为"所有字符")。对于 *StartingPos*, 指定 1 则从首个字符开始, 2 则从第二个字符开始, 以此类推(如果 *StartingPos* 超出了 *String* 的长度, 将返回一个空字符串)。如果 *StartingPos* 小于 1, 将被视为从字符串末尾开始的位置。例如, 0 提取最后一个字符, -1 提取最后两个字符(但是如果 *StartingPos* 超过了字符串的左侧末尾, 提取又会从左侧首个字符开始)。*Length* 是要获取的字符的最大数目(每当字符串剩余部分太短的时候, 获取的长度会比最大数目少一些)。指定一个负的 *Length* 从而在返回的字符串的末尾省略这么多个字符(如果省略了全部或更多字符, 将返回一个空字符串)。相关链接: [RegExMatch\(\)](#)(See 18.12), [StringMid\(See 27.19\)](#), [StringLeft/Right\(See 27.15\)](#), [StringTrimLeft/Right\(See 27.22\)](#)。

StrLen(String): 返回 *String* 的长度。如果 *String* 是 [ClipboardAll\(See 30.6\)](#) 先前分配的变量, 将返回它的总大小。相关命令: [StringLen\(See 27.17\)](#)。

WinActive([WinTitle, WinText, ExcludeTitle, ExcludeText]): 如果激活的窗口匹配指定的条件, 则返回窗口的唯一 ID (HWND)(See 28.15)。如果不匹配, 函数返回 0。因为所有非零的值都视为 "true", 所以每当匹配了 *WinTitle* 的窗口被激活时语句 *if WinActive("WinTitle")* 都是 true。最后, *WinTitle* 支持 ahk_id, ahk_class 以及其他特殊字符串。关于这些和窗口激活方面的其他信息, 详见 [IfWinActive\(See 28.6\)](#)。

WinExist([WinTitle**, **WinText**, **ExcludeTitle**, **ExcludeText**])**: 返回以十六进制整数表示的首个匹配窗口的唯一 **ID (HWND)**(See 28.15)(如果没有就是 0)。因为所有非零的值都视为 "true", 所以每当匹配了 **WinTitle** 的窗口存在时语句 *if WinExist("WinTitle")* 都是 true。最后, **WinTitle** 支持 **ahk_id**, **ahk_class** 以及其他特殊字符串。关于这些和窗口查找方面的其他信息, 详见 [IfWinExist\(See 28.7\)](#)。

杂项函数

Asc(String**)**: 返回 **String** 里首个字符的 ASCII 码(一个 1 到 255 之间的数字)。如果 **String** 为空, 则返回 0。

Chr(Number**)**: 根据 **Number** 返回 ASCII 码里相应的单个字符。如果 **Number** 不是 1 和 255 及其之间的数, 将返回一个空字符串。常用的 ASCII 代码包括 9 (tab), 10 (换行), 13 (回车), 32 (空格), 48-57 (数字 0-9), 65-90 (大写字母 A-Z) 和 97-122 (小写字母 a-z)。

DllCall(): 请看 [DllCall\(\)\(See 18.3\)](#)。

IsLabel(LabelName**)**: 如果 **LabelName** 是脚本中[子程序\(See 17.8\)](#), [热键\(See 4.\)](#)或[热字符串\(See 5.\)](#)的标签名称(在 **LabelName** 中不要包括尾部的冒号), 则返回一个非零值。例如: 如果标签存在, 语句 *if IsLabel(VarContainingLabelName)* 将为 true, 反之就是 false。当你在 [Gosub\(See 17.8\)](#), [Hotkey\(See 20.1.10\)](#), [Menu\(See 22.13\)](#) 和 [Gui\(See 19.3\)](#) 这样的命令中指定了一个动态标签时, 这个函数对避免运行错误将十分有用。

ListView 和 **TreeView** 函数: 详见 [ListView\(See 19.7\)](#) 和 [TreeView\(See 19.8\)](#) 页面。

NumGet(VarOrAddress** [, **Offset** = 0, **Type** = "UInt"])** [v1.0.47+]: 返回储存在指定地址 + 偏移量中的二进制数。对于 **VarOrAddress**, 传递 MyVar 相当于传递 &MyVar。不过, 省略 "&" 可以执行地更好并确保目标地址是有效的(See 18.17) (无效的地址将返回 "")。相比之下, 除了变量, 传递给 **VarOrAddress** 的其他任何东西都被当作原始地址; 因此, 指定 MyVar+0 会强制用 MyVar 中的数字来代替 MyVar 本身的地址。对于 **Type**, 可以指定为 UInt, Int, Int64, Short, UShort, Char, UChar, Double 或者 Float(不过与 DllCall 不同, 当这些作为原义字符串使用时必须被括在引号里); 详见 [DllCall Types\(See 18.3\)](#)。

NumPut(Number**, **VarOrAddress** [, **Offset** = 0, **Type** = "UInt"])** [v1.0.47+]: 在指定地址 + 偏移量中以二进制的格式储存 **Number**, 并在刚刚写入的项目的右边返回地址。对于 **VarOrAddress**, 传递 MyVar 相当于传递 &MyVar。不过, 省略 "&" 可以执行地更好并确保目标地址是有效的(See 18.17) (无效的地址将返回 "")。相比之下, 除了变量, 传递给 **VarOrAddress** 的其他任何东西都被当作原始地址; 因此, 指定 MyVar+0 会强制用 MyVar 中的数字来代替 MyVar 本身的地址。对于 **Type**, 可以指定为 UInt, Int, Int64, Short, UShort, Char, UChar, Double 或者 Float(不过与 DllCall 不同, 当这些作为原义字符串使用时必须被括在引号里); 详见 [DllCall Types\(See 18.3\)](#)。如果一个整数太大而无法适应指定的 **Type**, 它最重要的字节就被忽略了; 比如 *NumPut(257, var, 0, "Char")* 将存储数字 1。

OnMessage(MsgNumber** [, "**FunctionName**"])**: 监控消息/事件。详见 [OnMessage\(\)\(See 18.11\)](#)。

RegisterCallback(): 请看 [RegisterCallback\(\)\(See 18.14\)](#)。

VarSetCapacity(UnquotedVarName** [, **RequestedCapacity**, **FillByte**])**: 扩大一个变量的容积或者释放它的内存。详见 [VarSetCapacity\(\)\(See 18.17\)](#)。

常用数学函数

注意：如果传入的任何参数是非数值的，那么数学函数通常会返回一个空值(空字符串)。

Abs(Number): 返回 *Number* 的绝对值。返回值的类型和 *Number* 一致(整数或浮点数)。

Ceil(Number): 返回 *Number* 向上取整数(没有任何的 .00 后缀)的值。例如，Ceil(1.2) 是 2，Ceil(-1.2) 是 -1。

Exp(N): 返回 *e* (大约为 2.71828182845905) 的 *N* 次幂。*N* 可以是负数也可以包含小数点。要抬升除了 *e* 以外的数到某个幂，请用 [** 运算符\(See 9.\)](#)。

Floor(Number): 返回 *Number* 向下取整数(没有任何的 .00 后缀)的值。比如 Floor(1.2) 是 1，Floor(-1.2) 是 -2。

Log(Number): 返回 *Number* 的对数(以 10 为底)。结果被格式化为[浮点数\(See 21.10\)](#)。如果 *Number* 是负数，将返回一个空字符串。

Ln(Number): 返回 *Number* 的自然对数(以 *e* 为底)。结果被格式化为[浮点数\(See 21.10\)](#)。如果 *Number* 是负数，将返回一个空字符串。

Mod(Dividend, Divisor): 求模。返回被除数 *Dividend* 除以除数 *Divisor* 时余下的值。结果的正负号和被除数一致。例如：mod(5, 3) 和 mod(5, -3) 都得出 2，但是 mod(-5, 3) 和 mod(-5, -3) 得出 -2。如果输入的任何一个浮点数，那么结果也会是浮点数。例如，mod(5.0, 3) 结果为 2.0, mod(5, 3.5) 结果为 1.5。如果除数为零，则函数得到空值(空字符串)。

Round(Number [, N]): 如果 *N* 为 0 或者省略 *N*，*Number* 将四舍五入为整数。如果 *N* 是正数，*Number* 取 *N* 个小数位。如果 *N* 是负数，在 *Number* 的小数点的左边取 *N* 位来四舍五入。例如：Round(345, -1) 结果为 350, Round(345, -2) 结果为 300。与 [Transform Round\(See 21.11\)](#) 不同，每当 *N* 小于 1 或者省略时，结果没有 .000 后缀。在 v1.0.44.01 及之后的版本中，一个大于零的 *N* 值会确切地显示 *N* 个小数位，而不会服从 [SetFormat\(See 21.10\)](#)。要避免这种情况的话，对 Round() 的返回值再执行另一个数学操作；例如：*Round(3.333, 1)+0*。

Sqrt(Number): 返回 *Number* 的平方根。结果被格式化为[浮点数\(See 21.10\)](#)。如果 *Number* 是负数，函数将得出空值(空字符串)。

三角函数

Sin(Number) | Cos(Number) | Tan(Number): 返回 *Number* 的正弦|余弦|正切三角函数值。*Number* 必须用弧度表示(详见下面)。

ASin(Number): 返回以弧度表示的反正弦(其正弦是 *Number*)。如果 *Number* 小于 -1 或者大于 1，则函数得到一个空值(空字符串)。

ACos(Number): 返回以弧度表示的反余弦(其余弦是 *Number*)。如果 *Number* 小于 -1 或者大于 1，则函数得到一个空值(空字符串)。

ATan(Number): 返回以弧度表示的反正切(其正切是 *Number*)。

注意: 要将弧度转换成角度, 将其乘以 $180/\pi$ (大约为 57.29578)。要将角度转换成弧度, 将其乘以 $\pi/180$ (大约为 0.01745329252)。 π (大约为 3.141592653589793) 的值是 1 的反正切乘以 4。

其他函数

[Titan](#) 的命令函数: 为每个有 OutputVar 的 AutoHotKey 命令提供一个可调用的函数。可以通过 [#Include\(See 17.1\)](#) 将这个库包含在任何一个脚本里。

翻译: hsudatalks 修正: 天堂之门 menk33@163.com 2008 年 11 月 10 日

11. AutoIt2 用户注意事项

- 享受针对 AutoIt v2 近乎完整的向后兼容性, 你没有必要学习新的语法和命令。
- 所有 [If 类型\(See 17.10\)](#) 的命令都可以使用 [Else\(See 17.5\)](#)。
- IF 和 ELSE 可以使用 [区块\(See 17.2\)](#) (多行的部分) (实际上消除 Goto 的需要)。
- 在 [循环\(See 17.12\)](#) 内使用 [break\(See 17.3\)](#) 和 [contiuue\(See 17.4\)](#) (另一种消除 Goto 的方式)。
- 使用许多新方法以及对于原有命令的改进 (如下)。
- 在你没还没有完全准备好使用它们的时候, 更早地 (载入时) 而不是在后面 (运行时) 捕捉低级错误和拼写错误: 捕捉语法错误作为 [优化\(See 30.13\)](#) 程序的一部分。
- 更好的启动 [性能\(See 30.13\)](#)。

AutoHotkey 可以不需要改变最令人兴奋的 .aut 脚本而运行它们。只有以下一些 AutoIt v2 功能不支持:

- Adlib 部分 (可以使用 [SetTimer\(See 17.21\)](#))
- Break ([现在在循环\(See 17.12\)](#) 中可以使用 [break\(See 17.3\)](#)。)
- HideAutoItDebug

尽管 .aut 脚本可以运行, 但为了完全使用 AutoHotkey 的新功能 (如果你想改进脚本), 你应该把脚本转换到 .ahk 文件。有时你必须转换脚本, 因为有些命令可以接受新的可选参数, 而这些参数如果用在 AutoIt v2 脚本上会导致其中一些命令运行不正常 (细节如下)。

你可以自动把 AutoIt v2 (.aut) 脚本转换成 AutoHotkey 格式 (.ahk)。转换将把 [escape 字符\(See 29.5\)](#) 转换一个重音符号 (`) 而不是反斜线符号 (\)。由于每行 escape 序列的排列的原因转换比听起来更难, 因此你不能仅仅在文本编辑器中运用搜索替换。自动转换功能应该可以处理所有这些细节:

- 在 .aut 文件名后添加 .ahk, 使得后缀名变为 ".aut.ahk"
- 用 AutoHotkey 运行这个文件, 与直接运行这个文件不同, 将在同一文件夹生成一个以 "-NEW.ahk" 结尾的文件。这是转换后的版本。原文件保持不变。

3. 如果你在脚本中用 [#Escapechar](#)(See 29.5), 请从转换后的文件中删除。

把 .aut 转换成 .ahk 后, 主要就是检查在 .ahk 中支持新参数而在 .aut 中不支持的命令。如果你在转换前用非 escape 逗号做最后一个参数 (在 AutoIt v2 中) 而现在不是最后一个参数的话, 这些逗号会被当做分界符。比如:

```
WinActivate, title, text with literal comma, here.
```

上面这句在转换后就有问题了。字符串“here.”将会被认为是新的“非标题的”参数。

底下是接受新参数、有可能出问题的命令列表:

[StringReplace](#)(See 27.20)

[StringGetPos](#)(See 27.14)

[FileSelectFile](#)

(See 16.23) [FileRemoveDir](#)(See 16.22)

[WinClose](#)(See 28.14)/[Kill](#)(See 28.23)/[Activate](#)(See 28.12)/[Minimize](#)(See 28.25)/[Maximize](#)(See 28.24)/[Restore](#)(See 28.28)/[Hide](#)(See 28.22)/[Show](#)(See 28.31)

[WinSetTitle](#)(See 28.30)/[WinGetTitle](#)(See 28.21)

注意: 即使有些命令在 .ahk 文件中接受新参数, 像 [IfWinExist](#)(See 28.7) 这样的命令也不会像上边的命令那样出问题因为程序可以区别 AutoIt v2 和 AutoHotkey 的方法。[WinWait](#)(See 28.32) 这一系列命令也不会出现问题, [MsgBox](#)(See 19.11) 命令同样也不会出问题因为它可以很好的处理逗号。

除了少数例外, 所有命令都支持浮点数 (比如 3.5) 和 16 进制 (比如 0xFF)。

所有 IF 类的命令都支持关键字 [ELSE](#)(See 17.5)。同时也支持 [区块](#)(See 17.2)。比如:

原来的风格:

```
IfWinExist, Notepad, , WinActivate, Notepad
```

新风格 (当有多行要执行时使用 {...} 标明区块):

```
IfWinExist, Untitled - Notepad
{
    WinActivate ; 不需要再次指明标题因为他会自动使用“最后找到的”窗口。
    WinMaximize ; 如果没这行, 这段就不需要区块 (括号)。
}
else
    Run Notepad
```

嵌套的 IF 和区块例子:

```

if varx = 1

{
    if vary = 2

        if varz = 3

            sleep, 1

        else ; 这个 else 和最近的 IF 相匹配

            sleep, 1

    }

else ; 这个 else 和顶端的 IF 匹配因为区块防止它与“if vary = y”匹配

sleep, 1

```

同样，区块也可以用于循环。但这第一个例子不需要区块因为 **if+else** 表达式只算一句：

```

Loop, 5000

IfNotExist, c:\semaphore.txt

sleep, 1000

else

break ; 提前终止循环

```

这个例子需要区块：

```

Loop ; 因为没有指明次数，除非遇到 Exit, Return 或是 Break，这是个无限循环。

{

    ...

    if var <= 5

        continue ; 跳过区段剩下的语句从区块开始下次循环。

    ...

    if var > 25

        break ; 跳出循环。

}

```

浏览文件夹内容的 [文件循环 \(file-loop\)\(See 16.31\)](#) 的例子 ([注册表循环 \(registry-loops\)\(See 17.16\)](#) 和 [分解循环 \(parsing-loops\)\(See 17.14\)](#) 也支持) :

```
Loop, %A_ProgramFiles%\*.txt, , 1 ; Recurse into subfolders.

{
    MsgBox, 4, , Filename = %A_LoopFileFullPath%`n`nContinue?

    IfMsgBox, No
        break
}
```

最后，我们也支持 [复杂的表达式\(See 9.\)](#) 和 [函数调用\(See 10.\)](#)，比如：

```
NetPrice := Price * (1 - Discount/100)

LargestOfTwo := Max (x, y)

if (CurrentSetting > 100 or FoundColor <> "Blue")
    MsgBox The setting is too high or the wrong color is present.
```

AutoHotkey.exe 运行 AutoIt v2 (.aut) 脚本时：

- 反斜杠 (\) 作为 escape 字符。但是其他脚本 (.ahk, .ini) 用重音符号 (`)。用 [#EscapeChar\(See 29.5\)](#) 更改默认值。
- 支持禁用同行注释 (用 [#AllowSameLineComments\(See 29.1\)](#) 启用它们)。
- 井号 (#) 可以在 [Send\(See 20.12\)](#) 命令中作为注释符号。其他脚本类型用它作为 windows 键。比如 #y 会发送 Win-Y。
- [DetectHiddenText\(See 28.4\)](#) 默认是 OFF。其他脚本类型默认是 ON 因为这样更实用。
- [SetBatchLines\(See 17.20\)](#) 默认是 1，意思是默认运行速度比较慢。其他脚本用 10ms。
- [SetWinDelay\(See 28.9\)](#) 默认是 500。其他脚本类型默认 100，这样可以让窗口操作更快。
- 以下命令的扩展 (新的) 参数被禁用了。这是出于兼容性的考虑 (看上边的转换注意事项)：
 - [MsgBox\(See 19.11\)](#)
 - [FileSelectFile\(See 16.23\)](#)
 - [FileRemoveDir\(See 16.22\)](#)
 - [IniRead\(See 16.29\)](#)
 - [StringReplace\(See 27.20\)](#)

[StringGetPos\(See 27.14\)](#)
[WinClose\(See 28.14\)](#)
[WinKill\(See 28.23\)](#)
[WinActivate\(See 28.12\)](#)
[WinMinimize\(See 28.25\)](#)
[WinMaximize\(See 28.24\)](#)
[WinRestore\(See 28.28\)](#)
[WinHide\(See 28.22\)](#)
[WinShow\(See 28.31\)](#)
[WinSetTitle\(See 28.30\)](#)
[WinGetTitle\(See 28.21\)](#)

脚本编译器 ([Ahk2Exe\(See 8.\)](#)) 不支持 AutoIt v2 (.aut) 脚本，因为编译后的脚本无法判断以 AutoIt v2 模式或是 AutoHotkey 模式运行。所以先把 .aut 脚本转换成 .aut 然后在编译它 (看上边)。

与 AutoIt v2 不同，[IfGreater/Less\(See 17.10\)](#) 如果一个输入不是纯数字的话将比较字母。

[IniWrite\(See 16.30\)](#) 和 [IniDelete\(See 16.28\)](#): 对于 AutoIt v2 (.aut) 脚本：这两个命令没有设置 [ErrorLevel\(See 30.8\)](#)。对于其他脚本来说，如果有问题，ErrorLevel 是 1，否则是 0。

[InputBox\(See 19.10\)](#): 对于 AutoIt v2 (.aut) 脚本：如果用户按了取消键，输出变量会被设成空，这样可以知道是按了取消键。对于 AutoHotkey 脚本：即使用户按了取消键，输出变量也会是用户输入的值。为了区别，如果按下取消, Alt-F4, 或是 X-button 关闭窗口，[ErrorLevel\(See 30.8\)](#) 会被设成 1。否则 ErrorLevel 是 0。这样取消键可以设置对于不同输入文本的特殊操作。

[SplashTextOn\(See 19.14\)](#): AutoIt v2 (.aut) 脚本用的 splash 窗口比 AutoHotkey 脚本中的矮些。所以如果你要转换一个 .aut 脚本到 .ahk，或是从一个 .aut 脚本拷贝粘贴代码到一个 .auk 脚本，你可能会少掉它的标题行那么高的高度 (大概 20 像素，但是具体要看你的桌面设置)。

最后和 AutoIt v2 不一样，AutoHotkey 没有把它的 [变量\(See 9.\)](#) 存为环境变量。因为这样性能会降低而且操作系统对于这样的变量有 32KByte 的限制。要明确地将一个值存进环境变量中，用 [EnvSet\(See 15.3\)](#)，而不是 [SetEnv\(See 22.19\)](#).

同行注释可以在脚本扩展名不是 .aut (AutoIt v2) 的时候使用。分号可以用做这些注释的前缀字符，就跟用在整行注释的时候一样。但是分号只有在其左边至少有一个空格或是 tab 的时候才被当做注释的一部分。如果需要在空格或是 tab 后使用文字的 (*literal*) 分号，可以在它前面加上重音符号 (`)。在底下的两个例子中，变量都被符为文字的分号：

```
var = `;  
var =
```

为了方便起见，所有命令的第一个逗号可以省略 (除非第一个参数为空)。比如: Sleep 1, MsgBox test, etc.

一些命令可以选用新语法替代 AutoIt v2 的旧语法。这些命令是：

- Var = value ; 可以替代 SetEnv, var, value。Value 可以是空，比如 var =

- `if var <> value` ; 你可以用 `<> != <= >` 作为比较运算符替代 `ifequal/notequal/greater`, 等等。
- `var += 5` ; 这个等同于 `envadd, var, 5`。其他运算符有 `-=, *=, /=`
- 但是新旧命令语法不要在同行使用:
 - `IfEqual, x, 35, y = 55` ; 程序将无法识别 `y = 55` 作为 IF 的动作。
 - ; 应该是:
 - `IfEqual, x, 35, SetEnv, y, 55`
 - ; 或是:
 - `IfEqual, x, 35`

`y = 55`

你也可以确定一个变量是否 符合(See 27.4) 列表中的项目:

`if var [not] in item1,item2,item3` ; 完全匹配
`if var [not] contains item1,item2,item3` ; 包含

...或者一个变量是否在两个值 之间(See 21.7):

`if var [not] between 5 and 10`

...或者一个变量是否是 特定的数据类型(See 21.8) :

`if var is number`

`if var is not float` ; 不是浮点类型

`SetTitleMatchMode`(See 28.8) 命令得到了改进: 除了支持传统模式 1 和 2, 现在还支持单词 `fast` 和 `slow` (比如: `SetTitleMatchMode, slow`)。默认是 AutoIt v2 使用的 `fast`。但是, 对于某些窗口来说 `slow` 模式可以“看见”更多文本。`Fast` 模式速度更快, 可以提高调用许多窗口命令的脚本的运行速度。`Slow` 模式只有当没有其他办法识别某个窗口的时候才使用。程序所带的 `Window Spy` 只有在 `slow` 模式下才能显示窗口中哪些文字是可以使用的。

对于大多数窗口命令, 比如 `WinActivate` 不传递任何参数会激活被 `IfWinExist` 或 `WinWait` 最近发现的(See 30.2) 窗口。向 `WinActivate` 传递字母 A 作为窗口标题的值会使命令作用于当前激活的(前台)窗口。比如: `WinClose, A`。

同样, 大部分窗口命令支持 `Exclude-Title` 和 `Exclude-Text` 作为新参数。如果一个窗口的标题包含了 `Exclude-Title` 或是文本包含了 `Exclude-Text`, 那么这个窗口将不合格。

`StringGetPos`(See 27.14) 现在最后一个参数是一个新的可选参数。如果这个参数是字母 R, 搜索会从右边而不是左边开始, 并将返回最后一个符合条件的结果而不是第一个。

`MouseClick`(See 23.3) 支持一个新的参数。如果是字母 D, 将会一直按下按某个键直到用户手动点击它或是运行脚本中的下一个动作。如果最后一个参数是 U, 按键将会被释放(即使以前没有按它, 也会发送一个释放事件 (up-event))。除此之外, 你也会发现新的 `Click 命令`(See 23.1) 语法更简单灵活。

`A_Space` 和 `A_Tab` 是两个新的内置参数，分别代表一个空格和制表符。这样避免了需要用其他方法获取一个含有空格的变量。比如：

```
String = String with spaces

IfInString, String, %A_SPACE%

    MsgBox, A space was found.

; 但是要赋给一个变量单独一个空格，记得要关掉 AutoTrim，这样

; 赋值不会删除开头和结尾的空格：

AutoTrim, off

MySpace = %A_SPACE%
```

[ClipboardAll](#)(See 30.6) 是一个新的内置变量，它包含所有剪贴板上的内容（比如图片和格式）。它可以用在内存和硬盘中备份一个或多个“已存的剪贴板”，这些备份可以在以后恢复。这个命令尤其在脚本暂时用到剪贴板的时候有用，你不用害怕不知道用户都在上面存了什么。

[Run](#)(See 24.5) 和 [RunWait](#)(See 24.5) 可以运行快捷方式 (.lnk files)、文档，和网址。比如：

```
Run, www.yahoo.com

RunWait, New Document.doc
```

除此之外，也支持一些系统动作： properties、edit、print、find、explore、open 和 print。比如：

```
Run, properties c:\autoexec.bat ; 打开这个文件的属性对话框。

Run, edit %A_SCRIPTFULLPATH% ; 执行与这个文件相关联的“edit”动作（如果它有的话）。
```

[Send](#)(See 20.12) 和 [ControlSend](#)(See 20.6) 功能也得到了改进。你不需要用 `Sleep` 命令让用户有时间释放用 `Send` 命令发送的修饰键 (Ctrl/Alt/Shift/Win) 和热键。对于每个键，`Send` 命令知道应该把修饰键变成什么发送出去。

[StringReplace](#)(See 27.20) 可以替换所有搜索到的结果，只要把最后的参数设为“`A`”。

[StringGetPos](#)(See 27.14) 现在在命令最后支持新的参数。如果这个参数是字母 `R`，搜索将从右边开始而不是左边。也就是说将返回最后一个出现的结果而不是第一个。

[MsgBox](#)(See 19.11) 在最后支持一个新的可选参数：`Timeout`（按秒计算，经过多少秒 `MsgBox` 会关闭。[If MsgBox](#)(See 19.9) 在这种情况下可以看到 `TIMEOUT` 的值）。出于兼容性的考虑，只有当脚本名扩展名不是 `.aut` 的时候这个新的参数才可以用。

[IniRead](#)(See 16.29) 在最后支持一个新的可选参数：如果命令失败，默认赋予输出变量的值。出于兼容性的考虑，只有当脚本名扩展名不是 `.aut` 的时候这个新的参数才可以用。

[FileSelectFile](#)(See 16.23) 加入了很多新功能。

下面介绍一些最值得注意的新命令和功能：

[GUI](#)(See 19.3): 生成并管理图形用户界面 (GUI) 窗口和控件。生成你自己有专业外观的应用程序和表格而不用考虑复杂的编程语言。

[Menu](#)(See 22.13): 生成你自己定制的菜单栏，系统托盘图标菜单，和弹出菜单。

[Hotkeys](#)(See 4.): 用于键盘，游戏杆和鼠标。

(See 26.5)[Hotstrings](#)(See 5.): 当你键入它们的时候，这些缩写会自动扩展开 (自动替换)。

[DII Call](#)(See 18.3): 调用 DLL 中的函数，比如标准 Windows API 函数。

[OnMessage](#)(See 18.11): 定义当脚本收到某个特定消息时 (从操作系统、外部程序或是其他脚本) 自动运行的 [函数](#)(See 10.)。

[SendInput](#)(See 20.12) 和 [SendPlay](#)(See 20.12): 这两个新方法用于快速可靠地发送键击和鼠标手势的。而且，SendPlay 比起其他命令可以向更多种类的游戏发送键击和鼠标手势。

[Progress](#)(See 19.12) 和 [SplashImage](#)(See 19.12): 显示进度条和图像，以及可选文字。

[OnExit](#)(See 17.17): 可以定义当脚本退出时自动运行的程序。脚本也可以用它检测是否用户注销或是关机。

[Process](#)(See 24.4): 对一个进程执行如下之一的操作：检查是否存在；更改优先级；关掉它；等待它关闭。

[SoundSet](#)(See 26.5): 更改一个音频设备的设置 (主设备静音，主设备音量等等。)

[SetTimer](#)(See 17.21): 在自定义时间间隔内执行一个或多个程序。

[ClipWait](#)(See 15.1): 等待剪贴板包含文字 (变成非空)。

[StatusBarWait](#)(See 28.11): 等待一个窗口的状态栏包含特定文字或是变成空。

[StatusBarGetText](#)(See 28.10): 获取状态栏的文字。

[WinActivateBottom](#)(See 28.13): 这个命令与 WinActivate 相似但是是激活最老的 (最底层的) 窗口而不是最新的。如果只有一个窗口，结果将和 WinActivate 一样。

[DriveSpaceFree](#)(See 16.3), [DriveGet](#)(See 16.2) 和 [Drive](#)(See 16.1): 获取并改变一个分区的信息 (比如它的可用空间)。

[AutoTrim](#)(See 22.3), `on/off` (默认 `on`) : 当你给一个变量赋值时，它控制是否自动删掉字符串左右的空格。由于这个原因，空格是指空格和制表符而换行符 (`newline` 或是 `linefeed`) 则不包括在内。因此我相信“`on`”和 `Autoit v2` 中的效果是一样的。

[StringUpper](#)(See 27.18) 和 [StringLower](#)(See 27.18): 将一个字符串转换成大写或是小写。

[ControlFocus](#)(See 28.1.3), [ControlSetText](#)(See 28.1.10), [Control](#)(See 28.1.1), [ControlGet](#)(See 28.1.4): 和其他 Control 命令直接控制一个窗口的控件。

[SoundPlay](#)(See 26.4): 播放任何操作系统支持的媒体文件。

[SoundGet](#)(See 26.2), [SoundSet](#)(See 26.5)，和 [SoundSetWaveVolume](#)(See 26.6): 获取并改变一个声音设备的设置 (主设备静音，主设备音量，波形音量等。)

[PixelGetColor](#)(See 22.15) 和 [PixelSearch](#): (See 22.16)这两个命令可以获得屏幕上某点的颜色，可以用于得知一个应用程序或是游戏的状态是否改变了。

[GetKeyState\(See 20.7\)](#): 检测一个键，鼠标或是游戏杆的键是否按下或放开；检测游戏杆的位置；等。

[WinMenuItemSelectItem\(See 28.1.14\)](#): 不管目标窗口是否被激活，直接运行菜单栏项目。

[#AllowSameLineComments\(See 29.1\)](#): 为了提高和 AutoIt v2 的兼容性，以.aut 结尾的脚本通常不允许有同行注释（比如：Run, notepad ; this is a comment）。要允许同行注释可以在你的脚本第一行加上 #AllowSameLineComments

[#SingleInstance\(See 22.2\)](#): 在脚本的任何地方加上这行可以避免运行两个脚本实例。同时，你可以选择是保留原有的实例还是用新的替换掉原有的。

[#EscapeChar\(See 29.5\)](#): escape 字符通常是“`”，但你也可以用 AutoIt v2 中的 escape 字符 (\) 或是其他你选择的字符。注意 .aut 结尾的脚本默认使用\作为 escape 字符的。

[Suspend\(See 17.23\)](#) (和托盘菜单具有相同名字的选项) : 这个函数阻止新的热键运行（它无法停止已经运行的热键功能--要停止的话可以用 Pause）。

[Pause\(See 17.18\)](#) (和托盘菜单具有相同名字的选项) : 和 Suspend 不同--它完全停止了热键 --pause 会暂停正在运行的程序（如果没有， pause 不会有任何影响）。

文件和文件夹循环([See 16.31](#))、[注册表循环\(See 17.16\)](#)、以及功能更强的 循环([See 17.12](#))。

加入了 窗口组([See 28.2.2](#)) 的概念。一旦定义了一个组，这个组就可以用于 [GroupActivate\(See 28.2.1\)](#) 命令。

最后，还有其他许多命令。你可以从 [按字母排序的命令列表\(See 12.\)](#) 中查看。

12. 按字母顺序排列的命令列表

命令	说明
{ ... }(See 17.2)	一对大括号表示一个区块。区块通常和 函数(See 10.) 、 Else(See 17.5) 、 Loop(See 17.12) 和 IF 命令一起使用。
AutoTrim(See 22.3)	设置在像 "Var1 = %Var2%"(See 22.19)，给 Var1 赋值的时候是否自动省略 Var2 中首尾的空格和 Tab。
BlockInput(See 20.5)	开启或关闭用户通过鼠标和键盘与计算机交互的能力。
Break(See 17.3)	退出(终止)某个 loop(See 17.12) 。仅在 loop(See 17.12) 中有效。
Click(See 23.1)	在指定的坐标处模拟鼠标点击，同样也可以模拟按住鼠标、滚动滚轮、鼠标移动。
ClipWait(See 15.1)	等到 剪贴板(See 30.6) 包含数据。
Continue(See 17.4)	跳过当前 loop(See 17.12) 重复的剩余部分并开始一个新的重复。仅在 loop(See 17.12) 内部才有效。
Control(See 28.1.1)	Makes a variety of changes to a control.

ControlClick (See 23.2)	Sends a mouse button or mouse wheel event to a control.
ControlFocus (See 28.1.3)	在一个窗口特定的控件上设置输入焦点。
ControlGet (See 28.1.4)	Retrieves various types of information about a control.
ControlGetFocus (See 28.1.5)	检索目标窗口的哪个控件有输入焦点，若有的话。
ControlGetPos (See 28.1.6)	取得一个控件的位置和大小。
ControlGetText (See 28.1.7)	从一个控件获取文本。
ControlMove (See 28.1.8)	移动或调整一个控件的大小。
ControlSend / ControlSendRaw (See 20.6)	Sends simulated keystrokes to a window or control.
ControlSetText (See 28.1.10)	Changes the text of a control.
CoordMode (See 22.6)	为一些命令设置相对于激活窗口或屏幕的坐标模式。
Critical (See 22.7)	防止 当前线程 (See 30.19)被其他线程中断。
DetectHiddenText (See 28.4)	决定窗口中隐藏的文本是否在探测窗口时“可见”。这会影响像 IfWinExist 和 WinActivate 这样的命令。
DetectHiddenWindows (See 28.5)	决定不可见的窗口是否被脚本“看见”。
DIICall() (See 18.3)	调用一个 DLL 文件中的函数，例如一个标准的 Windows API 函数。
Drive (See 16.1)	弹出/缩进 CD 或 DVD 驱动器的托盘，或者设定一个驱动器的卷标。
DriveGet (See 16.2)	获得关于计算机驱动器各种类型的信息。
DriveSpaceFree (See 16.3)	获得一个驱动器的剩余磁盘空间信息，以兆字节表示。
Edit (See 22.9)	使用相关联的编辑器打开当前脚本进行编辑。
Else (See 17.5)	假如一个 IF-语句得出 FALSE，则执行指定的命令。当出现多个命令时，将它们括入一个 区块 (See 17.2)(大括号)。
EnvAdd (See 21.1)	将 变量 (See 9.)设置为它自身加上给定的值的总和（也能从一个 日期-时间 (See 16.26)的值里加上或减去时间）。同义于：var += value
EnvDiv (See 21.2)	将 变量 (See 9.)设置为它自身除以给定的值。同义于：var /= value
EnvGet (See 15.2)	获取一个环境变量。
EnvMult (See 21.3)	将 变量 (See 9.)设置为它自身乘以给定的值。同义于：var *= value
EnvSet (See 15.3)	向包含在系统环境中的 变量 (See 9.)写入值。
EnvSub (See 21.4)	将 变量 (See 9.)设置为它自身减去给定的值（也能比较 日期-时间 (See 16.26)的值）。同义于：var -= value

EnvUpdate(See 15.4)	通知操作系统和所有正在运行的程序环境变量(See 9.)已改变。
Exit(See 17.6)	退出当前线程(See 30.19)或(如果脚本非持续的(See 29.21)且未包含热键)整个脚本。
ExitApp(See 17.7)	无条件地终止脚本。
FileAppend(See 16.4)	在文件的结尾处追加文本(如果需要, 首先创建文件)。
FileCopy(See 16.5)	复制一个或多个文件。
FileCopyDir(See 16.6)	复制一个文件夹连同它的所有文件和子文件夹(和 xcopy 相似)。
FileCreateDir(See 16.7)	创建一个文件夹。
FileCreateShortcut(See 16.8)	创建一个快捷方式 (.lnk) 文件。
FileDelete(See 16.9)	删除一个或多个文件。
FileInstall(See 16.15)	装入指定文件到已编译形式(See 8.)的脚本中。
FileGetAttrib(See 16.10)	报告一个文件或文件夹是否为只读、隐藏等等。
FileGetShortcut(See 16.11)	获取某个快捷方式(.lnk)文件的信息, 比如它的目标文件。
FileGetSize(See 16.12)	获取一个文件的大小。
FileGetTime(See 16.13)	获取一个文件或文件夹的日期时间戳信息。
FileGetVersion(See 16.14)	获取一个文件的版本。
FileMove(See 16.16)	移动或重命名一个或者多个文件。
FileMoveDir(See 16.17)	移动一个文件夹连同它的所有文件和子文件夹。也能重命名一个文件夹。
FileRead(See 16.19)	读取一个文件的文本给一个变量(See 9.)。
FileReadLine(See 16.18)	读取文件的指定行, 并将所得文本存储于变量(See 9.)中。
FileRecycle(See 16.20)	发送一个文件或目录到回收站, 如果可行的话。
FileRecycleEmpty(See 16.21)	清空回收站。
FileRemoveDir(See 16.22)	删除一个文件夹。
FileSelectFile(See 16.23)	显示一个允许用户来打开或保存文件的标准对话框。
FileSelectFolder(See 16.24)	显示一个标准对话框从而允许用户选择文件夹。
FileSetAttrib(See 16.25)	更改一个或更多文件或者文件夹的属性。支持通配符。

FileSetTime (See 16.26)	Changes the datetime stamp of one or more files or folders. Wildcards are supported.
FormatTime (See 27.1)	Transforms a YYYYMMDDHH24MISS (See 16.26) timestamp into the specified date/time format.
GetKeyState (See 20.7)	Checks if a keyboard key or mouse/joystick button is down or up. Also retrieves joystick status.
Gosub (See 17.8)	跳到指定的标签并且继续执行，直到碰到 Return (See 17.19)。
Goto (See 17.9)	跳转到指定的标签并继续执行。
GroupActivate (See 28.2.1)	Activates the next window in a window group that was defined with GroupAdd (See 28.2.2).
GroupAdd (See 28.2.2)	Adds a window specification to a window group, creating the group if necessary.
GroupClose (See 28.2.3)	Closes the active window if it was just activated by GroupActivate (See 28.2.1) or GroupDeactivate (See 28.2.4). It then activates the next window in the series. It can also close all windows in a group.
GroupDeactivate (See 28.2.4)	Similar to GroupActivate (See 28.2.1) except activates the next window not in the group.
GUI (See 19.3)	创建并管理窗口和控件。这样的窗口可以被用作数据输入表格或者自定义用户界面。
GuiControl (See 19.5)	Makes a variety of changes to a control in a GUI window.
GuiControlGet (See 19.6)	Retrieves various types of information about a control in a GUI window.
HideAutoItWin , On Off	[Obsolete -- the following is equivalent: Menu (See 22.13), tray , NoIcon Icon]
Hotkey (See 20.1.10)	Creates, modifies, enables, or disables a hotkey while the script is running.
if (See 17.10)	指定一个 变量 (See 9.)和一个值比较得出 TRUE 时要执行的命令。当存在多个命令时，将它们括入一个 区块 (See 17.2)。
if (expression) (See 17.11)	指定 表达式 (See 9.)得出 TRUE 时要执行的命令。
If var [not] between (See 21.7)	Checks whether a variable's (See 9.) contents are numerically or alphabetically between two values (inclusive).

If var [not] in/contains MatchList (See 27.4)	Checks whether a variable's(See 9.) contents match one of the items in a list.
If var is [not] type(See 21.8)	检查一个变量(See 9.)内容是否为数值、大写字母或其他。
IfEqual/IfNotEqual(See 17.10)	比较一个变量(See 9.)和一个值是否相等。与此同义: if var = value if var <> value
IfExist(See 16.27) / FileExist()(See 10.)	检查文件或文件夹的存在。
IfGreater/IfGreaterOrEqual(See 17.10)	比较一个变量(See 9.)和一个值是否相等。与此同义: if var > value if var >= value
IfInString(See 27.3) / InStr()(See 10.)	检查一个变量(See 9.)是否包含指定的字符串。
IfLess/IfLessOrEqual(See 17.10)	比较一个变量(See 9.)和一个值是否相等。与此同义: if var < value if var <= value
IfMsgBox(See 19.9)	检测用户在最近的 MsgBox(See 19.11) 命令点击了哪个按钮。
IfWinActive / IfWinNotActive(See 28.6)	检查指定的窗口是否存在和当前是否激活(在最前面)。
IfWinExist / IfWinNotExist(See 28.7)	检查指定的窗口是否存在。
ImageSearch(See 22.10)	在屏幕某一区域中搜索图像。
IniDelete(See 16.28)	从标准格式的 .ini 文件中删除键值。
IniRead(See 16.29)	从一个标准格式的 .ini 文件读取一个键值。
IniWrite(See 16.30)	向一个标准格式的 .ini 文件中写入一个键值。
Input(See 20.11)	Waits for the user to type a string (not supported on Windows 9x: it does nothing).

InputBox (See 19.10)	Displays an input box to ask the user to enter a string.
KeyHistory (See 20.9)	列出脚本信息以及最近的按键和鼠标点击记录。
KeyWait (See 20.10)	等待一个按键或鼠标/操纵杆按钮被松开或者按下。
LeftClick	[Obsolete -- use Click (See 23.1) for greater flexibility]
LeftClickDrag	[Obsolete -- use MouseClickDrag (See 23.4) for greater flexibility]
ListHotkeys (See 20.1.11)	显示当前脚本使用的热键，不论它们的子程序当前是否运行，也不论它们是否使用 键盘 (See 20.2)或者 鼠标 (See 20.3)钩子。
ListLines (See 22.11)	显示最近已运行过的各脚本行。
ListVars (See 22.12)	列出脚本中的 变量 (See 9.): 它们的名称和当前的内容。
Loop (标 准 的) (See 17.12)	重复地执行一系列命令：可以是制定了数字的重复次数或直到遇上了 break (See 17.3)。
Loop (文件和文件夹)(See 16.31)	获取指定的文件或文件夹，一次一个。
Loop (解析字符串) (See 17.14)	从一个字符串中获取子字符串(片段)，一次获取一段。
Loop (读 取 文 件 内 容)(See 16.32)	逐行读取一个文本文件的内容 (比 FileReadLine (See 16.18) 执行地更好)。
Loop (registry)(See 17.16)	Retrieves the contents of the specified registry subkey, one item at a time.
Menu (See 22.13)	创建、删除、修改以及显示菜单和菜单项。更改托盘图标和它的提示。控制是否允许打开 已编译脚本 (See 8.)的主窗口。
MouseClick (See 23.3)	点击或按住一个鼠标按键，或者滚动鼠标滚轮。注意：一般来说， Click 命令(See 23.1)更加灵活和易用。
MouseClickDrag (See 23.4)	点击并按住指定的鼠标按键，移动鼠标到目标位置，释放鼠标按键。
MouseGetPos (See 23.5)	返回鼠标的当前位置，以及鼠标当前悬停的窗口和控件(可选)。
MouseMove (See 23.6)	移动鼠标指针。
MsgBox (See 19.11)	在一个包含一个或多个按钮(比如 Yes 和 No)的小窗口中显示指定的文本。
OnExit (See 17.17)	Specifies a subroutine (See 17.8) to run automatically when the script exits.

OnMessage() (See 18.11)	Specifies a function (See 10.) to call automatically when the script receives the specified message.
OutputDebug (See 22.14)	发送字符串到调试器(如果有的话)以显示。
Pause (See 17.18)	暂停脚本的 当前线程 (See 30.19)。
PixelGetColor (See 22.15)	Retrieves the color of the pixel at the specified x,y screen coordinates.
PixelSearch (See 22.16)	Searches a region of the screen for a pixel of the specified color.
PostMessage (See 28.1.12)	Places a message in the message queue of a window or control.
Process (See 24.4)	Performs one of the following operations on a process: checks if it exists; changes its priority; closes it; waits for it to close.
Progress (See 19.12)	Creates or updates a window containing a progress bar.
Random (See 21.9)	Generates a pseudo-random number.
RegExMatch() (See 18.12)	Determines whether a string contains a pattern (regular expression).
RegExReplace() (See 18.13)	Replaces occurrences of a pattern (regular expression) inside a string.
RegDelete (See 25.2)	从注册表中删除一个子键或一个值。
RegRead (See 25.3)	从注册表中读取一个值。
RegWrite (See 25.4)	写入一个值到注册表中。
RegisterCallback() (See 18.14)	Creates a machine-code address that when called, redirects the call to a function (See 10.) in the script.
Reload (See 20.1.13)	用一个新的脚本实例来替换当前运行的实例。
Repeat...EndRepeat	[Obsolete -- use Loop (See 17.12) for greater flexibility]
Return (See 17.19)	从一个子程序返回至先前通过 函数调用 (See 10.)、 Gosub (See 17.8)、 热键 (See 4.)激活、 GroupActivate (See 28.2.1)或者其他方法执行的跳转。
RightClick	[Obsolete -- use Click (See 23.1) for greater flexibility]
RightClickDrag	[Obsolete -- use MouseClickDrag (See 23.4) for greater flexibility]
Run (See 24.5)	运行一外部程序。
RunAs (See 24.6)	为所有随后要使用的 Run 和 RunWait 指定一组用户凭证。需要 Windows 2000/XP 或之后版本。

RunWait(See 24.5)	运行一个外部程序并等到它结束。
Send(See 20.12) / SendRaw(See 20.12) / SendInput(See 20.12) / SendPlay(See 20.12)	发送模拟的键击和鼠标点击到 激活的(See 28.12) 窗口。
SendMessage(See 28.1.12)	Sends a message to a window or control and waits for acknowledgement.
SendMode(See 20.13)	让 Send(See 20.12) 命令和 SendInput 或 SendPlay 具有同样的功能而不是其默认的(SendEvent)。也让 Click 和 MouseMove/Click/Drag 使用了指定的方法。
SetBatchLines(See 17.20)	决定脚本的执行速度(影响cpu占用)。
SetCapslockState(See 20.15)	设置 Capslock/NumLock/ScrollLock 的状态。也可以让这些键保持开启或关闭的状态。
SetControlDelay(See 28.1.13)	设置两次控件操作类命令之间的延时。
SetDefaultMouseSpeed(See 23.7)	设置鼠标速度，如果在 Click(See 23.1) 和 MouseMove(See 23.6) / Click(See 23.3) / Drag(See 23.4) 中未指定的话。
SetFormat(See 21.10)	设置数学运算得到的整数或浮点数的格式。
SetKeyDelay(See 20.14)	设置 Send(See 20.12) 或 ControlSend(See 20.6) 命令发送的两次按键事件间的延时。
SetMouseDelay(See 23.8)	设置每次鼠标移动或点击后会发生的延时。
SetNumlockState(See 20.15)	设置 Numlock 键的状态。也能强制此键保持 on 或 off。
SetScrollLockState(See 20.15)	设置 Scrolllock 键的状态。也能强制此键保持 on 或 off。
SetStoreCapslockMode(See 20.16)	设置在一个 Send(See 20.12) 命令之后是否恢复 CapsLock 的状态。
SetTimer(See 17.21)	以一个指定的时间间隔来自动重复地启动子程序。
SetTitleMatchMode(See 28.8)	在像 WinWait(See 28.32) 这样的命令中设置 WinTitle 参数的匹配模式。
SetWinDelay(See 28.9)	设置每个窗口命令后会发生的延时，例如 WinActivate(See 28.12) 。

SetWorkingDir (See 16.33)	更改脚本的当前工作目录。
Shutdown (See 24.7)	关机、重启或注销操作系统。
Sleep (See 17.22)	在继续前等待指定的时间量。
Sort (See 27.12)	Arranges a variable's contents in alphabetical, numerical, or random order (optionally removing duplicates).
SoundBeep (See 26.1)	从个人计算机的扬声器发出一个音调。
SoundGet (See 26.2)	Retrieves various settings from a sound device (master mute, master volume, etc.)
SoundGetWaveVolume (See 26.3)	获取一个音频设备的波形输出音量。
SoundPlay (See 26.4)	播放一个音频、视频或者其他支持的文件类型。
SoundSet (See 26.5)	Changes various settings of a sound device (master mute, master volume, etc.)
SoundSetWaveVolume (See 26.6)	更改一个音频设备的波形输出音量。
SplashImage (See 19.12)	Creates or updates a window containing a JPG, GIF, or BMP image.
SplashTextOn (See 19.14)	创建可定制的文本弹出窗口。
SplashTextOff (See 19.14)	关闭上一行所说的窗口。
SplitPath (See 16.34)	将一个文件名或 URL(统一资源定位符)分解成它的名称、目录、扩展名和驱动器。
StatusBarGetText (See 28.10)	Retrieves the text from a standard status bar control.
StatusBarWait (See 28.11)	Waits until a window's status bar contains the specified string.
StringCaseSense (See 27.13)	决定字符串比较是否区分大小写(默认是“不区分大小写”)。
StringGetPos (See 27.14) / InStr() (See 10.)	返回指定的子字符串在一个字符串中的位置。
StringLeft (See 27.15)	从一个字符串的左边提取一定数量的字符。
StringLen (See 27.17) / StrLen() (See 10.)	获取一个字符串中字符的数量。
StringLower (See 27.18)	将一个字符串中的英文字母转换为小写。

StringMid(See 27.19) / SubStr()(See 10.)	从字符串中指定的位置返回一个或多个字符。
StringReplace(See 27.20)	替换字符串中指定的子字符串为新的字符串。
StringRight(See 27.15)	从一个字符串的右边开始提取一定数量的字符。
StringSplit(See 27.21)	使用指定的分隔符将一个字符串分割为一个字符串数组。
StringTrimLeft(See 27.22)	从一个字符串的左边移除指定数量的字符。
StringTrimRight(See 27.22)	从一个字符串的右边移除指定数量的字符。
StringUpper(See 27.18)	将一个字符串中的英文字母转换为大写。
Suspend(See 17.23)	禁用或启用所有的或是选择的热键(See 4.)。
SysGet(See 22.21)	Retrieves screen resolution, multi-monitor info, dimensions of system objects, and other system properties.
Thread(See 22.22)	Sets the priority or interruptibility of threads(See 30.19). It can also temporarily disable all timers(See 17.21).
ToolTip(See 19.15)	创建一个在屏幕任何位置上总在最前端的窗口。
Transform(See 21.11)	Performs miscellaneous math functions, bitwise operations, and tasks such as ASCII/Unicode conversion.
TrayTip(See 19.16)	在托盘图标处创建一个气泡消息窗口。要求 Windows 2000/XP 或以上操作系统。
UrlDownloadToFile(See 22.24)	从 Internet(国际互联网) 下载一个文件。
Var = value (See 22.19)	为一个指定的变量(See 9.)赋值。
Var := expression(See 21.12)	计算一个表达式(See 9.)的值并将结果存储在一个变量(See 9.)中。
VarSetCapacity()(See 18.17)	Enlarges a variable's holding capacity or frees its memory. Normally, this is necessary only for unusual circumstances such as DllCall(See 18.3).

WinActivate(See 28.12)	激活指定的窗口(将它置于最前端)。
WinActivateBottom(See 28.13)	功能和 WinActivate(See 28.12) 一样，只是这个是激活最底端的(至少最近激活的)窗口，而不是最顶端的。
WinClose(See 28.14)	关闭指定的窗口。
WinGetActiveStats(See 28.16)	合并 WinGetActiveTitle(See 28.17) 和 WinGetPos(See 28.19) 的功能到一个命令。
WinGetActiveTitle(See 28.17)	获取激活窗口的标题。
WinGetClass(See 28.18)	获取指定窗口的类名。
WinGet(See 28.15)	返回符合指定条件的窗口的 uID，进程 ID，进程名称，或控件列表。它也可以返回一个列表，包含所有符合指定条件的窗口。
WinGetPos(See 28.19)	返回指定窗口的位置和大小。
WinGetText(See 28.20)	返回指定窗口中的文本。
WinGetTitle(See 28.21)	返回指定窗口的标题。
WinHide(See 28.22)	隐藏指定的窗口。
WinKill(See 28.23)	强制关闭指定的窗口。
WinMaximize(See 28.24)	使指定窗口扩展到它最大的尺寸。
WinMenuItem(See 28.1.14)	从指定窗口的菜单栏调用一个菜单项。
WinMinimize(See 28.25)	收缩指定的窗口到任务栏上的一个按钮。
WinMinimizeAll(See 28.26)	最小化所有窗口。
WinMinimizeAllUndo(See 28.26)	前一个 WinMinimizeAll(See 28.26) 的反效果。
WinMove(See 28.27)	改变指定窗口的位置和/或大小。
WinRestore(See 28.28)	如果指定窗口最小化或最大化，还原它们。
WinSet(See 28.29)	对指定窗口进行一系列改变，例如“置顶”和透明。

WinSetTitle (See 28.30)	改变指定窗口的标题。
WinShow (See 28.31)	显示指定的窗口。
WinWait (See 28.32)	等到指定的窗口存在。
WinWaitActive (See 28.33)	等到指定的窗口激活。
WinWaitClose (See 28.34)	等到指定的窗口不存在。
WinWaitNotActive (See 28.33)	等到指定的窗口不再激活。
#AllowSameLineComments (See 29.1)	只适用于 AutoIt v2 (.aut) 脚本：允许注解出现在命令的同一行。
#ClipboardTimeout (See 29.2)	更改首次尝试访问剪切板失败后需要间隔的时间。
#CommentFlag (See 29.3)	把脚本的注解符号从分号改为其他字符串。
#ErrorStdOut (See 29.4)	让脚本启动时将任何语法错误发送到标准输出而不是显示错误对话框。
#EscapeChar (See 29.5)	改变脚本的转义符号(例如重音符与反斜线)。
#HotkeyInterval (See 20.1.1)	随同 #MaxHotkeysPerInterval (See 20.1.5) 一起，指定 hotkey (See 4.) 激活的速率，当超过这一速率时，将会显示一个警告对话框。
#HotkeyModifierTimeout (See 20.1.2)	Affects the behavior of hotkey (See 4.) modifiers: CTRL, ALT, WIN, and SHIFT.
#Hotstring (See 20.1.3)	改变 热字符串 (See 5.)的选项或结尾字符。
#IfWinActive / #IfWinExist (See 20.1.4)	创建上下文相关的 热键 (See 4.)和 热字符串 (See 5.)。这样的热键视某类窗口的激活或存在而执行一个不同的动作(或根本不执行)。
#Include (See 17.1)	使脚本表现得好像指定文件的内容就出现在这个位置。
#InstallKeybdHook (See 20.2)	强制无条件地安装键盘钩子。

#InstallMouseHook(See 20.3)	强制无条件地安装鼠标钩子。
#KeyHistory(See 20.4)	设置 KeyHistory (See 20.9) 窗口显示的键盘和鼠标事件的最大数量。你可以将其设为 0 来禁用 key history。
#MaxHotkeysPerInterval(See 20.1.5)	随同 #(See 20.1.1) HotkeyInterval (See 20.1.1) 一起，指定热键激活的速率，当超过这一速率时，将会显示一个警告对话框。
#MaxMem(See 29.15)	设置每个 变量 (See 9.)可使用的最大内存兆数。
#MaxThreads(See 20.1.6)	设置同时启动的 线程 (See 30.19)的最大数量。
#MaxThreadsBuffer(See 20.1.7)	Causes some or all hotkeys (See 4.) to buffer rather than ignore keypresses when their #MaxThreadsPerHotkey (See 20.1.8) limit has been reached.
#MaxThreadsPerHotkey(See 20.1.8)	设置每个 热键 (See 4.)能同时启动的 线程 (See 30.19)的最大数量。
#NoEnv(See 29.19)	避免检查空变量是否为环境变量(推荐所有新建脚本使用)。
#NoTrayIcon(See 22.1)	不显示一个托盘图标。
#Persistent(See 29.21)	让脚本持久地运行(就是说，直到用户关闭它或者遇到了 ExitApp (See 17.7) 命令)。
#SingleInstance(See 22.2)	在一个脚本已经运行时决定是否允许它再次运行。
#UseHook(See 20.1.9)	强制使用钩子来实现部分或全部键盘 热键 (See 4.)。
#WinActivateForce(See 28.3)	跳过温和的方法激活窗口而直接使用强硬的方法。

13. 脚本展示

[NiftyWindows \(美妙的窗口\)-- by Enovatic-Solutions](#): 此脚本让你轻松地控制所有基本的窗口交互，例如拖拽、调整大小、最大化、最小化以及关闭。它最强大的特点是用鼠标右键拖拽来激发。想象将每个窗口分成一个 虚拟的三行三列 9 个单元的网格。中心的单元是最大的一个：通过点击并按住鼠标右键你能抓取并且将一个窗口到处移动。用同样的方法，其他八个单元被用来调整一个窗口的大小。NiftyWindows 也可提供“对齐到网格”，“保持窗口长宽比”，卷起一个窗口至它的标题栏，透明度控制，以及其它有用的快捷功能。

[屏 幕放大镜-- by Holomind](#): 此屏幕放大镜包含了操作系统的一些功能，集多个好处于一身，包括：可自定义的刷新间隔和放大倍数（包括收缩/反放大）；抗锯齿从而提供更优的输出；并且它是开源的（因此，可以从中挑选很多变体，或者你可以自己调整脚本）。

[LiveWindows: 以缩略图查看对话框-- by Holomind](#): 此脚本通过显示每个对话框和它的进度条的一个小型副本（会自动地检测对话框，即使它们位于其它窗口后方）可让你监视下载进度、文件复制以及其它对话框。

预览窗口置顶显示，但使用非常小的屏幕空间(它也能通过拖拽它的边缘来调整大小)。你也能监视任意窗口，通过在感兴趣的区域拖拽出一个矩形选框(用 **Control+Shift+拖拽**)，然后按 **Win+W** 用实时更新的预览窗口来显示选区。

鼠标手势-- by deguix(See 30.29): 此脚本在你按住鼠标右键时会监视如何移动鼠标。如果它看到你“画”出一个能识别的形状或符号，它将启动一个程序 或者执行其它由你选择的自定义动作(就像热键一样)。如何定义手势，请看内置的 **README** 文件。

在任意编辑器内上下文相关的帮助 -- by Rajat(See 30.21): 此脚本让 **Ctrl+2** (或你选择的其它热键)显示选中的 **AutoHotkey** 命令或者关键词的帮助文件页面。如果未选字词，将在当前行开头提取命令名称。

轻松拖拽窗口 (需要 XP/2k/NT)(See 30.22): 通常，窗口仅能在它的标题栏按住左键才能被拖拽。此脚本扩展了此功能以便窗口内的任何一个点都能用来拖拽。要激活此模式，在点击时按住 **CapsLock** 或者鼠标中键，然后拖拽窗口到一个新的位置。

轻松拖拽窗口-- KDE 风格(需要 XP/2k/NT)-- by Jonny: 此脚本使得移动或者调整一个窗口的大小更加轻松：1)按住 **ALT** 键并在窗口内的任何地方点击左键，从而拖拽它到一个新的位置；2)按住 **ALT** 并在窗口内的任何地方点击右键后拖拽，从而轻松地调整它的大小；3)按 **ALT** 两次但在第二次松开前，左键点击鼠标指针下的窗口使它最小化，右键点击使它最大化，或者中键点击来关闭它。

轻松打开收藏的文件夹-- by Savage(See 30.23): 当你在某些类型的窗口激活时按下鼠标中键，此脚本会显示一个你收藏的文件夹菜单。在上面选择一个收藏的文件夹，脚本会在激活的窗口中立即切换到那个文件夹。支持下列窗口类型：1)标准的文件打开或者文件保存对话框；2)资源管理器窗口；3)控制台(命令提示符)窗口。对于不支持的窗口类型，菜单也可以随意地显示，在这种情况下选择的收藏文件夹就在一个新的资源管理器窗口中打开。

IntelliSense(智能感应)-- by Rajat (需要 XP/2k/NT)(See 30.24): 此脚本在你编辑一个 **AutoHotkey** 脚本时会开始监视。当它看到你输入了一个命令紧跟一个逗号或空格时，它将显示那个命令的参数列表来引导你。此外，你可以按 **Ctrl+F1** (或你选择的其它热键)来显示帮助文件里那个命令的页面。要消除命令列表，按 **Escape** 或者 **Enter** 键。

把游戏操纵杆当作鼠标用(See 30.25): 此脚本把游戏操纵杆转换成一个三键鼠标。它允许每个按钮像鼠标按键那样拖拽并且它几乎不使用 CPU 时间。而且，你从中央将操纵杆推得越远，鼠标指针也将移动得越快。你可以在脚本的顶部个性化各种设置。

游戏操纵杆测试脚本(See 30.26): 此脚本能帮助你确定游戏操纵杆的按钮数量以及其它的属性。它也可能显示出你的游戏操纵杆是否需要校准；就是说，它的每个轴的活动范围是否是从百分之零到百分之百就如它应该的那样。如果需要校准，请使用操作系统的控制面板或者你的游戏操纵杆随带的软件。

屏幕键盘(需要 XP/2k/NT)-- by Jon(See 30.27): 此脚本在你屏幕的底端创建一个模拟的键盘来实时显示你按下的按键。我让它来帮助我学盲打(来习惯不去看键盘)。在脚本的顶部可以自定义屏幕键盘的尺寸。而且，你可以双击托盘图标来显示或者隐藏屏幕键盘。

最小化窗口至托盘菜单(See 30.28): 此脚本会分配出一个你选好的热键用来隐藏任何窗口，以便窗口在脚本的托盘菜单底部成为一个菜单项。通过在菜单上选择相应的项目，隐藏的窗口到时能单独地或者一次性全部反隐藏出来。如果脚本由于任何原因而退出，那么它隐藏的所有窗口将被自动地反隐藏出来。

更改 MsgBox 的按钮名称(See 30.30): 这是一个可用的例子脚本，使用一个定时器来更改一个 **MsgBox** 对话框里的按钮名称。虽然按钮名称被更改，但仍需要提交那些按钮原本的名称给 **IfMsgBox** 命令。

数字小键盘区的 000 按键(See 30.31): 此示例脚本让出现在某些小键盘上的特殊的 000 按键变成一个发送等号的按键。你可以用你自定义的命令行替换 "Send, ="

将数字小键盘当鼠标用-- by deguix(See 30.32): 此脚本让你的键盘当鼠标用并且几乎和用一个真正的鼠标一样(也许对某些任务来说甚至更轻松)。它支持多达五个鼠标按键以及鼠标滚轮的调节。它还有自定义移动速度、加速以及 "axis inversion"(反转轴)的特性。

Seek(搜 索)-- by Phi: 浏览开始菜单可能变成一件麻烦事，如果你之前安装了很多程序的话会尤其严重。'Seek' 能让你指定一个不区分大小写的关键词/短语，用它从开始菜单里仅筛选出匹配的程序和目录，以便你能轻松地从少量匹配的项目中打开你的目标程序。这样就不用 在开始菜单里干查找和穿越的苦力活了。

鼠标菜单型的工具提示(需要 XP/2k/NT)-- by Rajat(See 30.33): 此脚本对暂时地按住鼠标中键的动作做出反应而显示一个弹出菜单(译注：在松开后)。通过左键点击选择一个菜单项。在菜单外左键点击来取消菜单。最近的一个 改进是可以根据激活窗口的类型来改变菜单的内容(记事本和 Word 在这里被用作例子)。

屏显(OSD)音量-- by Rajat(See 30.34): 此脚本将指定你选择的热键来提高以及降低总音量和/或波形音量。这两种音量以不同的颜色条图形显示。

窗口遮蔽(将窗口卷起到它的标题栏)-- by Rajat(See 30.35): 此脚本将窗口裁减至它的标题栏，而之后通过按单个热键恢复它原本的大小。任意数量的窗口都能以这种方式裁减(脚本能记住每个窗口的原本大小)。如果脚本因为某种原因退出，所有“卷起”的窗口会被自动地恢复到它们原来的高度。

WinLIRC 客户端(See 30.36): 每当你在遥控器上按一个按键，此脚本就会 WinLIRC 接收到通知。它能用来自动化 Winamp, Windows Media Player 等等。它配置起来很简单。例如，如果 WinLIRC 能识别你遥控器上一个称为 "VolUp" 的按钮，创建一个命名为 VolUp 的标签并在它下方使用命令 "SoundSet +5%" 来按 5% 增加声卡的音量。

1 小时软件-- by skrommel: 这里收集了大量有用的脚本，专业地用简短的描述以及截屏来呈现。

Titan 的脚本: 此收藏包括了有用的脚本例如：

- 1) XML 阅读器/编辑器：对 XML 文件有简单界面来读取以及编辑它的值，就像 JavaScript。
- 2) Anchor(锚)：让 GUI 控制器附属在一个可调大小的 GUI 窗口的右边缘或者底部边缘。
- 3) 函数：wrapper(包装)函数的收藏，它给每个有 OutputVar 参数的 AutoHotkey 命令调用。
- 4) 天气：在托盘菜单中显示当前的天气或者在一个 rich GUI 中显示预报。

Toralf 的脚本: 此收藏包括了有用的脚本例如：

- 1) AHK 窗口信息：在窗口、控件等上面显示信息。
- 2) 电子节目指南：浏览你当地的电视节目/时间表(支持多个国家)。
- 3) 自动语法整理：更改缩进以及在脚本里整理命令使其格式/样式一致。

Sean 的脚本: 包括了有用的脚本例如：

- 1) 网络下载/上传流量计：在一个小小置顶的进度条上显示网络下载/上传了多少 KB。
- 2) StdoutToVar (标准输出到变量)：重定向命令或程序的输出到脚本的某个变量。
- 3) 截取一个矩形屏幕：一个截取部分屏幕并保存为文件(BMP/JPG/PNG/GIF/TIF)的可调用函数。它也能截取透明的窗口和鼠标指针。
- 4) 颜 色放大器/选取器：放大鼠标指针附近的区域，允许选取单像素以及识别它的颜色。
- 5) 嵌 入一个 Internet Explorer 控件：在一个脚本的 GUI 窗口中包含一个 MSIE(微软 IE)控件。控件能显示网页或者其他兼容浏览器的内容。

论坛的 [Scripts & Functions 版块](#): 这是一个近 1000 个待运行可供搜索的脚本和函数收藏。由 AutoHotkey 的用户建设并维护，此存档每天增长并改进。

-- 主页 --

翻译: 天堂之门 menk33@163.com 2008 年 10 月 24 日

14. 更新历史

Compatibility: The change most likely to affect backward compatibility is that floating point numbers stored in variables now have higher precision. Scripts that rely on tiny differences in precision would either need to be reviewed and updated, or have their compatibility improved by using "[SetFormat Float](#)(See 21.10)" (e.g. SetFormat, Float, 0.6) *anywhere* in the script. "SetFormat Float" disables the higher precision, but gives up some of the new improvement in floating point performance.

Performance: The main theme of this release is faster performance. Almost all scripts should run faster -- especially those that make heavy use of [expressions](#)(See 9.) and integer math/comparisons (which may run up to three times as fast). To achieve the full benefit, a script either should avoid using SetFormat or should use [SetFormat's fast mode](#)(See 21.10).

Performance improvements

[Expressions](#)(See 9.) and [function calls](#)(See 10.) are compiled more heavily, making them much faster (especially complex integer expressions, including those with [commas](#)(See 9.)).

Binary numbers are cached for variables to avoid conversions to/from strings. This makes numerical operations involving variables much faster.

Literal integers in expressions and math/comparison commands are replaced with binary integers, which makes them faster; e.g. X+5 and "if x > 5".

[LOOPs](#)(See 17.12), [IFs](#)(See 17.11), and [ELSEs](#)(See 17.5) that have blocks (braces) are faster due to skipping the opening '{'. A side-effect is that the '{' is omitted from [ListLines](#)(See 22.11).

[Thread-creation](#)(See 30.19) performance is improved, which should help rapid-fire threads in [OnMessage\(\)](#)(See 18.11), [RegisterCallback\(\)](#)(See 18.14), and [GUI events](#)(See 19.3).

Changes that might affect existing scripts (other than higher-precision floating point described at the top)

When "[SetFormat, Integer, Hex](#)(See 21.10)" is in effect, assigning a literal decimal integer to a variable also converts it to hex. Usually this is only a display issue.

For [OnMessage\(\)](#)(See 18.11) performance, the message number and HWND arrive as standard numbers rather than appearing unconditionally as hex. Usually this is only a display issue.

To achieve various improvements in performance, scripts now use slightly more memory (proportionate to the number of variables and expressions).

Changed and fixed "if var is time(See 21.8)" and other uses of YYYYMMDDHHMISS date-time stamps to recognize that months outside the range 1-12 are invalid. [thanks Nick]

Changed and improved [dynamic function calling](#)(See 10.) to allow passing more parameters than defined by a function, in which case the parameters are evaluated but discarded. [developed by Lexikos]

Other improvements

Added function [IsFunc\(\)](#)(See 10.), which indicates whether a function may be [called dynamically](#)(See 10.). [developed by Lexikos]

Added the [while-loop](#)(See 17.24), which repeats its commands until its [expression](#)(See 9.) evaluates to false. [developed by Lexikos]

Added an [assume-static mode](#)(See 10.) for functions. [developed by Lexikos]

Added built-in variables [A_IsPaused](#)(See 9.) and [A_IsCritical](#)(See 9.). [developed by Lexikos]

Improved [NumPut\(\)](#)(See 10.) to support UInt64 like [DllCall\(\)](#)(See 18.3). [thanks Sean]

Improved mouse wheel support by adding [WheelLeft](#) and [WheelRight](#)(See 4.) as hotkeys and supporting them in [Send](#)(See 20.12), [Click](#)(See 23.1), and related commands. However, WheelLeft/Right has no effect on operating systems older than Windows Vista. [developed by Lexikos]

Upgraded [compiled script](#)(See 8.) compressor from UPX 3.00 to 3.03.

Fixes

Fixed inability to use [MsgBox](#)(See 19.11)'s timeout parameter when the "Text" parameter had an expression containing commas.

Fixed "Menu, Delete, Item-that's-a-submenu(See 22.13)" not to disrupt the associated submenu. [thanks animeaime & Lexikos]

Fixed the [GUI Hotkey control](#)(See 19.4) to return usable hotkey names even for dead keys (e.g. "^\" instead of Zircumflex). [thanks DerRaphael]

Fixed [RegDelete](#)(See 25.2) so that it won't delete an entire root key when SubKey is blank. [thanks Icarus]

Fixed [registry loops](#)(See 17.16) to support subkey names longer than 259 (rare). In prior versions, such subkeys would either be skipped or cause a crash. [thanks Krzysztof Sliwinski & Eggi]

Fixed FileSelectFolder by providing [an option](#)(See 16.24) to make it compatible with BartPE/WinPE. [thanks markreflex]

Fixed [window/control IDs \(HWNDs\)](#)(See 30.2), which in rare cases wrongly started with 0xFFFFFFFF instead of just 0x. [thanks Micahs]

Fixed inability of [Send commands](#)(See 20.12) to use the Down/Up modifiers with the "}" character. [thanks neovars]

Fixed crash when a [function](#)(See 10.) was called concurrently with an optional [ByRef parameter](#)(See 10.) omitted by one [thread](#)(See 30.19) but not omitted by the other. [thanks DeathByNukes]

Fixed "[Menu, Tray, MainWindow](#)(See 22.13)" to enable the menu items in the main window's View menu. [thanks Lexikos]

Added [dynamic function calling](#)(See 10.). [developed by Lexikos]

Fixed the [Sort command](#)(See 27.12): 1) fixed the "function" option(See 27.12) not to misbehave when it's the last option in the list; 2) fixed the "unique" option(See 27.12) so that when the delimiter is CRLF, the last item can be detected as a duplicate even when it doesn't end in CRLF; 3) fixed the "unique" option not to append a trailing delimiter when the last item is a duplicate. [thanks Roland]

Fixed [RegExMatch\(\)](#)(See 18.12) and [RegExReplace\(\)](#)(See 18.13) to yield correct results even when Haystack and OutputVar are both the same variable. [thanks Superfraggle]

Fixed inability to pass a parameter that is "a variable to which [ClipboardAll](#)(See 30.6) has been assigned". [thanks Joy2DWorld & Lexikos]

Updated RegEx/PCRE from 7.0 to 7.4. For a summary of the major changes, see www.pcre.org/news.txt. For full details of every change and fix, see www.pcre.org/changelog.txt.

Added GUI control "[Tab2](#)(See 19.4)" that fixes rare redrawing problems in the original "Tab" control (e.g. activating a GUI window by clicking on a control's scrollbar). The original Tab control is retained for backward compatibility because "Tab2" puts its tab control after its contained controls in the tab-key navigation order. [thanks Xander]

Fixed [key-up hotkeys](#)(See 4.) like "a up::" not to block the pressing of the "a" key unless the hotkey's [#IfWin criteria](#)(See 20.1.4) are met. [thanks Roland]

Fixed [Round\(Var, NegativeNumber\)](#)(See 10.), which in rare cases was off by 1. [thanks Icarus]

Fixed crash of scripts that end in a syntax error consisting of an orphaned IF-statement (broken by 1.0.47.00). [thanks msgbox of the German forum]

Eliminated the "GetClipboardData" error dialog. Instead, an empty string is retrieved when the data cannot be accessed within the [#ClipboardTimeout](#)(See 29.2) period. [thanks ManaUser & Sean]

Changed GUI [checkboxes](#)(See 19.4) and [radio buttons](#)(See 19.4) to default to "[no word-wrap](#)"(See 19.3) when no width, height, or CR/LF characters are specified. This solves display issues under certain unusual DPI settings. [thanks Boskoop]

Fixed [expressions](#)(See 9.) to allow literal negative hexadecimal numbers that end in "E"; e.g. fn(-0xe). [thanks Laszlo]

Fixed [block syntax](#)(See 17.2) to allow a [function-call](#)(See 10.) immediately to the right of a '}'. [thanks Roland]

Fixed the [Number option](#)(See 19.4) of Edit controls to properly display a balloon tip when the user types something other than a digit. [thanks tfcahm]

Fixed WM_TIMER not to be blocked unless it's posted to the script's main window. [thanks tfcahm]

Fixed [wildcard hotkeys](#)(See 4.) not to acquire [tilde behavior](#)(See 4.) when the same hotkey exists in the script with a tilde. [thanks Lexikos]

Fixed [declaration initializers](#)(See 10.) not to retain whitespace at the end of literal numbers. Also, they now allow spaces between a closing quote and the next comma. [thanks Hardeep]

Fixed [RunAs](#)(See 24.6) not to crash or misbehave when a domain is specified. [thanks Markus Frohnmaier]

Changed [relational operators](#)(See 9.) to yield integers even when the inputs are floating point; e.g. $1.0 < 2.0$ yields 1 vs. 1.0. [thanks Lexikos]

Added support for [function libraries](#)(See 10.), which allow a script to call a function in an external file without having to use [#Include](#)(See 17.1).

Added [RegisterCallback\(\)](#)(See 18.14), which creates a machine-code address that when called, redirects the call to a function in the script. [developed by Jonathan Rennison (JGR)]

Added [NumGet\(\)](#)(See 10.) and [NumPut\(\)](#)(See 10.), which retrieve/store binary numbers with much greater speed than Extract/InsertInteger.

Improved [Sort](#)(See 27.12) with an option to do custom sorting according to the criteria in a callback function. [thanks Laszlo]

Improved [OnMessage\(\)](#)(See 18.11) with an option to allow more than one simultaneous [thread](#)(See 30.19). [thanks JGR]

Improved Critical with an option to change the [message-check interval](#)(See 22.7), which may improve reliability for some usages. [thanks Majkinetor and JGR]

Changed [Critical](#)(See 22.7) to put [SetBatchLines -1](#)(See 17.20) into effect.

Changed the error messages produced by [#ErrorStdOut](#)(See 29.4) to contain a space before the colon. [thanks Toralf]

Fixed [OnMessage\(\) functions](#)(See 18.11) that return one of their own local variables to return the number in that variable, not 0.

Fixed potential crashing of built-in variables that access the registry (e.g. A_AppData, A/Desktop, A/MyDocuments, A/ProgramFiles). [thanks Tekl]

Fixed [A_UserName](#)(See 9.) (broken by 1.0.46.16).

Fixed "Gui, Tab, TabName"(See 19.4)" when used after a previous "Gui Tab". [thanks Toralf]

Improved SetTimer to treat [negative periods](#)(See 17.21) as "run only once". [thanks Majkinetor]

Added "GuiControlGet Hwnd"(See 19.6)", which is a more modular/dynamic way to retrieve a control's HWND. [thanks Majkinetor]

Added built-in variables [A_ThisLabel](#)(See 9.) and [A_ThisFunc](#)(See 9.), which contain the names of the currently-executing label/function. [thanks Titan & Majkinetor]

Fixed [GuiControl](#)(See 19.5), [GuiControlGet](#)(See 19.6), and "Gui ListView"(See 19.7)/[TreeView](#)(See 19.8)" to support [static variables](#)(See 10.) and [ByRefs](#)(See 10.) that point to globals/statics. [thanks Peter]

Fixed [FileInstall](#)(See 16.15) causing the [Random](#)(See 21.9) command to become non-random in [compiled scripts](#)(See 8.). [thanks Velocity]

Reduced the size of [compiled scripts](#)(See 8.) by about 16 KB due to UPX 3.0. [thanks to atnbueno for discovering the optimal command-line switches]

Added the "require administrator" flag to the installer to avoid a warning dialog on Windows Vista. [thanks Roussi Nikolov]

Fixed [hotkeys](#)(See 4.) like `*x` to fire even when `x` is also a hotkey that is prevented from firing due to [#IfWin](#)(See 20.1.4). [thanks Joy2DWorld & Engunneer]

Improved [optional parameters](#)(See 10.) to accept quoted/literal strings as default values.

Improved [ByRef parameters](#)(See 10.) with the ability to be [optional](#)(See 10.) (i.e. they may accept default values). [thanks Corrupt]

Fixed inability to recognize a literal scientific notation number that begins with 0, such as `0.15e+1`. [thanks Laszlo]

Fixed inability to have a [function-call](#)(See 10.) as the first item in certain [comma-separated expressions](#)(See 9.). [thanks Majkinetor]

Fixed WinTitles like "`ahk_id %ControlHwnd%`" in [ControlGet](#)(See 28.1.4)'s [FindString](#)/[Choice](#)/[List](#), and [Control](#)(See 28.1.1)'s [Add](#)/[Delete](#)/[Choose](#). [thanks Freightner & PhiLho]

Improved floating point support to recognize scientific notation; e.g. `1.2e-5` (the decimal point is mandatory). Also improved "[SetFormat Float](#)(See 21.10)" with an option to output in scientific notation. [thanks Laszlo]

Fixed [StringSplit](#)(See 27.21) inside assume-local [functions](#)(See 10.) so that it creates a local [array](#)(See 30.5) even when [OutputArray0](#) exists as a global but not a local. [thanks KZ]

Improved [ListView's item-changed notification \("I"\)](#)(See 19.7) to indicate via [ErrorLevel](#) whether the item has been selected/deselected, focused/unfocused, and/or checked/unchecked. [thanks foom]

Added an additional layer of protection to [compiled scripts](#)(See 8.). It is recommended that scripts containing sensitive data or source code be recompiled with the [/NoDecompile switch](#)(See 8.).

Fixed ":=(See 21.12)" deep inside expressions when used to assign the result of a recursive [function](#)(See 10.) to a [local variable](#)(See 10.) (broken by 1.0.46.06). [thanks Laszlo]

Fixed inability to pass certain [ternary expressions](#) (See 9.)to [ByRef parameters](#)(See 10.). [thanks Titan]

Fixed "[GuiControlGet](#)(See 19.6), OutputVar, Pos" so that it doesn't make the OutputVar blank. [thanks PhiLho]

Changed and fixed continuation sections so that the "[Comment](#)" option(See 8.) doesn't force the [LTrim option](#)(See 8.) into effect. [thanks Titan]

Changed the [Terminal Server Awareness flag](#) back to "disabled" on AutoHotkey.exe and compiled scripts. This improves flexibility and backward compatibility (see [discussion](#) at forum).

Fixed unreliability of [ComSpec](#)(See 9.) and [environment variables](#)(See 9.) on Windows 9x (broken by v1.0.46.07). [thanks Loriss]

Changed: When AutoHotkey.exe is launched without a script specified, it will now run (or prompt you to create) the file AutoHotkey.ahk in the [My Documents](#)(See 9.) folder. The only exception is when AutoHotkey.ini exists in the working directory, in which case it uses the old behavior of executing that file.

Improved [DIICall](#)(See 18.3) to support an integer in place of the function name, which is interpreted as the address of the function to call. [thanks Sean]

Fixed crash of illegally-named [dynamic variables](#)(See 30.5) on the left of an equal-sign assignment (broken by v1.0.45). [thanks PhiLho]

Fixed [FileMoveDir](#)(See 16.17)'s "Option 2" to work properly even when the directory is being both renamed and moved. [thanks bugmenot]

Fixed inability to pass a variable [ByRef](#)(See 10.) if that same expression changed it from empty to non-empty (when [#NoEnv](#)(See 29.19) is absent). [thanks Joy2DWorld]

Changed DIICall's [A_LastError](#)(See 18.3) to reflect only changes made by the script, not by AutoHotkey itself. [thanks Azerty]

Applied minor fixes and improvements to [regular expressions](#)(See 30.14) by upgrading from PCRE 6.7 to 7.0. One of the most notable improvements is the `a option, which recognizes any type of newline (namely `r, `n, or `r` n). Similarly, the \R escape sequence means "any single newline of any type". See also: [Full PCRE changelog](#)

Changed and fixed all [Control commands](#)(See 28.1.1) and [StatusBarWait](#)(See 28.11) to obey [SetTitleMatchMode RegEx](#)(See 28.8) as documented.

Changed [RegExReplace\(\)](#)(See 18.13) to return the original/unaltered string rather than "" when an error occurs.

Changed: Enabled the [Terminal Server Awareness](#) flag on AutoHotkey.exe and [compiled scripts](#)(See 8.).

Improved performance when [assigning](#)(See 9.) large strings returned from [user-defined functions](#)(See 10.). [thanks Laszlo]

Fixed the [Input command](#)(See 20.11) to allow named end keys like {F9} to work even when the shift key is being held down (broken by v1.0.45). [thanks Halweg]

Fixed inability of "Gui Show(See 19.3)" to focus the GUI window when the tray menu is used both to reload the script and to show the GUI window. [thanks Rnon]

Fixed inability to pass some types of [assignments](#) (:=)(See 9.) to a [ByRef](#)(See 10.) parameter. [thanks Laszlo]

Fixed inability to pass the result of an [assignment](#) (:=)(See 9.) to a [ByRef](#)(See 10.) parameter. [thanks Titan]

Fixed [ListView](#)(See 19.7)'s floating point sorting to produce the correct ordering. [thanks oldbrother/Goyyah/Laszlo]

Fixed [environment variables](#)(See 9.) to work properly as input variables in various commands such as [StringLen](#)(See 27.17) and [StringReplace](#)(See 27.20) (broken by 1.0.44.14). [thanks Camarade_Tux]

NOTE: Although this release has been extensively tested, several low-level enhancements were made. If you have any mission-critical scripts, it is recommended that you retest them and/or wait a few weeks for any bugs to get fixed.

Fixed comma-separated [declaration initializers](#)(See 10.) such as "*local* *x* = 1, *y* = 2" to work even when immediately below an if/else/loop statement.

Fixed [comma-separated expressions](#)(See 9.) so when the leftmost item is an assignment, it will occur before the others rather than after. [thanks Laszlo]

Changed and fixed [function-calls](#)(See 10.) so that any changes they make to [dynamic variable names](#)(See 9.), [environment variables](#)(See 9.), and [built-in variables](#)(See 9.) (such as [Clipboard](#)(See 30.6)) are always visible to subsequent parts of the expression that called them.

Changed: When a [multi-statement comma](#)(See 9.) is followed immediately by a variable and an equal sign, that equal sign is automatically treated as a [:= assignment](#)(See 21.12). For example, all of the following are assignments: *x*:=1, *y*=2, *a*=

Changed [comma-separated expressions](#)(See 9.) to produce the following effects: 1) the leftmost */=* operator becomes [true divide](#)(See 9.) rather than [EnvDiv](#)(See 21.2); 2) blank values are not treated as zero in math expressions (thus they yield blank results).

Improved the performance of [expressions](#)(See 9.) by 5 to 20% (depending on type).

Improved the [new assignment operators such as .=](#)(See 9.) to support the [Clipboard variable](#)(See 30.6) (even in [comma-separated expressions](#)(See 9.)).

Improved the [.= operator](#)(See 9.) so that it doesn't require a space to its left.

Improved GUI controls to accept [static variables](#)(See 10.) as their [associated variables](#)(See 19.3) (formerly only globals were allowed).

Added option [HwndOutputVar](#)(See 19.3) to "Gui Add", which stores a control's HWND in OutputVar. [thanks Corrupt & Toralf]

NOTE: Although this release has been extensively tested and is not expected to break any existing scripts, several low-level enhancements were made. If you have any mission-critical scripts, it is recommended that you retest them and/or wait a few weeks for any bugs to get fixed.

Added function [SubStr\(\)](#)(See 10.), which retrieves the specified number of characters at the specified position in a string.

Added assignment operators `//=`, `.=`, `|=`, `&=`, `^=`, `>>=`, and `<<=`(See 9.), which can be used anywhere in expressions. For example, `Var .= "abc"` appends the string "abc" to `Var`'s current contents.

Added full support in expressions for the operators `:=`, `++`, `--`, `+=`, `-=`, `*=`, and `/=`(See 9.) (formerly, they could be used only as the leftmost operator on a line). All [assignment operators](#)(See 9.) (especially `++` and `--`(See 9.)) behave in a C-like way when their result is used by some other operator.

Added the [ternary operator \(?:\)](#)(See 9.), which is a shorthand replacement for the [if-else statement](#)(See 17.11). For example, `var := x>y ? 2 : 3` assigns the value 2 if `x` is greater than `y`; otherwise it assigns 3.

Added support for [comma-separated expressions](#)(See 9.), which allow a single line to contain multiple assignments, function calls, and other expressions. [thanks PhiLho & Titan]

Improved [variable declarations](#)(See 10.) to support initialization on the same line. Note: A [static variable](#)(See 10.)'s initialization occurs only once, before the script begins executing.

Improved [line continuation](#)(See 8.) to support all expression operators. For example, a line that starts with "?" or "+" is automatically appended to the line above it.

Improved performance of operators `".."`(See 9.) and `".=`(See 9.) to be as fast as the percent-sign method of appending a string.

Improved expressions to allow more types of [consecutive unary operators](#)(See 9.) such as `!!Var`. [thanks Laszlo]

Changed [Critical](#)(See 22.7) to check messages less often (20 vs. 10ms), which improves the reliability of frequently-called [OnMessage functions](#)(See 18.11). [thanks Majkinetor]

Changed: A variable named simply "?" is no longer valid in expressions due to the new [ternary operator](#)(See 9.).

Fixed [hotkeys](#)(See 4.) to support "`:::`" (colon as a hotkey) and "`: & x`" (colon as a [hotkey prefix](#)(See 4.)).

Fixed the installer to remove psapi.dll from the AutoHotkey folder (except on Windows NT4). This avoids a conflict with Internet Explorer 7. [thanks to all who reported it]

Fixed crash on Windows 9x when a script doesn't actually run (e.g. due to syntax error) (broken by v1.0.45). [thanks rogerg]

Changed "[Control Style|ExStyle](#)(See 28.1.1)" to report ErrorLevel 0 vs. 1 when the requested style change wasn't necessary because it was already in effect.

Improved [#Include](#)(See 17.1) to support `%A_AppData%`(See 9.) and `%A_AppDataCommon%`(See 9.). [thanks Tekl]

Fixed [file-pattern loops](#)(See 16.31) not to crash when recursing into paths longer than 259 characters. In addition, such paths and files are now ignored (skipped over) by file-pattern loops, [FileSetAttrib](#)(See 16.25), and [FileSetTime](#)(See 16.26). [thanks PhiR]

Fixed [functions](#)(See 10.) that call themselves and assign the result to one of their own [locals](#)(See 10.) (broken by v1.0.45). [thanks bjennings]

Fixed crash of scripts containing [regular expressions](#)(See 30.14) that have compile errors. [thanks PhiLho]

Fixed [GuiControl](#)(See 19.5) not to convert [checkboxes](#)(See 19.4) into 3-state unless requested. [thanks JBensimon]

Changed [UrlDownloadToFile](#)(See 22.24) to announce a user-agent of "AutoHotkey" to the server rather than a blank string. [thanks jaco0646]

Improved [continuation sections](#)(See 8.) to support semicolon comments inside the section via the option-word *Comments*.

Fixed [StringUpper](#)(See 27.18) and [StringLower](#)(See 27.18) to work when OutputVar is the clipboard (broken by v1.0.45). [thanks songsoverruins]

Fixed the hotkeys ~Alt, ~Control, and ~Shift to fire upon press-down rather than release (broken by v1.0.44).

Background: Without the tilde, Alt/Control/Shift fire upon release to avoid taking over both the left and right key. But a specific left/right hotkey like LAlt or RShift fires upon press-down.

Fixed [FileReadLine](#)(See 16.18) and [FileSelectFile](#)(See 16.23) not to crash or misbehave when other [threads](#)(See 30.19) interrupt them (broken by v1.0.45). [thanks toralf]

Fixed [RegExMatch\(\)](#)(See 18.12) so that when there's no match, named subpatterns are properly set to "" in the output array. [thanks PhiLho]

Fixed [RegExMatch\(\)](#)(See 18.12)'s "J" option to properly write duplicate named subpatterns to the output array. [thanks PhiLho]

Changed [SetWorkingDir](#)(See 16.33) and [#Include DirName](#)(See 17.1) to succeed even for a root directory such as C: that lacks a backslash.

Improved [DIICall\(\)](#)(See 18.3) to display a warning dialog if the called function writes to a variable of zero capacity.

NOTE: Although this release has been extensively tested and is not expected to break any existing scripts, several low-level performance enhancements were made. If you have any mission-critical scripts, it is recommended that you retest them and/or wait a few weeks for any bugs to get fixed.

Added support for [regular expressions](#)(See 30.14) via [RegExMatch\(\)](#)(See 18.12), [RegExReplace\(\)](#)(See 18.13), and [SetTitleMatchMode RegEx](#)(See 28.8). [thanks Philip Hazel & PhiLho]

Improved performance and memory utilization of [StringReplace](#)(See 27.20).

Improved performance of the [:= operator](#)(See 21.12) for expressions and functions involving long strings.

Improved [ControlClick](#)(See 23.2) with a new option "NA" that avoids activating the target window (this mode also improves reliability in some cases). In addition, it's been documented that [SetControlDelay -1](#)(See 28.1.13) can improve the reliability of ControlClick in some cases. [thanks nnesori]

Changed [GUI buttons](#)(See 19.4) to default to "[no word-wrap](#)"(See 19.3) when no width, height, or CR/LF characters were specified. This may solve button display issues under some desktop themes.

Fixed "[Transform HTML](#)(See 21.11)" for the following characters: &`n><

Fixed misinterpretation of lines starting with "if not is" such as "if not **IsDone**".

Fixed inability of "[Gui Show](#)(See 19.3)" to move a window vertically downward to where its bottommost row of pixels is now.

Fixed inability to use [GroupActivate](#)(See 28.2.1) as the only line beneath an IF or ELSE.

Fixed inability of the [Input command](#)(See 20.11) to differentiate between end-keys enclosed in braces and their (un)shifted counterparts; e.g. '{' vs. '['. [thanks Laszlo]

Visit www.autohotkey.com/changelog/ for the older changes and a list of planned features.

15. 环境管理

15.1 ClipWait

等到[剪贴板](#)(See 30.6)包含数据。

`ClipWait [, SecondsToWait, 1]`

参数

SecondsToWait	如果省略，此命令会无限地等待。否则，它将等待不超过这么多秒（可包含小数点或者是 一个表达式(See 9.)）。指定为 0 则等同于 0.5。
1	如果此参数被省略，命令将更有选择性，明确地等待文本或文件的出现（“文本”包括任何 在你粘贴进记事本时会产生文本的东西）。如果此参数为 1（可以是一个表达式(See 9.)）， 命令等待任何类型的数据出现在剪贴板。

ErrorLevel

如果等待时间到期，ErrorLevel(See 30.8) 会被设置成 1。否则（例如剪贴板包含数据），ErrorLevel 设置成 0。

说明

使用此命令比一个你自己用来检查剪贴板是否为空的循环要好。这是因为剪贴板从未被此命令打开，因此它执行地更好并且避免了对可能使用剪贴板的其他应用程序的干扰。

此命令把任何可转换成文本的东西（例如 HTML）视为文本。它也把文件视为文本，例如那些在资源管理器窗口通过 Control-C 复制的文件。每当剪贴板变量(%clipboard%)在脚本中被使用，这样的文件就被自动地转换为它们的文件名（以绝对路径）。详见剪贴板(See 30.6)。

最后的参数是 1 时，任何数据出现在剪贴板上命令都将满意。它能和 ClipboardAll(See 30.6) 一起用来保存非文本数据，例如图片。

当命令处于等待状态，新的线程(See 30.19)能通过 热键(See 4.)、自定义菜单项(See 22.13)或者定时器(See 17.21)被运行。

相关命令

[Clipboard\(See 30.6\)](#), [WinWait\(See 28.32\)](#), [KeyWait\(See 20.10\)](#)

范例

```
clipboard = ;清空剪贴板
Send, ^c
ClipWait, 2
if ErrorLevel
{
    MsgBox, 尝试复制文本到剪贴板失败。
}
return
```

```

}

MsgBox, clipboard = %clipboard%
return

```

翻译：天堂之门 menk33@163.com 2008 年 11 月 6 日

15.2 EnvGet

获取一个环境变量。

`EnvGet, OutputVar, EnvVarName`

参数

<code>OutputVar</code>	存储字符串的变量名。
<code>EnvVarName</code>	要获取的 environment variable (See 9.)(环境变量) 的名称。例如： <code>EnvGet, OutputVar, Path</code>

注意

如果指定的环境变量为空或不存在，`OutputVar` 将被设为空。

操作系统把每个环境变量限制在 32 KB 大小的文本内。

相关命令

[EnvSet](#)(See 15.3), [#NoEnv](#)(See 29.19), [environment variables](#)(See 9.), [EnvUpdate](#)(See 15.4), [SetEnv](#)(See 22.19), [Run](#)(See 24.5), [RunWait](#)(See 24.5)

示例

```
EnvGet, OutputVar, LogonServer
```

翻译：Xianggee 修正：天堂之门 menk33@163.com 2008 年 8 月 6 日

15.3 EnvSet

向包含在系统环境中的 [variable](#)(See 9.)(变 量) 写入值。

`EnvSet, EnvVar, Value`

参数

<code>EnvVar</code>	要使用的 environment variable (See 9.)(环境 变量) 名称，例如，"COMSPEC" 或 "PATH"。
<code>Value</code>	要设置为 environment variable (See 9.) 的值。

ErrorLevel

如果发生问题则 [ErrorLevel\(See 30.8\)](#) 设置为 1，反之为 0。

注意

操作系统把每个环境变量限制在 32 KB 大小的文本内。

通过此命令创建和修改的环境变量，只能由脚本通过 [Run\(See 24.5\)](#) 或 [RunWait\(See 24.5\)](#) 命令启动的程序读取。查看 [environment variables\(See 9.\)](#) 获得更多细节。

此命令和 [SetEnv\(See 22.19\)](#) 分开存在，因为 [normal script variables\(See 9.\)](#)(标准的脚本变量) 并没有存储在系统环境中。这是由于性能将会变得较差而且也是因为操作系统把环境变量限制在 32 KB 大小内。

相关命令

[EnvGet\(See 15.2\)](#), [#NoEnv\(See 29.19\)](#), [environment variables\(See 9.\)](#), [EnvUpdate\(See 15.4\)](#), [SetEnv\(See 22.19\)](#), [Run\(See 24.5\)](#), [RunWait\(See 24.5\)](#)

示例

`EnvSet, AutGUI, 一些要放入变量的文本。`

翻译: Xianggee 修正: 天堂之门 menk33@163.com 2008年8月6日

15.4 EnvUpdate

通知操作系统和所有正在运行的程序 [environment variable\(s\)\(See 9.\)](#)(环境变量) 已改变。

[EnvUpdate](#)

[ErrorLevel](#)

如果发生问题则 [ErrorLevel\(See 30.8\)](#) 设置为 1，反之为 0。

注意

更新操作系统的环境。类似于注销后再登入的效果。

例如，在 Windows NT/2000/XP 或之后的版本上，修改了以下注册表键值之后，EnvUpdate 可以被用来广播这一改变： HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session Manager\Environment

相关命令

[EnvSet](#)(See 15.3), [RegWrite](#)(See 25.4)

示例

EnvUpdate

翻译: Xianggee 修正: 天堂之门 menk33@163.com 2008 年 8 月 6 日

16. 文件、目录以及磁盘管理

16.1 Drive

如下所述, sub-command、drive, 和参数 value 之间相互依赖。

Label, Drive [, NewLabel]: 将 *Drive* (驱动器)的卷标改为 *NewLabel* (如果省略 *NewLabel*, 驱动器卷标为空)。*Drive* 是字母后跟一个冒号和可选的反斜杠(可能是 UNC 和驱动器映射)。例如: *Drive, Label, C:, Seagate200*

按照以下例子, 取得当前驱动器的卷标: [DriveGet](#)(See 16.2), OutputVar, Label, C:

Lock, Drive: 锁定驱动器, 防止其正在工作的时候被弹出。例如: "Drive, Lock, D:". 大多数的驱动器不能 "locked open"(锁定之后被打开)。然而, 在驱动器打开时锁定它, 将会使驱动器在关闭后进入锁定状态。对于不支持锁定的驱动器(像大多数的仅读驱动器), 此命令将不起作用。该命令同样不支持 Windows 95/98/Me 下非 IDE 驱动器。如果脚本在结束前锁定了某个驱动器, 则该驱动器将一直保持锁定状态, 直到其它的脚本或程序将其解锁, 或系统重启。如果指定的驱动器不存在或者不支持锁定特性, *ErrorLevel* (警告级别)将被设为 1, 否则为 0。

Unlock, Drive: 功能同上述命令相反。在 Window NT/2000/XP 及以后版本, 如果驱动器锁定了多次, 则需要多次执行 *Unlock* 命令。例如, 如果 "Drive, Lock, D:" 执行了三次, 需要执行三次 "Drive, Unlock, D:" 来解锁。事实上, 因为不能确定驱动器是否处于锁定状态, 因此, 经常需要一个变量(Sec 9.)来记录锁定的状态。

Eject [, Drive, 1]: 弹出或收回 CD 或 DVD 驱动器的托盘(弹出其它类型的影音设备或驱动器, 请参见本页结尾使用的 *DllCall*)。

如果省略 *Drive*, 则使用默认的 CD / DVD 驱动器。省略最后一个参数将则弹出托盘, 将其设为 1 则该命令收回或关闭托盘。例如: *Drive, Eject, D:, 1*

Drive Eject 命令在托盘弹出或收回完成后返回。如果托盘成功到达指定状态(打开或关闭), *ErrorLevel*(See 30.8) 将被设为 0(就是"没有错误")。

对于网络驱动器或者非 CD / DVD 驱动器, *Drive Eject* 命令很可能无法正常工作。因为这些或其它原因导致的错误, *ErrorLevel*(See 30.8) 将被设为 1。

通过检测执行命令所花费的时间，可以有效地确定托盘以前的状态。例如：下面的热键将触发托盘到相反状态(打开或关闭)：

```
#C:::  
  
Drive, Eject  
  
; 如果很短时间内完成了这条命令，托盘很可能已经弹出了。  
  
; 那样的话，收回托盘。  
  
if A_TimeSinceThisHotkey < 1000 ; 调整到合适的时间。  
  
    Drive, Eject,, 1  
  
return
```

如果想测定 CD 或 DVD 的影音播放状态(比如正在播放，暂停，打开等)，请参见 [DriveGet\(See 16.2\)](#) 命令。

ErrorLevel

出现错误时，[ErrorLevel\(See 30.8\)](#) 将被设为 1，否则为 0。

注意

下面的例子对弹出方法进行了改动，使之不仅能用于 CD 或 DVD 驱动器，还能用于 media/devices(影音设备)：

```
; 根据需要更新第二行的驱动器标号(不用考虑其它行)。  
  
Driveletter = I: ; 设定为你想弹出的驱动器的标号。  
  
  
hVolume := DllCall("CreateFile"  
  
    , Str, "\.\\" . Driveletter  
  
    , UInt, 0x80000000 | 0x40000000 ; GENERIC_READ | GENERIC_WRITE  
  
    , UInt, 0x1 | 0x2 ; FILE_SHARE_READ | FILE_SHARE_WRITE  
  
    , UInt, 0  
  
    , UInt, 0x3 ; OPEN_EXISTING  
  
    , UInt, 0, UInt, 0)  
  
if hVolume <> -1  
  
{  
  
    DllCall("DeviceIoControl"
```

```

    , UInt, hVolume
    , UInt, 0x2D4808 ; IOCTL_STORAGE_EJECT_MEDIA
    , UInt, 0, UInt, 0, UInt, 0, UInt, 0
    , UIntP, dwBytesReturned ; Unused.
    , UInt, 0)

DllCall("CloseHandle", UInt, hVolume)
}

```

相关命令

[DriveGet](#)(See 16.2), [DriveSpaceFree](#)(See 16.3)

示例

```

Drive, Label, D:, BackupDrive
Drive, Eject,, 1 ; 收回(关闭)默认 CD 或 DVD 驱动器的托盘。

```

16.2 DriveGet

获得关于计算机驱动器各种类型的信息。

`DriveGet, OutputVar, Cmd [, Value]`

参数

OutputVar	要存储 <i>Cmd</i> 结果的变量名称。
Cmd, Value	见下表。

Cmd, Value

参数 *Cmd* 和 *Value* 相互依赖，其用法描述如下。如果遇到问题 *OutputVar* 会被置空并且 [ErrorLevel](#)(See 30.8) 被设为 1 。

List [, Type]: 设置 *OutputVar* 为一串字母，其中每个字母代表系统中的一个驱动器号。例如: ACDEZ 。如果省略参数 *Type*，将获取所有类型的驱动器。否则，参数 *Type* 应该设定为 CDROM, REMOVABLE, FIXED, NETWORK, RAMDISK, UNKNOWN 其中之一以获取该特定类型的驱动器。

Capacity (或 Cap), Path: 以兆字节为单位取得参数 *Path* (例如: C:\) 的总容量。用 [DriveSpaceFree](#)(See 16.3) 命令来确定可用空间。

Filesystem (或 FS), Drive: 获取参数 *Drive* 的文件系统类型，这里参数 *Drive* 由驱动器字母后接一个冒号和一个可选的反斜线组成，或者是一个 UNC(通用命名规则) 名称如 \\server1\share1 。

OutputVar 将被设为以下单词中的一个：FAT, FAT32, NTFS, CDFS (一般指 CD), UDF (一般指 DVD)。如果驱动器不包含格式化的存储介质，*OutputVar* 会被置空并且 [ErrorLevel\(See 30.8\)](#) 被设为 1。

Label, Drive: 获取参数 *Drive* 的卷标，这里参数 *Drive* 由驱动器字母后接一个冒号和一个可选的反斜线组成，或者是一个 UNC 名称如 `\server1\share1`。要更改卷标，参照此例：[Drive\(See 16.1\)](#), Label, C:, MyLabel

Serial, Drive: 获取参数 *Drive* 的卷序列号以十进制整数表示，这里参数 *Drive* 由驱动器字母后接一个冒号和一个可选的反斜线组成，或者是一个 UNC 名称如 `\server1\share1`。如何转换成十六进制请看 [SetFormat\(See 21.10\)](#) 命令。

Type, Path: 获取参数 *Path* 的驱动器类型，它是后面单词中的一个：Unknown, Removable, Fixed, Network, CDROM, RAMDisk。

Status, Path: 获取参数 *Path* 的状态，它是后面单词中的一个：Unknown (可能表示未格式化/RAW), Ready, NotReady (通常表示不含存储介质的可移动驱动器), Invalid (参数 *Path* 不存在或者是一个当前无法访问的网络驱动器等)

StatusCD [, Drive]: 获取 CD 或 DVD 驱动器中存储介质的状态，这里参数 *Drive* 由驱动器字母后接一个冒号组成 (如果省略参数 *Drive*, 将使用默认的 CD/DVD 驱动器)。如果状态无法确定 *OutputVar* 将被置空。否则，它将设定为下列字符串之一：

not ready	驱动器未准备好被访问，也许是因为正忙于写操作。已知的限制：当驱动器里是一个 DVD 而不是 CD 时，也会出现"not ready" 的情况。
open	驱动器里没有光盘，或者托盘已弹出。
playing	驱动器正在播放光盘。
paused	先前播放的音频或视频目前已暂停。
seeking	驱动器正在查找。
stopped	驱动器里有 CD 但是当前没有在访问。

此命令很可能对网络驱动器或者非 CD/DVD 驱动器不起作用；如果由此或者其他原因而失败，*OutputVar* 将被置空并且 [ErrorLevel\(See 30.8\)](#) 设为 1。

如果托盘刚被关闭，命令完成之前会有一定延迟。

要弹出或者缩回托盘，请看 [Drive\(See 16.1\)](#) 命令。

ErrorLevel

出错时 [ErrorLevel\(See 30.8\)](#) 被设为 1，否则为 0。

注意

其中某些命令能接受网络共享名称作为参数 *Path*，例如 `\MyServer\MyShare\`

相关命令

[Drive](#)(See 16.1), [DriveSpaceFree](#)(See 16.3)

示例

```
; 这是一个可用的示例脚本。
FileSelectFolder, folder, , 3, 选一个驱动器来分析:
if folder =
return
DriveGet, list, list
DriveGet, cap, capacity, %folder%
DrivespaceFree, free, %folder%
DriveGet, fs, fs, %folder%
DriveGet, label, label, %folder%
DriveGet, serial, serial, %folder%
DriveGet, type, type, %folder%
DriveGet, status, status, %folder%
MsgBox 所有的驱动器: %list%`n 选中的驱动器: %folder%`n 驱动器类型: %type%`n 状
态: %status%`n 容量: %cap% M`n 可用空间: %free% M`n 文件系统: %fs%`n 卷
标: %label%`n 序列号: %serial%
```

翻译: tcgbp 修正: 天堂之门 menk33@163.com 2008 年 9 月 18 日

16.3 DriveSpaceFree

获得一个驱动器的剩余磁盘空间信息，以 Megabytes(兆字节) 表示。

`DriveSpaceFree, OutputVar, Path`

参数

<code>OutputVar</code>	用来存储结果的变量，四舍五入到最接近整数。
<code>path</code>	需要获得信息的驱动器路径。由于 NTFS 支持 <code>mounted volumes</code> (挂载卷) 和 <code>directory junctions</code> (目录联结)，有时候相同“驱动器”下的不同文件夹可能有不同的剩余空间总量。

注意

`OutputVar` 被设置为以兆字节显示的剩余磁盘空间总量(四舍五入到最接近兆字节)。

相关命令

[Drive](#)(See 16.1), [DriveGet](#)(See 16.2)

示例

DriveSpaceFree, FreeSpace, c:\

翻译: Xianggee 修正: 天堂之门 menk33@163.com 2008 年 8 月 6 日

16.4 FileAppend

在文件的结尾处追加文本(如果需要, 首先创建文件)。

FileAppend [, Text, Filename]

参数

Text	<p>要追加到文件的文本。此文本可以包含换行符(`n)来开始新的一行。另外, 可以使用一个连续部分(See 8.)将一个单独的长行分解成多个较短的部分。</p> <p>如果 <i>Text</i> 为空, <i>Filename</i> 将被创建为一个空白文件(但如果文件已存在, 它的修改时间将被更新)。</p> <p>如果 <i>Text</i> 是 %ClipboardAll%(See 30.6) 或一个之前被指定为 ClipboardAll 的变量, <i>Filename</i> 将会被剪贴板的全部内容无条件地覆盖(也就是不需要 FileDelete(See 16.9))。</p>
Filename	<p>要被追加的文件名, 如果没有指定一个绝对路径, 将假设它在 %A_WorkingDir%(See 9.)中。</p> <p>二进制模式: 要以二进制方式而不是文本方式追加的话, 在文件名前添加星号。这将使换行符(`n)被写成一个单独的换行(LF)而不是 Windows 标准的 CR+LF(回车+换行)。例如: *C:\My Unix File.txt</p> <p>如果文件尚未被打开(由于在文件读取循环(See 16.32)中), 那么即使 <i>Text</i> 包含任何一对回车和换行(`r`n), 文件也将自动以二进制方式打开。换言之, 上一段提到的星号选项将自动生效。不过, 当 <i>Text</i> 包含 `r`n 时指定星号可以改善性能, 因为程序将不需要检索 <i>Text</i> 中的 `r`n。</p> <p>标准输出 (stdout): 给 <i>Filename</i> 指定一个星号 (*) 将使 <i>Text</i> 以标准输出(stdout)发送。这样的文本可以重定向到一个文件、传送到另一个 EXE 程序或被高级的文本编辑器(See 29.4)获取。例如, 如果在一个命令提示符中, 将可以有效地输入以下文本: "%ProgramFiles%\AutoHotkey\AutoHotkey.exe" "My Script.ahk" >"Error Log.txt"</p> <p>不过, 以标准形式输出的文本并不会在执行它的命令提示符中出现。这可以通过将一个脚本的输出传送给另一个命令或程序来解决。例如:</p> <ol style="list-style-type: none"> 1) "%ProgramFiles%\AutoHotkey\AutoHotkey.exe" "My Script.ahk" more 2) For /F "tokens=*" %L in ("%%ProgramFiles%\AutoHotkey\AutoHotkey.exe" "My Script .ahk") do @Echo %L

ErrorLevel

出错时, [ErrorLevel\(See 30.8\)](#) 被设为 1, 否则为 0。

注意

要覆盖一个已存在的文件, 需在使用 `FileAppend` 前用 [FileDelete\(See 16.9\)](#) 命令删除文件。

文本附加到文件后, 目标文件自动关闭(除了在[文件读取/写入循环\(See 16.32\)](#)中, 当 `FileAppend` 以它的单参数方式使用时)。

相关命令

[FileRead\(See 16.19\)](#), [file-reading loop\(See 16.32\)](#), [FileReadLine\(See 16.18\)](#), [IniWrite\(See 16.30\)](#), [FileDelete\(See 16.9\)](#), [OutputDebug\(See 22.14\)](#), [continuation sections\(See 8.\)](#)

示例

```
FileAppend, Another line.`n, C:\My Documents\Test.txt
```

;下面的例子中, 使用了[连续部分\(See 8.\)](#)来增强文本的可读性和可维护性:

```
FileAppend,
```

```
(
```

一行文本。

默认的, 在上一行和这行之间的硬回车([Enter](#))将被写入文件。

此行被一个 `tab`(制表符) 错开; 默认的, 这个制表符也将被写入文件。

变量引用例如 `%Var%` 被默认地扩展。

```
), C:\My File.txt
```

;下面这个例子演示了怎样使用操作系统内建的 `FTP` 命令自动化上传文件到 `FTP`。此脚本已在 Windows XP 和 98se 上测试过。

```
FTPCommandFile = %A_ScriptDir%\FTPCommands.txt
```

```
FTPLogFile = %A_ScriptDir%\FTPLog.txt
```

`FileDelete %FTPCommandFile% ; 以免前一次运行时被过早地终止而影响本次。`

```
FileAppend,
```

```
(
```

open host.domain.com

username

password

binary

```

cd htdocs
put %VarContainingNameOfFile%
delete SomeOtherFile.htm
rename OldFileName.htm NewFileName.htm
ls -l
quit
), %FTPLogFile%

```

RunWait %comspec% /c ftp.exe -s:"%FTPLogFile%" >"%FTPLogFile%"
 FileDelete %FTPLogFile%;出于安全考虑而删除文件。
 Run %FTPLogFile%;显示日志文件，以供查阅。

翻译：lwjeee 修正：天堂之门 menk33@163.com 2008年11月7日

16.5 FileCopy

复制一个或多个文件。

FileCopy, SourcePattern, DestPattern [, Flag]

参数

SourcePattern	一个单独文件或文件夹的名称，或是通配符样式，如 C:\Temp*.tmp。如果没有定义绝对路径，则假定 <i>SourcePattern</i> 是在 %A_WorkingDir%(See 9.)。
DestPattern	目标的名称或通配符样式，如果没有定义绝对路径，则假定 <i>SourcePattern</i> 是在 %A_WorkingDir%(See 9.)。执行一个简单的复制 -- 保留已存在文件名称 -- 仅指定文件夹名称像那些功能相同的例子显示的那样： <i>FileCopy, C:*.txt, C:\My Folder</i> <i>FileCopy, C:*.txt, C:\My Folder*.*</i>
Flag	(可选参数) 此 flag(标志) 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 此参数可为 <i>expression</i> (See 9.)(表达式)，甚至可以取值为 true 或 false (因为在内部 true 和 false 以 1 和 0 表示)。

ErrorLevel

ErrorLevel(See 30.8) 将设置为由于错误而无法复制的文件数，或相反为 0。如果无法复制任何文件，则 AutoIt v2 (.aut) 脚本会将 ErrorLevel 设为 1。

另外一种情况，如果源文件是一个单独的文件(无通配符)并且文件不存在，ErrorLevel 将设为 0。要侦测这种情况，在复制源文件前对它使用 **IfExist**(See 16.27) 或 **FileExist()**(See 10.)。

与 [FileMove](#)(See 16.16) 不同的是，复制文件覆盖自己通常会导致错误，即使打开了覆盖模式。

注意

FileCopy 只能复制文件。而要复制文件夹中的所有内容(所有文件和子文件夹)，参看下面的示例。要复制一个单独的文件夹(包括它的子文件夹)，使用 [FileCopyDir](#)(See 16.6) 。

即使错误发生，操作也会继续。

相关命令

[FileMove](#)(See 16.16), [FileCopyDir](#)(See 16.6), [FileMoveDir](#)(See 16.17), [FileDelete](#)(See 16.9)

示例

```
FileCopy, C:\My Documents\List1.txt, D:\Main Backup\ ; 复制且保留原始文件名。
FileCopy, C:\My File.txt, C:\My File New.txt ; 通过提供一个新的名字来复制文件到同个文件夹中。
FileCopy, C:\Folder1\*.txt, D:\New Folder\*.b kp ; 复制到新的位置并改变扩展名。
```

```
; 下面的例子复制一个文件夹下的所有文件和文件夹到一个不同的文件夹中:
ErrorCount := CopyFilesAndFolders("C:\My Folder\*.*", "D:\Folder to receive all files &
folders")
if ErrorCount <> 0
MsgBox %ErrorCount% 个文件/文件夹不能被复制。
CopyFilesAndFolders(SourcePattern, DestinationFolder, DoOverwrite = false)
; 复制匹配 SourcePattern 的所有文件和文件夹到名为 DestinationFolder 的文件夹
; 并返回不能被复制的文件/文件夹的数量。
{
; 首先复制所有的文件(非文件夹):
FileCopy, %SourcePattern%, %DestinationFolder%, %DoOverwrite%
ErrorCount := ErrorLevel
; 现在复制所有的文件夹:
Loop, %SourcePattern%, 2 ; 2 表示“只获取文件夹”。
{
FileCopyDir, %A_LoopFileFullPath%, %DestinationFolder%\%A_LoopFileName%, %DoO
verwrite%
ErrorCount += ErrorLevel
if ErrorLevel ; 报告每一个有问题的文件夹名称。
 MsgBox 不能复制 %A_LoopFileFullPath% 到 %DestinationFolder%.
}
return ErrorCount
}
```

翻译: Xianggee 修正: 天堂之门 menk33@163.com 2008年8月7日

16.6 FileCopyDir

复制一个文件夹连同它的所有文件和子文件夹(和 `xcopy` 相似)。

`FileCopyDir, Source, Dest [, Flag]`

参数

Source	源目录的名称(末尾不带反斜线), 如果绝对路径未指定将被假设在 %A_WorkingDir%(See 9.) 。例如: C:\My Folder
Dest	目标目录的名称(末尾不带反斜线), 如果绝对路径未指定将被假设在 %A_WorkingDir%(See 9.) 。例如: C:\Copy of My Folder
Flag	<p>(可选)此 flag(标记)决定是否覆盖已经存在的文件:</p> <p>0(默认): 不覆盖存在的文件。如果 <i>Dest</i> 有一个文件或目录存在, 复制操作将不生效并失败。</p> <p>1: 覆盖存在的文件。不过, 任何 <i>Dest</i> 中的文件或子文件夹没在 <i>Source</i> 有对应的将不会被删除。</p> <p>此参数可以是一个 expression(See 9.)(表达式)甚至是一个算得的 <code>true</code>(真)或 <code>false</code>(假)(因为真或假在内部被存为 1 和 0)。</p>

ErrorLevel

如果发生了一个问题 [ErrorLevel\(See 30.8\)](#) 会被设为 1, 此外为 0。

但是, 如果源目录包含网页组成的 `PageName.htm` 和相应的 `PageName_files` 目录时, 即使成功, 也会返回 1。

注意

如果目标目录结构不存在, 可能的话它将被创建。

由于操作将 `recursively`(递归地) 复制一个文件夹连同它的所有文件和子文件夹, 所以复制一个文件夹到它自身内部的某处目的地的结果未被定义。要绕弯解决这种情况, 先复制它到它自身外部的一个目的地, 然后使用 [FileMoveDir\(See 16.17\)](#) 来移动复制的文件夹到需要的位置。

`FileCopyDir` 复制单个文件夹。要代替它复制一个文件夹的内容(它的所有文件和子文件夹), 请看 [FileCopy\(See 16.5\)](#) 的示例部分。

相关命令

[FileMoveDir\(See 16.17\)](#), [FileCopy\(See 16.5\)](#), [FileMove\(See 16.16\)](#), [FileDelete\(See 16.9\)](#), [file-loops\(See 16.31\)](#), [FileSelectFolder\(See 16.24\)](#), [SplitPath\(See 16.34\)](#)

示例

```
FileCopyDir, C:\My Folder, C:\Copy of My Folder

; 例子 #2: 一个运行的脚本提示你去复制一个文件夹。
FileSelectFolder, SourceFolder, , 3, Select the folder to copy
if SourceFolder =
    return
    ; 否则, 继续。
FileSelectFolder, TargetFolder, , 3, Select the folder IN WHICH to create the duplicate
folder.
if TargetFolder =
    return
    ; 否则, 继续。
MsgBox, 4, , A copy of the folder "%SourceFolder%" will be put into "%TargetFolder%".
Continue?
If MsgBox, No
    return
SplitPath, SourceFolder, SourceFolderName ; 仅从它的完全路径提取文件夹名称。
FileCopyDir, %SourceFolder%, %TargetFolder%\%SourceFolderName%
if ErrorLevel
    MsgBox 文件夹没能复制, 也许因为在 "%TargetFolder%" 里已经存在同名文件夹。
    return
```

翻译：天堂之门 menk33@163.com 2008 年 8 月 7 日

16.7 FileCreateDir

创建一个目录/文件夹。

`FileCreateDir, DirName`

参数

DirName	要创建的目录名, 如果绝对路径未指定将被假设在 %A_WorkingDir% (See 9.)。
---------	--

`ErrorLevel`

如果发生了一个问题 `ErrorLevel`(See 30.8) 会 被设为 1 , 此外为 0 。

注意

如果在 `DirName` 中给出的所有父目录不存在, 此命令也将创建它们。

相关命令

[FileRemoveDir](#)(See 16.22)

示例

```
FileCreateDir, C:\Test1\My Images\Folder2
```

翻译: 天堂之门 menk33@163.com 2008 年 8 月 7 日

16.8 FileCreateShortcut

创建快捷方式文件(.lnk)。

`FileCreateShortcut, Target, LinkFile [, WorkingDir, Args, Description, IconFile, ShortcutKey, IconNumber, RunState]`

参数

<code>Target</code>	快捷方式所指向的文件名，该参数应包含绝对路径，除非该文件被集成到系统中(如 <code>NotePad.exe</code>)。在快捷方式被创建时，所指向的文件不一定要存在；换言之，指向无效目标的快捷方式可以被创建。
<code>LinkFile</code>	要创建的快捷方式文件名，如果未指定绝对路径则假设在 <code>%A_WorkingDir%</code> (See 9.) 目录下。确保该参数包含 <code>.lnk</code> 扩展名。如果该文件已存在，它将被覆盖。
<code>WorkingDir</code>	启动快捷方式时 <code>Target</code> 的当前工作目录。如果留空或省略此参数，快捷方式的“开始位置”为空，当运行该快捷方式时，系统将指定默认的工作目录。
<code>Args</code>	启动快捷方式时，传递给 <code>Target</code> 的参数。使用空格分隔多个参数。如果一个参数包含空格，使用双引号将这参数括住。
<code>Description</code>	对快捷方式的描述(操作系统用来显示一个提示信息，等等)。
<code>IconFile</code>	被 <code>LinkFile</code> 显示的图标的完整路径及文件名。该参数必须是一个 <code>ico</code> 文件或者是 <code>EXE</code> 或 <code>DLL</code> 文件的第一个图标。
<code>ShortcutKey</code>	<p>单个字母、数字或者 按键列表(See 7.) 中的单个按键名称(可能不支持鼠标按键和其它不标准的按键)。不要使用修饰键符号。目前，所有的快捷键被创建为 <code>CTRL+ALT</code> 快捷方式。例如，如果给该参数指定字母 <code>B</code>，快捷键会是 <code>CTRL-ALT-B</code>。</p> <p>对于 <code>Windows 9x</code>：需要重启才能使快捷键生效。或者你可以打开快捷方式的属性对话框，重新设定快捷键来使它立即生效。</p>
<code>IconNumber</code>	要使用 <code>IconFile</code> 里除了首个图标外的其他图标，在这里指定编号(可以是 表达式 (See 9.))。例如， <code>2</code> 表示第二个图标。
<code>RunState</code>	<p>要最大化或最小化启动 <code>Target</code>，指定下面的一个数字：</p> <p>1 - 正常(默认) 3 - 最大化</p>

7 - 最小化

ErrorLevel

出错则将 [ErrorLevel](#)(See 30.8) 设为 1, 否则为 0 。

注意

如果目标文件位于系统的 PATH 环境变量所列出的某个文件夹里, 参数 Target 可能不需要包含路径。

最近创建的快捷方式的只有在桌面上或者在开始菜单的某处, *ShortcutKey* 才会生效。如果你选择的 *ShortcutKey* 已在使用, 那么你新建的快捷方式有优先权。

下面的例子是一个可选用的方法来创建指向某个 URL 的快捷方式, 它创建了一个特殊的 URL 快捷方式。改变前两个参数来满足你的偏好:

[IniWrite](#)(See 16.30), http://www.google.com, C:\My Shortcut.url, InternetShortcut, URL

下面的命令可随意地添加进来给上面的快捷方式指定一个图标:

[IniWrite](#)(See 16.30), <IconFile>, C:\My Shortcut.url, InternetShortcut, IconFile

[IniWrite](#)(See 16.30), 0, C:\My Shortcut.url, InternetShortcut, IconIndex

上面的命令中, 使用图标编号替换 0(0 用来指定首个图标)并使用 URL、EXE、DLL 或者 ICO 文件替换 <IconFile> 。例如: C:\Icons.dll, C:\App.exe, http://www.somedomain.com/ShortcutIcon.ico

操作系统会把上面创建的 .URL 文件看作一个真的快捷方式, 即使它是一个纯文本文件而不是 .LNK 文件。

相关命令

[FileGetShortcut](#)(See 16.11), [FileAppend](#)(See 16.4)

示例

```
; 最后一个参数中的字母 "i" 将快捷键设成 Ctrl-Alt-I :
```

```
FileCreateShortcut, Notepad.exe, %A_Desktop%\My Shortcut.lnk, C:\,
"%A_ScriptFullPath%", My Description, C:\My Icon.ico, i
```

翻译: lwjeee 修正: 天堂之门 menk33@163.com 2008 年 8 月 22 日

16.9 FileDelete

删除一个或多个文件。

[FileDelete](#), [FilePattern](#)

参数

FilePattern	<p>单个文件的名称或一个像 C:\Temp*.tmp 这样的通配符样式。如果绝对路径没被指定, <i>FilePattern</i> 会被假设在 %A_WorkingDir%(See 9.)。</p> <p>要移除一整个文件夹, 与所有它的子文件夹和文件一起, 请用 FileRemoveDir(See 16.22)。</p>
--------------------	---

ErrorLevel

[ErrorLevel\(See 30.8\)](#) 被设置为删除失败的文件数目(如果有的话)或反之为 0。删除像 *.tmp 的通配符样式被视为成功, 即使它没有匹配任何文件; 因此 ErrorLevel 被设置为 0。

说明

要删除一个只读文件, 先要移除只读属性。例如: [FileSetAttrib, -R, C:\My File.txt\(See 16.25\)](#)

相关命令

[FileRecycle\(See 16.20\)](#), [FileRemoveDir\(See 16.22\)](#), [FileCopy\(See 16.5\)](#), [FileMove\(See 16.16\)](#)

范例

```
FileDelete, C:\temp files\*.tmp
```

翻译: 天堂之门 menk33@163.com 2008 年 5 月 21 日

16.10 FileGetAttrib

报告一个文件或文件夹是否为只读、隐藏等等。

FileGetAttrib, OutputVar [, Filename]
[AttributeString\(See 16.10\) := FileExist\(See 10.\)\(FilePattern\)](#)

参数

OutputVar	储存获取的文本的变量名称。
Filename	目标文件的名称, 如果绝对路径未指定将被假设在 %A_WorkingDir%(See 9.)。如果省略, 装在 File-Loop(See 16.31) 最内层的当前文件将被代为使用。

ErrorLevel

如果发生了一个问题 [ErrorLevel\(See 30.8\)](#) 会被设为 1 , 此外为 0 。

注意

返回的字串将包含一个在字符串 "RASHNDOCT" 中的字母子集:

R = READONLY (只读)
A = ARCHIVE (存档)
S = SYSTEM (系统)
H = HIDDEN (隐藏)
N = NORMAL (普通)
D = DIRECTORY (目录)
O = OFFLINE (离线)
C = COMPRESSED (压缩)
T = TEMPORARY (临时)

要检查特定的属性是否在获得的字串里存在, 参照此例:

```
FileGetAttrib, Attributes, C:\My File.txt
IfInString, Attributes, H
MsgBox 文件是隐藏的。
```

相关提示, 要获取一个文件的 8.3 短名格式, 参照此例:

```
Loop(See 16.31), C:\My Documents\Address List.txt
ShortPathName = %A_LoopFileShortPath% ; 将产生一些路径类似于
C:\MYDOCU~1\ADDRES~1.txt
```

一个相似的手法能用来得到 8.3 短名格式的长文件名。

相关命令

[FileExist\(\)](#)(See 10.), [FileSetAttrib](#)(See 16.25), [FileGetTime](#)(See 16.13), [FileSetTime](#)(See 16.26), [FileGetSize](#)(See 16.12), [FileGetVersion](#)(See 16.14), [File-loop](#)(See 16.31)

示例

```
FileGetAttrib, OutputVar, C:\New Folder
```

翻译: 天堂之门 menk33@163.com 2008 年 8 月 7 日

16.11 FileGetShortcut

获取某个快捷方式(.lnk)文件的信息, 比如它的目标文件。

FileGetShortcut, LinkFile [, OutTarget, OutDir, OutArgs, OutDescription, OutIcon, OutIconNum, OutRunState]

参数

LinkFile	待分析的快捷方式文件名, 如果未指定绝对路径, 则假定文件在 %A_WorkingDir% (See
----------	--

	9.) 目录下。须确认文件包括 .lnk 扩展名。
OutTarget	用于保存快捷方式的目标的变量名(不包括任何可能带有的参数)。比如： C:\WINDOWS\system32\notepad.exe
OutDir	用于保存快捷方式工作目录的变量名。比如：C:\My Documents。如果字符串中存在环境变量(如 %WinDir%)，一种解决方法是用 StringReplace(See 27.20) 。比如： <i>StringReplace, OutDir, OutDir, '%WinDir%', %A_WinDir(See 9.)%</i>
OutArgs	用于保存快捷方式参数的变量名(若无则为空)。
OutDescription	用于保存快捷方式说明的变量名(若无则为空)。
OutIcon	用于保存快捷方式图标文件名的变量名(若无则为空)。
OutIconNum	用于保存快捷方式的图标在图标文件中的序号的变量名(若无则为空)。该值通常为 1，表示第一个图标。
OutRunState	用于保存快捷方式启动时的初始状态的变量名，其值为以下数字之一： 1: 标准 3: 最大化 7: 最小化

ErrorLevel

如果有问题，比如 **LinkFile** 不存在，则所有输出变量被置为空，同时 [ErrorLevel\(See 30.8\)](#) 被置为 1。否则 **ErrorLevel** 为 0。

注意

如果不需输出变量提供的相应信息，可将其省略。

相关命令

[FileCreateShortcut\(See 16.8\)](#), [SplitPath\(See 16.34\)](#)

示例

```
FileSelectFile, file, 32,, 选择待分析的快捷方式。, 快捷方式 (*.lnk)

if file =
    return

FileGetShortcut, %file%, OutTarget, OutDir, OutArgs, OutDesc, OutIcon, OutIconNum,
OutRunState

MsgBox %OutTarget%`n%OutDir%`n%OutArgs%`n%OutDesc%`n%OutIcon%`n%Out
IconNum%`n%OutRunState%
```

翻译：甲壳虫<jdchenjian@gmail.com>

16.12 FileGetSize

返回文件的大小。

`FileGetSize, OutputVar [, Filename, Units]`

参数

<code>OutputVar</code>	存储返回值(该文件的大小)的变量名(其结果进行四舍五入)。
<code>Filename</code>	目标文件的文件名，如果绝对路径没有指定的话假定在 %A_WorkingDir%(工作目录)(See 9.) 中。如果忽略不写，当前 File-Loop(See 16.31) 中最深处的文件将被替代使用。
<code>Units</code>	如果这个参数出现，那么它将使得最终返回的结果遵循以下的方案(否则将以字节的形式显示): <code>K</code> = 千字节 <code>M</code> = 兆字节

ErrorLevel

如果有问题的话 [ErrorLevel\(See 30.8\)](#) 被设置为 1，否则设置为 0。

注意

支持大于 4G 字节的文件(甚至就算参数 `Units` 是字节也可以)。

如果目标文件是一个目录的话，不论操作系统把它当作什么(可能是 0)都会返回它的大小。

如果要计算文件夹的大小，像下面这个例子一样计算里面所有的文件：

```
SetBatchLines, -1 ; 使得操作以最大的速度运行。
FolderSize = 0
FileSelectFolder, WhichFolder ; 让用户选择一个目录。
Loop, %WhichFolder%\*.*,, 1
    FolderSize += %A_LoopFileSize%
MsgBox Size of %WhichFolder% is %FolderSize% bytes.
```

相关命令

[FileGetAttrib\(See 16.10\)](#), [FileSetAttrib\(See 16.25\)](#), [FileGetTime\(See 16.13\)](#), [FileSetTime\(See 16.26\)](#), [FileGetVersion\(See 16.14\)\(See 16.14\)](#), [File-loop\(See 16.31\)](#)

示例

```
FileGetSize, size, C:\My Documents\test.doc ; 以字节的形式返回 Test.doc 文件的大小
```

```
FileGetSize, size, C:\My Documents\test.doc, K ; 以千字节的形式返回 Test.doc 文件的大小
```

翻译: hsudataalks

16.13 FileGetTime

返回指定文件或文件夹的时间戳(时间和日期)信息。

`FileGetTime, OutputVar [, Filename, WhichTime]`

参数

<code>OutputVar</code>	以 YYYYMMDDHH24MISS(See 16.26) 格式存储返回的时间日期的变量名。时间就是本地时间，不是 UTC/GMT(格林尼治标准时间)。
<code>Filename</code>	[可选参数] 目标文件或文件夹的名字，如果绝对路径没有指定，就假设在 %A_WorkingDir%(工作目录)(See 9.) 中。如果忽略不写，当前 File-Loop(See 16.31) 中最深的文件将被代替使用。
<code>WhichTime</code>	[可选参数] 指定时间戳的返回类型： M = 修改时间(参数被省略时的默认值)。 C = 创建时间。 A = 访问时间。

ErrorLevel

如果有问题的话 [ErrorLevel\(See 30.8\)](#) 被设置为 1，否则设置为 0。

注意

在操作系统上，一个文件的访问时间在 NT 上要比在 WIN9x 上更加精确。

参见 [YYYYMMDDHH24MISS\(See 16.26\)](#) 来获取日期和时间的解释。

相关命令

[FileSetTime\(See 16.26\)](#), [FormatTime\(See 27.1\)](#), [If var is \[not\] type\(See 21.8\)](#),
[FileGetAttrib\(See 16.10\)](#), [FileSetAttrib\(See 16.25\)](#), [FileGetSize\(See 16.12\)](#),
[FileGetVersion\(See 16.14\)\(See 16.14\)](#), [File-loop\(See 16.31\)](#), [EnvAdd\(See 21.1\)](#), [EnvSub\(See 21.4\)](#)

示例

```
FileGetTime, OutputVar, C:\My Documents\test.doc ; 默认地返回修改时间。
```

```
FileGetTime, OutputVar, C:\My Documents\test.doc, C ; 返回创建时间。
```

翻译: hsudataalks

16.14 FileGetVersion

返回指定“文件”（通常是指可执行文件）的版本信息。

`FileVersion, OutputVar [, Filename]`

参数

OutputVar	存储返回值(数字或字符串)的变量名。
Filename	[可选参数] 目标文件或文件夹的名字，如果绝对路径没有指定，就假设在 %A_WorkingDir% (See 9.) 中。如果忽略不写，当前 File-Loop (See 16.31) 中最深的文件将被代替使用。

ErrorLevel

如果有问题的话 [ErrorLevel](#)(See 30.8) 被设置为 1，否则设置为 0。

注意

大部分不可执行的文件(甚至有些EXE文件)没有版本信息，所以参数OutputVar在这些例子中便成了空值。

相关命令

[FileGetAttrib](#)(See 16.10), [FileSetAttrib](#)(See 16.25), [FileGetTime](#)(See 16.13), [FileSetTime](#)(See 16.26), [FileGetSize](#)(See 16.12)(See 16.14), [File-loop](#)(See 16.31)

示例

```
FileVersion, version, C:\My Application.exe
```

```
FileVersion, version, %A_ProgramFiles%\AutoHotkey\AutoHotkey.exe
```

翻译: hsudataalks

16.15 FileInstall

包含并装入指定文件到 [compiled version](#)(See 8.) 的脚本程序中。

`FileInstall, Source, Dest [, Flag]`

参数

Source	<p>要装入到编译程序中的文件的路径。如果没有指定绝对路径，文件被假定为在脚本自己的路径里。</p> <p>文件名不能包含双引号，变量引用(比如%A_ProgramFiles%)，或者通配符。任何特殊字符比如原义百分比符号和逗号都必须为 escaped(See 29.5) (就如所有命令中的参数一样)。最终，这个参数必须在 <code>FileInstall</code> 命令的右边被列出来(不能有 continuation line(See 8.) 在它下面)。</p>
Dest	<p>当 <code>Source</code> 从 EXE 中提取出来时，这参数就是要被创建的文件名称。如果绝对路径没有指定，就假设在 %A_WorkingDir%(See 9.) 中。目标目录必须已存在。与参数 <code>Source</code> 不一样的是，变量引用可以被使用。</p>
Flag	<p>[可选参数] 此标志参数用以决定是否覆盖已存在的文件：</p> <p>0 = (默认) 不覆盖已存在的文件 1 = 覆盖已存在的文件</p> <p>这个参数可以成为一个 expression(See 9.)(表达式)，甚至能当作 <code>true</code> 或者 <code>False</code> 来使用(因为 <code>True</code> 和 <code>false</code> 其实是以 1 和 0 的形式存储的)。</p>

ErrorLevel

如果有问题的话 [ErrorLevel](#)(See 30.8) 被设置为 1，否则设置为 0。

注意

这个命令是为 [Ahk2Exe compiler](#)(See 8.) 而设置的允许你添加额外文件到最终的编译脚本的。随后，当编译脚本运行的时候，这些文件被提取到磁盘里。

文件是在脚本编译中被添加的。当编译脚本执行，并且执行到相同的"FileInstall"命令时，文件随后被提取到目标(Dest)里。

被添加的文件一般都会被压缩和加密。

如果这个命令出现在一段普通脚本(不是编译脚本)里，一个文件备份将代替出现，这将能帮助我们测试那些最终会被编译的脚本。

相关命令

[FileCopy](#)(See 16.5), [#Include](#)(See 17.1)

示例

```
FileInstall, C:\My Documents\My File.txt, %A_ProgramFiles%\My Application\Readme.txt,  
1
```

16.16 FileMove

移动或重命名一个或者多个文件。

`FileMove, SourcePattern, DestPattern [, Flag]`

参数

SourcePattern	单个文件的名称或一个通配符类型例如 <code>C:\Temp*.tmp</code> 。如果未指定绝对路径, <i>SourcePattern</i> 将假设在 <code>%A_WorkingDir%</code> (See 9.)。
DestPattern	目标文件的名称或通配符类型, 如果绝对路径未指定将假设在 <code>%A_WorkingDir%</code> (See 9.)。要执行一个简单的移动 -- 保留已存在的文件名 -- 仅指定文件夹名称像那些功能相同的例子显示的那样: <code>FileMove, C:*.txt, C:\My Folder</code> <code>FileMove, C:*.txt, C:\My Folder*.*</code>
Flag	<p>(可选参数) 此 <code>flag</code>(标识) 决定是否覆盖已存在的文件:</p> <p><code>0</code> = (默认) 不覆盖存在的文件 <code>1</code> = 覆盖存在的文件</p> <p>此参数可以是一个 <code>expression</code>(See 9.)(表 达式) 甚至是一个算得的 <code>true</code>(真) 或 <code>false</code>(假) (因为真或假在内部被存为 <code>1</code> 和 <code>0</code>)。</p>

ErrorLevel

`ErrorLevel`(See 30.8) 被设为由于出错而不能被移动的文件数量, 或相反是 `0`。不过, 如果源文件是单个文件(无通配符)并且文件不存在, `ErrorLevel` 将设为 `0`。要侦测这种情况, 在移动源文件前对它使用 `IfExist`(See 16.27) 或 `FileExist()`(See 10.)。

与 `FileCopy`(See 16.5) 不同的是, 移动一个文件覆盖它自己总是视为成功的, 即使覆盖模式没有启用。

注意

`FileMove` 仅移动文件。要代替它移动一个文件夹的内容(它的所有文件和子文件夹), 请看下面的示例部分。要移动或重命名单个文件夹, 使用 `FileMoveDir`(See 16.17)。

即使错误发生, 操作也将继续。

尽管此命令能够移动文件到一个不同的 `volume`(卷), 但此操作将比在同一个卷上移动花更长的时间。这是因为同一个卷移动类似于重命名, 因此更加迅速。

相关命令

`FileCopy`(See 16.5), `FileCopyDir`(See 16.6), `FileMoveDir`(See 16.17), `FileDelete`(See 16.9)

示例

```
FileMove, C:\My Documents\List1.txt, D:\Main Backup\ ; 不重命名地移动文件。
FileMove, C:\File Before.txt, C:\File After.txt ; 重命名单个文件。
FileMove, C:\Folder1\*.txt, D:\New Folder\*.bkp ; 移动并重命名文件为一个新的扩展名。
```

; 下面的例子移动一个文件夹内的所有文件和文件夹到一个不同的文件夹:

```
ErrorCount := MoveFilesAndFolders("C:\My Folder\*.*", "D:\Folder to receive all files & folders")
if ErrorCount <> 0
    MsgBox %ErrorCount% 个文件/文件夹不能被移动。
MoveFilesAndFolders(SourcePattern, DestinationFolder, DoOverwrite = false)
; 移动所有匹配 SourcePattern 的文件和文件夹到名为 DestinationFolder 的文件夹
; 并返回不能被移动的文件/文件夹的数量。此函数需要 v1.0.38+
; 因为它使用了 FileMoveDir 的模式 2。
{
if DoOverwrite = 1
    DoOverwrite = 2 ; 请看 FileMoveDir(See 16.17) 关于模式 2 对模式 1 的描述。
; 首先移动所有文件(而不是文件夹):
FileMove, %SourcePattern%, %DestinationFolder%, %DoOverwrite%
ErrorCount := ErrorLevel
; 现在移动所有的文件夹:
Loop, %SourcePattern%, 2 ; 2 表示 "仅获取文件夹"。
{
    FileMoveDir, %A_LoopFilePath%, %DestinationFolder%\%A_LoopFileName%, %DoOverwrite%
    ErrorCount += ErrorLevel
    if ErrorLevel ; 报告每一个有问题的文件夹名称。
    MsgBox 不能移动 %A_LoopFilePath% 到 %DestinationFolder% 。
}
return ErrorCount
}
```

翻译: 天堂之门 menk33@163.com 2008 年 8 月 8 日

16.17 FileMoveDir

移动一个文件夹连同它的所有文件和子文件夹。也能重命名一个文件夹。

FileMoveDir, Source, Dest [, Flag]

参数

Source	源目录的名称(末尾不带反斜线), 如果绝对路径未指定将被假设
--------	--------------------------------

	在 %A_WorkingDir% (See 9.)。例如: C:\My Folder
Dest	目录的新路径和名称(末尾不带反斜线)，如果绝对路径未指定将被假设在 %A_WorkingDir% (See 9.)。例如: D:\My Folder。 注意: Dest 是目录移动后的实际路径和名称；它不是 Source 移入的目录(除了下面提到的已知限制)。
Flag	<p>(可选参数) 指定下列单个字符中的一个：</p> <p>0 (默认): 不覆盖存在的文件。如果 Dest 已经作为一个文件或目录存在，移动操作将失败。</p> <p>1: 覆盖存在的文件。不过，Dest 中任何不存在于 Source 的文件或子文件夹将不会被删除。已知限制: 如果 Dest 已经作为一个文件夹存在并且和 Source 在同一个 volume(卷)上，Source 将被移入 Dest 而不是覆盖它。要避免这种情况，请看下一个选项。</p> <p>2: 和上面的模式 1 相同，除了不存在那个限制。</p> <p>R: 重命名目录而不是移动它。虽然重命名目录通常和移动的效果一样，但如果你想要 "all or none" 的表现时，它会很有用；就是说，当 Source 或它的部分文件被锁定时(在使用)你不想移动操作仅部分成功。尽管这种方法不能把 Source 移动到另一个卷，但能把它移到它所在卷的其他任何目录。如果 Dest 已经作为一个文件或目录存在，移动操作将失败。</p>

ErrorLevel

如果发生了一个问题 [ErrorLevel](#)(See 30.8) 会被设为 1，此外为 0。

注意

[FileMoveDir](#) 移动单个文件夹到一个新的位置。要替代它移动一个文件夹的内容(它的所有文件和子文件夹)，请看 [FileMove](#)(See 16.16) 的示例部分。

如果 source 和 destination 在不同的卷或 UNC(通用命名规则) 路径，一个复制/删除操作将被执行而不是移动。

相关命令

[FileCopyDir](#)(See 16.6), [FileCopy](#)(See 16.5), [FileMove](#)(See 16.16), [FileDelete](#)(See 16.9),
[File-loops](#)(See 16.31), [FileSelectFolder](#)(See 16.24), [SplitPath](#)(See 16.34)

示例

```
FileMoveDir, C:\My Folder, D:\My Folder ; 移动到一个新的驱动器。
```

```
FileMoveDir, C:\My Folder, C:\My Folder (renamed), R ; 简单的重命名。
```

```
FileMoveDir, C:\My Folder, C:\New Location\My Folder, R ; 文件夹可以 "重命名至" 另一个目录，只要都在同一盘符下。
```

16.18 FileReadLine

读取文件的指定行，并将所得文本存储于[变量\(See 9.\)](#)中。

`FileReadLine, OutputVar, Filename, LineNum`

参数

<code>OutputVar</code>	变量(See 9.) 名，用以储存解读出的文本行。
<code>Filename</code>	文件名，若未指定绝对路径将被默认为 当前脚本运行路径(See 9.) (<code>%A_WorkingDir%</code>)。支持 Windows 和 Unix 文件格式，也就是说，行尾标识可以是回车换行符(`r`n)或单纯的换行符(`n)。
<code>LineNum</code>	行号，用以表示该行将被读取(1 是第一行，2 是第二行，等等)。可以是 表达式(See 9.) 。

ErrorLevel

[ErrorLevel\(See 30.8\)](#) 返回值为 1 时说明读取失败，反之，为 0 则表明已正常读取。

注意

一般说来，该命令应当仅被用于处理较小的文件，或用于处理文件中的某一特定单行。如果您要扫描并处理许多行（一行又一行）的话，请使用[文件读取循环\(See 16.32\)](#)（file-reading loop）以获得命令执行的最佳性能。如果您要将整个文件读取到变量中，请使用[文件读取\(See 16.19\)](#)（FileRead）命令。

虽然任何存在于行首和行尾的制表符及空格都会被一并记录到 `OutputVar` 变量中，但是，行尾的换行符(`n)是个例外。当[自动修剪\(See 22.3\)](#)（AutoTrim）开关处于开启状态时（默认），任何变量均可以通过运用“赋值于该变量自身”的这一技巧，来自动修剪掉它首尾的制表符及空格。举例来说，如果储存文本行的变量名为 `MyLine` 的话，修剪该行首尾的制表符及空格只需要在 `FileReadLine` 命令行后添加一句：`MyLine = %MyLine%`。

该命令不能用于读取超过六万五千五百三十四（65,534）字符的超长行。如果一行中的字符数逾越了这个极限，超限部分的字符将无法获取（在这种特殊情形下，应该取而代之的正确命令是[文件读取\(See 16.19\)](#)（FileRead）或[文件读取循环\(See 16.32\)](#)（file-reading loop））。

相关命令

[FileRead\(See 16.19\)](#), [FileAppend\(See 16.4\)](#), [File-reading loop\(See 16.32\)](#), [IniRead\(See 16.29\)](#)

示例

```
Loop
{
    FileReadLine, line, D:\我的文档\ContactList.txt, %A_Index%
    if ErrorLevel
        break
```

```

 MsgBox, 4,FileReadLine 官方示例程序 , 第%A_Index%行的内容为"%line%"。是否继续?
 IfMsgBox, No
 return
}
 MsgBox, 已到达文件结尾或已出错。
return

```

示例+ /poetbox

```

Loop ;Loop 引领的大括号内,各行行首均有缩进空格
{
;FileReadLine, line, D:\我的文档\ContactList.txt, %A_Index% ;原例程,重在演示绝对路径写法,需要自行创建 Contactlist.txt 以便测试
FileReadLine, line, %A_ScriptName%, %A_Index% ;为方便直接测试,改为读取脚本自身内容
line=%line% ;新增此行以演示去除首尾制表符及空格,可注释掉此行以观效果(此行行尾有空格)
if ErrorLevel
break
MsgBox, 4,FileReadLine 示例程序 , 第%A_Index%行的内容为"%line%"。是否继续?
IfMsgBox, No
return
}
 MsgBox, 已到达文件结尾或已出错。
return

```

16.19 FileRead

读取一个文件的内容给一个 [变量\(See 9.\)](#)。

`FileRead, OutputVar, Filename`

参数

<code>OutputVar</code>	用来存储获取的数据的 变量(See 9.) 名称。如果遇上一个例如文件正在“使用”或不存在的问题, <code>OutputVar</code> 将被设为空(这种情况下 ErrorLevel(See 30.8) 被设为 1)。如果 <code>Filename</code> 是一个空白文件, 它也将被设为空(这种情况下 ErrorLevel 被设为 0)。
<code>Filename</code>	<p>要读取的文件名称, 如果绝对路径未指定将被假设在 %A_WorkingDir%(See 9.)。</p> <p>选项: 零个或多个下列字串也可以直接在文件名称前面提供。将每个选项和其下一个用单个空格或 tab 分隔开来。例如: <code>*t *m5000 C:\Log Files\200601.txt</code></p> <p>*c: 读取一个保存了 ClipboardAll(See 30.6) 的文件。当 *c 提供后, 所有其他选项将被忽略。</p> <p>*m1024 [v1.0.43.03+]: 如果此选项被省略, 将读取整个文件除非它大于 1 GB, 这种情况下它将根本不会被读取。否则要读取部分的话, 将字节在 1 到 1073741824 (1 GB)</p>

之间的十进制或十六进制数字来替换掉 1024。如果文件大于 1 GB，将仅仅读取它的前面部分。注意：此选项可能导致在最后一行以一个光秃秃的回车(`r)而不是 `r`n 来结束。

***t:** 用换行(`n)替换出现的部分/全部回车和换行(`r`n)。不过，这种转换减低了性能并且往往不必要。例如，包含 `r`n 的文本已经是用来添加到一个[图形编辑控件](#)(See 27.20)的正确的格式。类似地，[FileAppend](#)(See 16.4) 在它打开一个新文件时探测到 `r`n 的存在；它知道将 `r`n 以现状写入而不是将其转换为 `r`r`n。最后，后面例子中的[解析循环](#)(See 17.14)将正确地运转而不管每行是否以 `r`n 或者仅以 `n 结尾：Loop, parse, MyFileContents, `n, `r

ErrorLevel

如果读取成功 [ErrorLevel](#)(See 30.8) 设为 0。如果遇到问题它将被设为 1，例如：1) 文件不存在；2) 文件被锁定或无法访问；3) 操作系统缺乏足够的记忆存储来读取文件。

注意

当要读取一个文件的大部分或所有内容到记忆存储中时，[FileRead](#) 比使用 [file-reading loop](#)(See 16.32) (文件读取循环) 执行地更好。

一个文件的大小超过 1 GB 将导致 [ErrorLevel](#)(See 30.8) 被设为 1 并且 [OutputVar](#) 成为空，除非提供 ***m** 选项，这种情况下读取文件的前面部分。

[FileRead](#) 不服从 [#MaxMem](#)(See 29.15) 命令。如果担心使用了太多的记忆存储，预先用 [FileGetSize](#)(See 16.12) 来检查文件的大小。

如果指定的文件包含了任何二进制零(它决不会出现在一个正确的文本文件中)，那么仅仅在二进制零前面的文本会被 AutoHotkey 命令和函数“看到”。不过，整个内容仍然提交给 [OutputVar](#) 并且能被高级的方法访问像 [NumGet\(\)](#)(See 10.) 和 [地址运算符\(&\)](#)(See 9.)；例如：*(&OutputVar + 1000)

和 [file-reading loops](#)(See 16.32)，[FileReadLine](#)(See 16.18) 不同，[FileRead](#) 并不认为字符 Control-Z (0x1A) 是文件的结尾的标志。所以对于任何的 Control-Z，即使出现在文件的最末尾，也会被照常读入。

[FileRead](#) 能被用来快速地整理一个文件的内容像下面的例子那样：

```
FileRead, Contents, C:\Address List.txt
if not ErrorLevel ;成功载入。
{
    Sort(See 27.12), Contents
    FileDelete, C:\Address List (alphabetical).txt
    FileAppend, %Contents%, C:\Address List (alphabetical).txt
    Contents = ;释放记忆存储。
```

```
}
```

相关命令

[file-reading loop](#)(See 16.32), [FileReadLine](#)(See 16.18), [FileGetSize](#)(See 16.12),
[FileAppend](#)(See 16.4), [IniRead](#)(See 16.29), [Sort](#)(See 27.12), [UrlDownloadToFile](#)(See 22.24)

示例

```
FileRead, OutputVar, C:\My Documents\My File.txt
```

翻译：天堂之门 menk33@163.com 2008 年 11 月 8 日

16.20 FileRecycle

把指定的文件或目录放入回收站(如果可行的话)。

[FileRecycle](#), [FilePattern](#)

参数

FilePattern	<p>一个单独的文件名，或者有通配符的模式，比如 C:\Temp*.tmp. 如果没有指定绝对路径 参数 <i>FilePattern</i> 被假定为在 %A_WorkingDir%(See 9.) 中。</p> <p>为了回收整个目录，可以以名字后面不加反斜线符号方式来达到。</p>
-------------	--

ErrorLevel

如果有问题的话 [ErrorLevel](#)(See 30.8) 被设置为 1，否则设置为 0。

注意

无

相关命令

[FileRecycleEmpty](#)(See 16.21), [FileDelete](#)(See 16.9), [FileCopy](#)(See 16.5), [FileMove](#)(See 16.16)

示例

```
FileRecycle, C:\temp files\*.tmp
```

翻译：hsudatalks

16.21 FileRecycleEmpty

清空回收站。

FileRecycleEmpty [, DriveLetter]

参数

DriveLetter	[可选参数]若留空, 将清空所有驱动器中的回收站。否则, 指定一个驱动器, 比如 C:\
-------------	--

ErrorLevel

如果有问题的话 [ErrorLevel\(See 30.8\)](#) 被设置为 1, 否则设置为 0。

注意

这个命令需要微软的 IE4.0 或者更高的版本。

相关命令

[FileRecycle\(See 16.20\)](#), [FileDelete\(See 16.9\)](#), [FileCopy\(See 16.5\)](#), [FileMove\(See 16.16\)](#)

示例

```
FileRecycleEmpty, C:\
```

翻译:hsudatalks

16.22 FileRemoveDir

删除一个文件夹。

FileRemoveDir, DirName [, Recurse?]

参数

DirName	要删除的目录名称, 如果绝对路径未指定将被假设在 %A_WorkingDir%(See 9.) 。
Recurse?	0 (默认): 不移除包含在 <i>DirName</i> 中的文件和子目录。既然如此, 如果 <i>DirName</i> 内不是空的, 删除将不会执行并且 <i>ErrorLevel</i> 将被设为 1。 1: 移除所有文件和子目录(像 Windows 命令行的 "rmdir /S")。 此参数可以是一个 表达式(See 9.) 甚至是一个算得的 true 或 false (因为 true 和 false 在内部被存为 1 和 0)。

ErrorLevel

如果发生了一个问题 [ErrorLevel\(See 30.8\)](#) 会被设为 1, 此外为 0。

注意

无:-)

相关命令

[FileCreateDir\(See 16.7\)](#), [FileDelete\(See 16.9\)](#)

示例

```
FileRemoveDir, C:\Download Temp
```

```
FileRemoveDir, C:\Download Temp, 1
```

翻译：天堂之门 menk33@163.com 2008 年 11 月 7 日

16.23 FileSelectFile

显示一个允许用户来打开或保存文件的标准对话框。

`FileSelectFile,OutputVar[,Options,RootDir\Filename,Prompt,Filter]`

参数

OutputVar	用来保存用户选择的文件名的变量名称。如果用户取消了对话框(即不想选择一个文件), 该变量为空。
Options	<p>如果省略此项, 默认值为 0, 意味着以下选项都不生效。</p> <p>M: 多选。指定字母 M 表示允许用户通过按住 Shift、Control 键或者其它方式来选择多个文件。M 后面可以跟一个下面提到的数字(例如, M 和 M1 都是有效的)。要提取单个文件, 请看本页底部的例子。</p> <p>S: 保存按钮。指定字母 S 使对话框上总是包含保存按钮而不是打开按钮。S 后面可以跟一个下面提到的数字(或几个数的和。例如, S 和 S24 都是有效的)。</p> <p>即使没有 M 和 S, 下列数字也可以被使用。要同时使多个选项生效, 只要把数字加起来。例如, 要使用 8 和 16, 指定数字 24。</p> <p>1: 文件必须存在 2: 路径必须存在 8: 提示创建新文件 16: 提示覆盖文件 32 [v1.0.43.09+]: 选择的快捷方式(.lnk 文件)是它本身而不是它所指向的目标。此选项也防止了选择一个文件夹快捷方式会跳转到那个文件夹。</p> <p>如果用“提示覆盖文件”选项并且不用“提示创建新文件”选项的话, 对话框将包含保存按</p>

	钮而不是打开按钮。这个问题是 Windows 的毛病。
RootDir\Filename	<p>如果存在，此参数可包含下列的一项或全部：</p> <p>RootDir: 根(起始)目录，如果没有使用完全路径的话，假设为 %A_WorkingDir%(See 9.) 里的子文件夹。如果省略此参数或留空，起始目录会是默认值，而且随操作系统版本而有所不同(在 WinNT/2k/XP 及之后的版本中，它通常是用户上一次使用 FileSelectFile 时选择的目录)。在 v1.0.43.10 及之后的版本中，支持使用 CLSID(Windows 类标识符)(See 30.7)比如 ::{20d04fe0-3aea-1069-a2d8-08002b30309d}(即我的电脑)，这时在 CLSID 后面的任何子目录路径都必须以反斜线结尾(否则，最后一个反斜杠之后的字符串将被认为是默认的文件名，请看下面)。</p> <p>Filename: 一开始显示在对话框里的编辑框中的默认文件名。只有单独的文件名(不带路径)才会被显示。要确保对话框正确地显示，请不要提交非法的字符(比如 /< :)。</p> <p>示例：</p> <pre>C:\My Pictures\Default Image Name.gif ; RootDir 和 Filename 都提供了。 C:\My Pictures ; 只提供了 RootDir。 My Pictures ; 只提供了 RootDir，它相对于当前的工作目录。 My File ; 只提供了 Filename(但如果存在"My File"文件夹，则会被认为是 RootDir)。</pre>
Prompt	显示在窗口标题栏指示用户做什么的文本。如果省略或留空，它将默认为"Select File - %A_SCRIPTNAME%" (即当前脚本的名称)。
Filter	<p>指示哪些类型的文件会被对话框显示。</p> <p>例如：Documents (*.txt)</p> <p>例如：Audio (*.wav; *.mp2; *.mp3)</p> <p>如果省略，过滤器默认是 All Files (*.*)。另外在对话框的“文件类型”下拉列表中的 Text Documents (*.txt) 选项也是可用的。</p> <p>否则，过滤器将使用此参数指示的字符串，不过在对话框的“文件类型”下拉列表中仍然提供 All Files (*.*) 的选项。要在过滤器中包含多个文件扩展名，像上面的例子中那样用分号将它们隔开。</p>

ErrorLevel

如果用户没有选择文件(比如按了取消按钮)就关掉了对话框，[ErrorLevel](#)(See 30.8) 设为 1。如果系统拒绝显示对话框时(罕见)也将设为 1。之外设为 0。

注意

如果用户什么也没选(例如按了取消), *OutputVar* 为空。

如果没有使用多选, *OutputVar* 的值将设为用户选择的单个文件的完整路径和名称。

如果使用了 **M** 选项(多选), *OutputVar* 的值将设为一个列表, 其中除了最后一个外, 每个后面都跟了一个换行符(`n)。列表中的第一行是所有被选择的文件的路径(只有在此路径是根目录比如 C:\ 时才以反斜线结尾)。其它的条目是被选择的文件名(不带路径)。例如:

C:\My Documents\New Folder [这是下面所有文件所在的路径]

test1.txt [这些是单独的文件名: 没有路径]

test2.txt

.....等。

(本页底部的例子示范了如何一个个提取文件。)

在使用多选时, 被选择的文件名的总长度被限制在 64 KB。不过通常这足以容纳几千个文件了, 如果超出此限制的话, *OutputVar* 将为空。

GUI 窗口可以用 [Gui +OwnDialogs](#)(See 19.3) 来显示一个 modal(模态)文件选择对话框。模态对话框可以阻止用户在关掉对话框之前与 GUI 窗口的交互。

已知限制: 在文件选择对话框显示期间运行的[定时器](#)(See 17.21)会延迟用户在对话框中点击的效果直到定时器结束。要绕弯解决此限制, 应避免使用那些子程序会运行很长时间才结束的定时器, 或者在使用对话框时禁用所有的定时器:

[Thread](#)(See 22.22), NoTimers

[FileSelectFile](#), *OutputVar*

[Thread](#), NoTimers, false

作废的选项: 在 v1.0.25.06 及之后的版本中, 多选选项“4”已作废。可是为了与以前的脚本兼容, 这个选项仍然可用。具体是, 如果用户只选择了一个文件, *OutputVar* 会包括它的完整路径和文件名再加上一个换行符(`n)。如果用户选择了多个文件, 除了最后一行还是以换行符(`n)结尾外, 其它的格式都和上面描述的 **M** 选项一样。

相关命令

[FileSelectFolder](#)(See 16.24), [MsgBox](#)(See 19.11), [InputBox](#)(See 19.10), [ToolTip](#)(See 19.15),
[GUI](#)(See 19.3), [CLSID List](#)(See 30.7), [parsing loop](#)(See 17.14), [SplitPath](#)(See 16.34)

操作系统也提供了标准的对话框来提示用户选择字体、颜色或图标。这些对话框可以通过 [DIICall\(\)](#)(See 18.3) 命令来显示, 请看 www.autohotkey.com/forum/topic17230.html。

示例

```
FileSelectFile, SelectedFile, 3, , 打开一个文件 (*.txt; *.doc)
```

```
if SelectedFile =
```

```
    MsgBox, 用户什么也没有选择。
```

```
else
```

MsgBox, 用户选了下列文件: `n%SelectedFile%

[CLSID\(See 30.7\)](#) 示例:

```
FileSelectFile, OutputVar,, ::{645ff040-5081-101b-9f08-00aa002f954e} ;回收站。
```

;多选示例:

```
FileSelectFile, files, M3 ; M3 = 选择多个存在的文件。
```

```
if files =
{
    MsgBox, 用户按了取消。
    return
}
Loop, parse, files, `n
{
    if a_index = 1
        MsgBox, 选择的文件都包含在 %A_LoopField%。
    else
    {
        MsgBox, 4, , 下一个文件是 %A_LoopField%。继续?
        IfMsgBox, No, break
    }
}
return
```

翻译: 单菜子 修正: 天堂之门 menk33@163.com 2008年11月8日

16.24 FileSelectFolder

显示允许用户选择文件夹的标准对话框。

```
FileSelectFolder, OutputVar [, StartingFolder, Options, Prompt]
```

参数

OutputVar	<p>用来保存用户所选择的文件夹的变量名。如果用户取消了对话框，则该变量将被置为空（比如不想选择文件夹）。若用户选择了根目录（如 C:\），<i>OutputVar</i> 会以反斜杠结尾。如果不需这个反斜杠，可用下面的方法去掉它：</p> <pre>FileSelectFolder, Folder</pre> <pre>Folder := RegExReplace(Folder, "\\\\$") ; 若以反斜杠结尾，则去掉它。</pre>
StartingFolder	<p>若为空或省略，对话框选择的起始目录为用户的我的文档文件夹（或者可能是我的电脑）。CLSID 文件夹(See 30.7)，比如 ::{20d04fe0-3aea-1069-a2d8-08002b30309d}（也就是我的电脑）可以作为特殊文件被指定为起始目录。</p> <p>另外，该参数最常见的用法是一个星号通配符后紧跟着作为起始位置的驱动器或文件夹的绝对路径。比如 *C:\ 将把 C 盘作为起始选择位置。与此类似，*C:\My Folder 将把该文件夹作为起始选择位置。</p> <p>星号表示允许用户从起始目录向上导航（接近根目录）。如果没有星号，用户只能选择 <i>StartingFolder</i> 中的文件夹（或者 <i>StartingFolder</i> 本身）。省略星号的一个好处是用户不必点击目录前的加号，<i>StartingFolder</i> 一开始即以树状目录形式显示出来。</p> <p>若存在星号，向上导航可被限制到除桌面外的某个文件夹。在星号前加上最上级文件夹的绝对路径和一个空格或制表符，即可达到此目的。在下面的例子中，不允许用户导航到高于 C:\My Folder 的目录（但起始目录为 C:\My Folder\Projects）：</p> <p><i>C:\My Folder *C:\My Folder\Projects</i></p>
Options	<p>以下数字之一：</p> <p>0：以下所有选项无效（Windows 2000 除外，它可能始终显示“新建文件夹”按钮）。</p> <p>1（默认）：提供允许用户新建文件夹的按钮。然而，该按钮在 Windows 95/98/NT 中不会出现。</p> <p>上述数字加 2 提供允许用户输入文件夹名的编辑框。比如该参数为数值 3，则同时提供一个编辑框和一个“新建文件夹”按钮。</p> <p>上述数字加 4 可以忽略 BIF_NEWDIALOGSTYLE 属性。加 4 确保了 FileSelectFolder 可以在诸如 WinPE 或者 BartPE 之类的预安装环境中也能正常工作。然而，这可能会妨碍“新建文件夹”按钮的出现，至少在 Windows XP 中是这样。 ["4" 需要 1.0.48+ 版本]</p> <p>如果用户在编辑框中输入非法文件夹名，<i>OutputVar</i> 将会被置为在导航树中选择的文件夹名，而不是用户输入的名称，至少在 Windows XP 中是这样。</p> <p>该参数可以是个表达式(See 9.)。</p>
Prompt	显示在窗口中，告诉用户该做什么的文字。如果省略或空白，则默认为“选择文件夹”。

- %A_SCRIPTNAME%" (也就是当前脚本的名称)。

ErrorLevel

如果用户关闭对话框而未选择文件，则 [ErrorLevel\(See 30.8\)](#) 被置为 1（比如按了取消按钮）。如果系统拒绝显示对话框（罕见），该变量也会被置为 1。否则其值为 0。

注意

在 GUI 窗口中，可以用 [Gui +OwnDialogs\(See 19.3\)](#) 来显示一个选择文件夹的模式对话框。在模式对话框关闭之前，禁止用户与 GUI 窗口进行交互操作。

已知的限制：在显示 FileSelectFolder 对话框时启动的[定时器\(See 17.21\)](#)，将会使用户在对话框中的点击延迟生效，直到该定时器程序执行完毕。为避免这一问题，应避免使用子程序耗时很长的定时器，或者在显示对话框时禁止所有的定时器：

[Thread\(See 22.22\)](#), NoTimers

FileSelectFolder, OutputVar,, 3

Thread, NoTimers, false

相关命令

[FileSelectFile\(See 16.23\)](#), [MsgBox\(See 19.11\)](#), [InputBox\(See 19.10\)](#), [ToolTip\(See 19.15\)](#),
[GUI\(See 19.3\)](#), [CLSID List\(See 30.7\)](#), [FileCopyDir\(See 16.6\)](#), [FileMoveDir\(See 16.17\)](#),
[SplitPath\(See 16.34\)](#)

操作系统还提供提示用户选择字体、颜色或图标的标准对话框。这些对话框可以通过 [DIIICall\(\)\(See 18.3\)](#) 来显示，示例见 www.autohotkey.com/forum/topic17230.html。

示例

```
FileSelectFolder, OutputVar, , 3

if OutputVar =
    MsgBox, 您未选择文件夹。
else
    MsgBox, 您选择了文件夹 "%OutputVar%"。
```

[CLSID\(See 30.7\)](#) 示例：

```
FileSelectFolder, OutputVar, ::{20d04fe0-3aea-1069-a2d8-08002b30309d} ; 我的电脑。
```

翻译：甲壳虫<jdchenjian@gmail.com>

16.25 FileSetAttrib

更改一个或更多文件或者文件夹的属性。支持通配符。

`FileSetAttrib, Attributes [, FilePattern, OperateOnFolders?, Recurse?]`

参数

Attributes	要更改的属性(请看注意部分)
FilePattern	<p>单个文件或文件夹的名称，或是通配符类型比如 <code>C:\Temp*.tmp</code>。如果没有指定绝对路径，那么程序就假设 <code>FilePattern</code> 在 %A_WorkingDir%(See 9.) 里。</p> <p>如果省略的话，会用封装在 File-Loop(See 16.31) 最里面的当前文件来代替。</p>
OperateOnFolders?	<p>0 (缺省值) 不对文件夹进行操作(只操作文件)。</p> <p>1 所有与通配符类型匹配的文件和文件夹都会被操作。</p> <p>2 只操作文件夹(不操作文件)。</p> <p>注意：如果 <code>FilePattern</code> 是单个文件夹而不是通配符样式，那么此设置将总是被忽略。</p> <p>此参数可以是一个表达式(See 9.)。</p>
Recurse?	<p>0 (缺省值) 子文件夹不进行遍历。</p> <p>1 遍历子文件夹以便操作包含在它里面的与 <code>Filepattern</code> 匹配的文件和文件夹。所有的子文件夹都会被遍历，而不仅仅是那些名字与 <code>Filepattern</code> 匹配的。然而，文件和文件夹的完整路径超过 259 个字符的都会被跳过，就像它们根本不存在一样。通常这样的文件极少，因为操作系统不允许它们产生。</p> <p>这个参数可以是一个表达式(See 9.)。</p>

ErrorLevel

[ErrorLevel](#)(See 30.8) 被设为更改失败的文件个数，没有的话就是 0。

注意

`Attributes` 参数是由下面的这些运算符和属性字母组成。

运算符：

+	启用属性
-	取消属性
^	切换属性(设置为与现在相反的值)

属性字母:

R = READONLY (只读)

A = ARCHIVE (存档)

S = SYSTEM (系统)

H = HIDDEN (隐藏)

N = NORMAL (普通)

O = OFFLINE (离线)

T = TEMPORARY (临时)

注意: 通常, 文件的压缩状态是不能用此命令来更改的。(译注: 应该是指 NTFS 分区上的压缩属性)

相关命令

[FileGetAttrib](#)(See 16.10), [FileGetTime](#)(See 16.13), [FileSetTime](#)(See 16.26), [FileGetSize](#)(See 16.12), [FileGetVersion](#)(See 16.14), [File-loop](#)(See 16.31)

示例

```
FileSetAttrib, +RH, C:\MyFiles\*.* , 1 ; +RH 等同于 +R+H
```

```
FileSetAttrib, ^H, C:\MyFiles ; 切换文件夹的“隐藏”属性。
```

```
FileSetAttrib, -R+A, C:\New Text File.txt
```

```
FileSetAttrib, +A, C:\*.ini, , 1 ; 遍历 C 分区上所有的 .ini 文件。
```

翻译: handt 修正: 天堂之门 menk33@163.com 2008 年 10 月 20 日

16.26 FileSetTime

更改一个或多个文件、文件夹的时间戳。支持通配符。

`FileSetTime [, YYYYMMDDHH24MISS, FilePattern, WhichTime, OperateOnFolders?, Recurse?]`

参数

YYYYMMDDHH24MISS	如果为空或省略, 默认使用当前时间。否则则需要指定操作所使用的时间 (参见 Remarks for the format). 早于 1601 年以前的年份不被支持。
FilePattern	单个文件或文件夹的名称, 或通配符, 比如 C:\Temp*.tmp. 如果绝对路径没有给定, <i>FilePattern</i> 被假设为在 %A_WorkingDir% (See 9.) 中。 如果被省略, 按 File-Loop (See 16.31) 规则的最优先文件将被作为该选项的值。
WhichTime	那个时间戳将被设置: M = 修改时间 (如果该值为空或忽略, 这是默认值) C = 创建时间

	A = 访问时间
OperateOnFolders?	<p>0 (默认值) 不处理文件夹 (仅处理文件)。 1 所有与通配符匹配的文件和文件夹都被执行操作。 2 仅处理文件夹 (不处理文件)。</p> <p>注意: 如果 FilePattern 是单个文件夹名而不是通配符, 操作将总是被执行而与该设置无关。</p> <p>该参数可以是一个 表达式(See 9.)。</p>
Recurse?	<p>0 (默认) 不处理子目录。 1 递归的处理所有匹配 FilePattern 的子目录及其中文件。所有子目录都将被递归的打开, 而不仅仅是那些匹配 FilePattern 的子目录。但是, 整个路径名长于 259 个字符的文件和文件夹将被认为不存在。这些文件是罕见的, 因为通常情况下操作系统不允许创建他们。</p> <p>该参数可以是一个 表达式(See 9.)。</p>

ErrorLevel

ErrorLevel(See 30.8) 被设置为 没有 被修改的文件个数, 否则则为 0。如果设置的时间戳是无效的, 或者 **FilePattern** 匹配的值为空, **ErrorLevel** 也将被置为 1。

注意

在 Windows 95/98/ME 下, 不支持修改文件夹的时间戳。这样的尝试将被忽略。

一个在 NTFS 分区上文件的访问时间可能不会像在 FAT16 & FAT32 分区上那样准确。

YYYYMMDDHH24MISS 代表的意义分别是:

YYYY	四位数的年份
MM	两位数的月份 (01-12)
DD	两位数的天数 (01-31)
HH24	两位数的小时数 (00-23). 例如, 09 表示上午 9 点, 而 21 表示下午 9 点。
MI	两位数的分钟数 (00-59)
SS	两位数的秒数 (00-59)

如果只给定部分的 YYYYMMDDHH24MISS (例如 200403), 其他的未给定的值将使用如下值代替:

MM: Month 01

DD: Day 01

HH24: Hour 00

MI: Minute 00

SS: Second 00

内置变量 [A_Now](#)(See 9.) 包含如上格式的当前时间。相似地, [A_NowUTC](#)(See 9.) 包含当前的 UTC 时间。

注意: 日期时间变量可以使用 [EnvAdd](#)(See 21.1) 和 [EnvSub](#)(See 21.4) 比较和加减。同样, 除非两时间有相同的字符串长度, 最好不要使用 **greater-than** 或者 **less-than** 来比较时间。这是因为它们将作为数来比较; 例如, 20040201 始终在数值上小于 (但是实际上大于) 200401010533。所以应该使用 [EnvSub](#)(See 21.4) 来确定一组时间谁正谁负。

相关命令

[FileGetTime](#)(See 16.13), [FileGetAttrib](#)(See 16.10), [FileSetAttrib](#)(See 16.25), [FileGetSize](#)(See 16.12), [FileGetVersion](#)(See 16.14), [FormatTime](#)(See 27.1), [File-loop](#)(See 16.31), [EnvAdd](#)(See 21.1), [EnvSub](#)(See 21.4)

示例

```
; 将所有匹配的文件的修改时间设置为当前时间:
```

```
FileSetTime, , C:\temp\*.txt
```

```
; 设置修改时间 (时间将为午夜):
```

```
FileSetTime, 20040122, C:\My Documents\test.doc
```

```
; 设置创建时间。 时间将设置为 4:55pm:
```

```
FileSetTime, 200401221655, C:\My Documents\test.doc, C
```

```
; 修改匹配一种格式的所有文件的修改时间。
```

```
; 由于最后一个参数所有匹配的文件夹也将被修改:
```

```
FileSetTime, 20040122165500, C:\Temp\*.*, M, 1
```

16.27 IfExist/IfNotExist

检查文件或文件夹的存在。

[IfExist](#), FilePattern

[IfNotExist](#), FilePattern

[AttributeString](#)(See 16.10) := [FileExist](#)(See 10.)(FilePattern)

参数

FilePattern	要检查的路径、文件名或者文件类型。如果绝对路径没被指定, <i>FilePattern</i> 会被假设
-------------	--

	在 %A_WorkingDir%(See 9.) 。
--	--

相关命令

[FileExist\(\)](#)(See 10.), [Blocks](#)(See 17.2), [Else](#)(See 17.5), [File-loops](#)(See 17.12)

范例

```
IfExist, D:\  
    MsgBox, 驱动器存在。  
  
IfExist, D:\Docs\*.txt  
    MsgBox, 至少存在一个 .txt 文件。  
  
IfNotExist, C:\Temp\FlagFile.txt  
    MsgBox, 目标文件不存在。
```

翻译：天堂之门 menk33@163.com 2008 年 6 月 3 日

16.28 IniDelete

从标准格式的 .ini 文件中删除键值。

IniDelete, Filename, Section [, Key]

参数

Filename	.ini 文件名，如果不指定绝对路径，则假设它在 %A_WorkingDir%(See 9.) 目录下。
Section	.ini 文件中的段名，即包含在方括号里的标题词组(在这个参数里不需要写方括号)。
Key	.ini 文件中的键名。 如果省略，整个 Section 将被删除。

ErrorLevel

如果有问题，[ErrorLevel](#)(See 30.8) 被设为 1，否则为 0。然而，如果脚本是 .aut (AutoIt v2) 类型的，[ErrorLevel](#) 不会改变(兼容性的原因)。

注意

一个标准的 ini 文件看起来像这样：

```
[SectionName]  
Key=
```

相关命令

[IniRead\(See 16.29\)](#), [IniWrite\(See 16.30\)](#), [RegDelete\(See 25.2\)](#)

示例

```
IniDelete, C:\Temp\myfile.ini, section2, key
```

翻译：惊幻 修正：天堂之门 menk33@163.com 2008 年 8 月 24 日

16.29 IniRead

从标准格式的 .ini 文件读取一个键值。

`IniRead, OutputVar, Filename, Section, Key [, Default]`

参数

<code>OutputVar</code>	保存返回值的变量。如果得不到返回值，它被指定为 <i>Default</i> (如下所述)。
<code>Filename</code>	.ini 文件的名字。如果不指定绝对路径，则认为它在 %A_WorkingDir%(See 9.) 目录下。
<code>Section</code>	.ini 文件中的段名，是包含在方括号里的标题词组(这个参数不需要写方括号)。
<code>Key</code>	.ini 文件中的键值。
<code>Default</code>	<p>如果找不到指定键值，将保存到 <i>OutputVar</i> 中的内容。忽略此参数，它将被预设为 ERROR。指定 %A_Space%(See 9.) 来存放空值。</p> <p>对于 AutoIt (.aut) 脚本：由于兼容性问题，这个参数将被忽略。在读取 .ini 文件键值出错误时，<i>OutputVar</i> 中将总被指定为 ERROR。</p>

ErrorLevel

该命令不影响 [ErrorLevel\(See 30.8\)](#)。如上所述，出错时，*OutputVar* 将总被设定为参数 *Default* 的值。

注意

返回字符串中，开头和结尾的空格/制表符，将会被操作系统所忽略。

标准的 ini 文件具有如下形式：

```
[SectionName]
```

```
Key=
```

相关命令

[IniDelete\(See 16.28\)](#), [IniWrite\(See 16.30\)](#), [RegRead\(See 25.3\)](#), [file-reading loop\(See 16.32\)](#), [FileRead\(See 16.19\)](#)

示例

```
IniRead, OutputVar, C:\Temp\myfile.ini, section2, key
MsgBox, The value is %OutputVar%.
```

16.30 IniWrite

向标准格式的 .ini 文件中写入键值。

IniWrite, Value, Filename, Section, Key

参数

Value	写在键值等号(=)右侧的字符串或数字。如果文本过长，使用 continuation section (See 8.) 方法将它分成较短的几行，以增加可读性和可维护性。
Filename	.ini 文件的文件名，如果没有指定绝对路径，默认在 %A_WorkingDir% (See 9.) 目录下。
Section	.ini 文件中的段名，是包含在方括号里的标题词组(这个参数不需要写方括号)。
Key	.ini 文件中的键值。

ErrorLevel

出错的话，[ErrorLevel](#)(See 30.8) 被设为 1，否则为 0。但是，在 .aut (AutoIt v2) 脚本中，ErrorLevel 不发生变化(由于兼容性原因)。

注意

标准的 ini 文件具有如下形式：

```
[SectionName]
Key=
```

相关命令

[IniDelete](#)(See 16.28), [IniRead](#)(See 16.29), [RegWrite](#)(See 25.4)

示例

```
IniWrite, this is a new value, C:\Temp\myfile.ini, section2, key
```

16.31 Loop (files & folders)

获取指定的文件或文件夹，一次一个。

Loop, FilePattern [, IncludeFolders?, Recurse?]

参数

FilePattern	<p>单个文件或文件夹的名称，或一个通配符类型例如 C:\Temp*.tmp。如果未指定绝对路径将假设 <i>FilePattern</i> 在 %A_WorkingDir% (See 9.) 中。</p> <p>星号和问号标记都被支持作为通配符。当文件类型出现在文件的长/普通名称或它的 8.3 短名称 中时都将获得匹配。</p> <p>如果此参数是单个文件或文件夹（即没有通配符）并且 <i>Recurse</i> 设为 1，如果指定的文件名称出现在多个被搜索的文件夹中时，将找到多个匹配。</p>
IncludeFolders?	<p>使用下面的数字，或留空而使用默认：</p> <p>0 (默认) 不获取文件夹 (仅文件)。</p> <p>1 获取所有匹配通配符类型的文件和文件夹。</p> <p>2 只获取文件夹 (没有文件)。</p>
Recurse?	<p>使用下面的数字，或留空而使用默认：</p> <p>0 (默认) 子文件夹不被 <i>recessed into</i>(搜索)。</p> <p>1 搜索子文件夹以便获取包含在其中匹配 <i>FilePattern</i> 的文件和文件夹。所有的子文件夹将被搜索，不仅仅是那些名称匹配 <i>FilePattern</i> 的。</p>

可用在 文件-Loop 里的特有变量

下面的变量存在于任何 `文件-loop` 中。如果一个内部的 `文件-loop` 封在一个外部的 `文件-loop` 中，最里面 `loop` 的文件将享有优先权：

A_LoopFileName	当前取得的文件或文件夹名称 (不带路径)。
A_LoopFileExt	文件的扩展名 (例如 TXT, DOC, 或 EXE)。不包括点号 (.)。
A_LoopFileFullPath	当前取得的文件/文件夹的完全路径和名称。不过，如果 <i>FilePattern</i> 包含了一个相对路径而不是一个绝对路径，那么这里的路径也将是相对的。此外，任何在 <i>FilePattern</i> 里短 (8.3) 名称的文件夹也仍是短的 (请看下一行来获取长版名称)。
A_LoopFileLongPath	此变量在以下方面和 A_LoopFileFullPath 不同： 1) 它总是包含文件的绝对 / 完全路径，即使 <i>FilePattern</i> 包含的是一个相对路径； 2) 任何本身在 <i>FilePattern</i> 中是文件夹的短 (8.3) 名称的，将转换为它们的长名称； 3) 在 <i>FilePattern</i> 里的字符被转化为大写或小写，用来匹配存在文件系统里的大小写。将文件名称转换为它们被 Explorer(资源管理器) 显示的确切路径名称，这会很有用 -- 例如那些传递给一个脚本作为命令行的参数。
A_LoopFileShortPath	<p>当前取得的文件/文件夹的 8.3 短路径和名称。例如：</p> <p>C:\MYDOCU~1\ADDRES~1.txt。不过，如果 <i>FilePattern</i> 包含一个相对路径而不是一个绝对路径，那么这里的路径也将是相对的。</p> <p>要获取单个文件或文件夹完整的 8.3 路径和名称，像在这个例子中那样为</p>

	<p><i>FilePattern</i> 指定它的名称:</p> <pre>Loop, C:\My Documents\Address List.txt ShortPathName = %A_LoopFileShortPath%</pre> <p>注意: 如果文件没有短名称, 此变量将为空, 这会在那些注册表里设置了 NtfsDisable8dot3NameCreation(Ntfs 禁用 8 点 3 名称创建) 的操作系统上发生。如果 <i>FilePattern</i> 包含一个相对路径并且 <i>loop</i> 内部使用 SetWorkingDir(See 16.33) 从对 <i>loop</i> 自身起作用的工作目录切换出来, 它也将为空。</p>
A_LoopFileName	8.3 短名称, 或文件替换名称。如果文件没有此名称 (由于长名称比 8.3 格式更短或者大概因为短名称在一个 NTFS 文件系统上被禁止创建), A_LoopFileName 将代替它被获得。
A_LoopFileDir	A_LoopFileName 所属目录的完全路径。不过, 如果 <i>FilePattern</i> 包含一个相对路径而不是绝对路径, 那么这里的路径也将是相对的。根目录不包含结尾反斜线。例如: C:
A_LoopFileTimeModified	文件被最后修改的时间。格式 YYYYMMDDHH24MISS(See 16.26) 。
A_LoopFileTimeCreated	文件被创建的时间。格式 YYYYMMDDHH24MISS(See 16.26) 。
A_LoopFileTimeAccessed	文件被最后访问的时间。格式 YYYYMMDDHH24MISS(See 16.26) 。
A_LoopFileAttrib	当前获取的文件的 属性(See 16.10) 。
A_LoopFileSize	当前获取的文件的字节大小。同样支持大于 4 G 的文件。
A_LoopFileSizeKB	当前获取的文件的千字节大小, 四舍五入为整数。
A_LoopFileSizeMB	当前获取的文件的兆字节大小, 四舍五入为整数。

注意

当你想要对收集的文件和/或文件夹一次一个进行操作时, 文件-loop 是很有用的。

所有匹配的文件会被获取, 包括隐藏文件。相比之下, 操作系统的特性例如 DIR 命令默认省略隐藏文件。要避免处理隐藏、系统和/或只读文件, 在 loop 里使用一些像下面这样的命令:

```
if A_LoopFileAttrib contains H,R,S ; 跳过任何具有 H (隐藏), R (只读), 或 S (系统)的文件。
注意: 在 "H,R,S" 之间无空格。

continue ; 跳过此文件并继续下一个。
```

要在递归搜索期间获得文件的相对路径而不是绝对路径, 使用 [SetWorkingDir\(See 16.33\)](#) 来改变在 loop 之前所基于的文件夹, 并在之后的 Loop 里省略路径 (例如 Loop, *.*, 0, 1)。这将导致 A_LoopFilePath 包含的文件路径相对于它所基于文件夹。

如果一个文件-loop 在它自己的范围内创建或重命名文件或文件夹, 它会搅乱它自己。例如, 如果它通过 [FileMove\(See 16.16\)](#) 或其他方法重命名文件, 每个这样的文件可能被找到两次: 一次以它的旧名称, 再一次以它的新名称。要绕弯解决这个问题, 在创建一个文件的列表之后重命名它们。例如:

```
FileDialog =
Loop *.jpg
    FileList = %FileDialog%%A_LoopFileName%`n
Loop, parse, FileList, `n
    FileMove, %A_LoopField%, renamed_%A_LoopField%
```

在一个 NTFS 文件系统上，文件可能总会以字母顺序来获取。在其他文件系统里，文件没有以特别的顺序来获取。要确保一个特别的顺序，以 [Sort\(See 27.12\)](#) 命令示例部分下面显示的那样来使用。

文件和文件夹的完整路径名称长于 259 字符的部分将被略过，就像它们不存在一样。这样的文件相当罕见，因为通常操作系统不允许它们创建。

请看 [Loop\(See 17.12\)](#) 关于 [Blocks\(See 17.2\)](#), [Break\(See 17.3\)](#), [Continue\(See 17.4\)](#), 和 [A_Index](#) 变量（其存在于每种类型的 loop 中）的信息。

相关命令

[Loop\(See 17.12\)](#), [Break\(See 17.3\)](#), [Continue\(See 17.4\)](#), [Blocks\(See 17.2\)](#), [SplitPath\(See 16.34\)](#), [FileSetAttrib\(See 16.25\)](#), [FileSetTime\(See 16.26\)](#)

示例

```
; 例子 #1:
Loop, %A_ProgramFiles%\*.txt, , 1 ; 在子文件夹中搜索。
{
    MsgBox, 4, , 文件名称 = %A_LoopFilePath%`n`n继续?
    IfMsgBox, No
        break
}
```

```
; 例子 #2: 计算一个文件夹的大小，包括它所有子文件夹里的文件:
SetBatchLines, -1 ; 让操作以最快的速度运行。
FolderSizeKB = 0
FileSelectFolder, WhichFolder ; 告诉用户去挑选一个文件夹。
Loop, %WhichFolder%\*.*,, 1
    FolderSizeKB += %A_LoopFileSizeKB%
MsgBox %WhichFolder% 的大小是 %FolderSizeKB% KB 。
```

; 例子 #3: 获得以名称来分类的文件名 (请看下一个例子来以日期分类):

```
FileDialog = ; 初始化为空。

Loop, C:\*.*

    FileList = %FileDialog%%A_LoopFileName%`n

Sort, FileList, R ; R 选项以反方向分类。其他选项请看 Sort(See 27.12) 命令页面。

Loop, parse, FileList, `n

{

    if A_LoopField = ; 忽略在列表底部的空白。

        continue

    MsgBox, 4,, 文件编号 %A_Index% 是 %A_LoopField% 。继续?

    IfMsgBox, No

        break

}
```

; 例子 #4: 获得以修改日期分类的文件名称:

```
FileDialog =

Loop, %A_MyDocuments%\Photos\*.* , 1

    FileList = %FileDialog%%A_LoopFileTimeModified%`t%A_LoopFileName%`n

Sort, FileList ; 以日期分类。

Loop, parse, FileList, `n

{

    if A_LoopField = ; 省略在列表尾部的最后一个换行 (空项) 。

        continue

    StringSplit, FileItem, A_LoopField, %A_Tab% ; 以 tab 符号分为两部分。

    MsgBox, 4,, 下一个文件 (修改于 %FileItem1%) 是: `n%FileItem2%`n`n继续?

    IfMsgBox, No

        break

}
```

; 例子 #5: 仅复制比它们目标文件更新的源文件:

```
CopyIfNewer:
; 调用方已为我们设置了变量 CopySourcePattern 和 CopyDest 。
Loop, %CopySourcePattern%
{
    copy_it = n
    IfNotExist, %CopyDest%\%A_LoopFileName% ; 如果目标文件还未存在, 将总去复制。
    copy_it = y
    else
    {
        FileGetTime, time, %CopyDest%\%A_LoopFileName%
        EnvSub, time, %A_LoopFileTimeModified%, seconds ; 目标文件的时间减去源文件
        的时间。
        if time < 0 ; 源文件比目标文件更新。
        copy_it = y
    }
    if copy_it = y
    {
        FileCopy, %A_LoopFileFullPath%, %CopyDest%\%A_LoopFileName%, 1 ; 确定
        覆盖地复制
        if ErrorLevel
            MsgBox, 无法复制 "%A_LoopFileFullPath%" 到
            "%CopyDest%\%A_LoopFileName%" 。
    }
}
Return
```

; 例子 #6: 像存在文件系统中那样把通过命令行参数传递进来的文件名称转变为长文件名、完整路径以及正确的大写/小写字符。

```
Loop %0% ; 将每个文件拖到脚本 (或以一个参数传递)。
{
```

```

GivenPath := %A_Index% ; 获取下一命令行参数。
Loop %GivenPath%, 1
    LongPath = %A_LoopFileLongPath%
MsgBox 大小写正确的长路径文件名`n%GivenPath%`n 是: `n%LongPath%
}

```

翻译：天堂之门 menk33@163.com 2008 年 8 月 30 日

16.32 Loop (read file contents)

逐行读取一个文本文件的内容 (比 [FileReadLine\(See 16.18\)](#) 执行地更好)。

`Loop, Read, InputFile [, OutputFile]`

参数

Read	此参数必须是单词 READ。
InputFile	文本文件的名称，它的内容将被 loop 读取，如果未指定绝对路径，则假设位于 %A_WorkingDir%(See 9.) 。支持 Windows 和 Unix 格式的文件；就是说，文件行可以是回车加换行符 (`r`n) 结尾或者只用换行符 (`n) 结尾。
OutputFile	<p>(可选) 输出文件的名称，它在循环过程中会保持打开状态，如果未指定绝对路径，则假设位于 %A_WorkingDir%(See 9.)。</p> <p>在循环体内，请用只带一个参数(要写入的文本)的 FileAppend(See 16.4) 命令来将文本追加到此文件。以这种方式追加到一个文件比使用 2 个参数模式的 FileAppend(See 16.4) 执行地更好，因为每次操作不需要关闭文件再重新打开。如果需要，记得在写入内容后跟一个换行符 (`n)。</p> <p>如果没有写入操作，该文件不会被打开。例如循环次数是零，或者没有使用 FileAppend(See 16.4) 命令。</p> <p>二进制模式：要以二进制模式而不是文本模式追加的话，在文件名前加一个星号。这会使每个换行符 (`n) 作为一个单独的换行 (LF) 写入，而不是 Windows 标准的回车加换行 (CR+LF)。例如：*C:\My Unix File.txt。即使不用星号，如果循环第一次使用 FileAppend(See 16.4) 命令的时候写入了任何成对的回车和换行符 (`r`n)，二进制模式也会自动生效。</p> <p>标准输出 (stdout)：给 OutputFile 指定一个星号 (*) 可以将任何由 FileAppend(See 16.4) 写入的文本发送到标准输出 (stdout)。这样的文本可以重定向到一个文件、传送到另一个 EXE 程序或被 高级的文本编辑器(See 29.4) 获取。不过，以标准形式输出的文本并不会在执行它的命令提示符中出现。详见 FileAppend(See 16.4)。</p> <p>逗号转义：和其它大多数命令的最后一个参数不同，在 OutputFile 参数中，逗号必须进行转义 (`,)。</p>

注意

当你想对一个文本文件中的内容逐行进行操作的时候，文件读取循环十分有用。它比使用 [FileReadLine\(See 16.18\)](#) 执行地更好，因为：1) 文件在整个操作过程中可以保持打开状态；2) 不必每次都重新扫描文件来查找请求的行数。

内置变量 **A_LoopReadLine** 存在于任何文件读取循环中。它包含了当前行的内容，行尾的回车和换行符 (`r`n) 标记除外。如果一个内层文件读取循环被装在一个外层文件读取循环中，则最里层循环的文件行将享有优先。

每行最多可以读取 65,534 个字符。如果某行的长度有超出，那它剩余的字符将在下次循环中被读取。

在文件读取循环中，经常使用 [StringSplit\(See 27.21\)](#) 或者 [字符串解析循环\(See 17.14\)](#) 来解析从 *InputFile* 中读取的每一行内容。例如，如果 *InputFile* 中的每一行内容都是由 *tab* 分隔的一系列片段，那么这些片段可以用下面这段示例逐个读取：

```
Loop, read, C:\Database Export.txt
{
    Loop, parse, A_LoopReadLine, %A_Tab%
    {
        MsgBox, 第 %A_Index% 个片段是 %A_LoopField%.
    }
}
```

要将整个文件内容读取到一个变量中的话，请用 [FileRead\(See 16.19\)](#)，因为它比一个循环执行地好得多（特别是读取大文件的时候）。

要同时打开多个文件，请用 [DIICall\(\)](#)，参考[这个例子\(See 18.3\)](#)。

请看 [Loop\(See 17.12\)](#) 命令获取 [Blocks\(See 17.2\)](#), [Break\(See 17.3\)](#), [Continue\(See 17.4\)](#) 以及内置变量 *A_Index* (存在于任何类型的循环中) 的相关信息。

相关命令

[FileRead\(See 16.19\)](#), [FileReadLine\(See 16.18\)](#), [FileAppend\(See 16.4\)](#), [Sort\(See 27.12\)](#), [Loop\(See 17.12\)](#), [Break\(See 17.3\)](#), [Continue\(See 17.4\)](#), [Blocks, \(See 17.2\)](#)[FileSetAttrib\(See 16.25\)](#), [FileSetTime\(See 16.26\)](#)

示例

```
;示例 1: 第一个文件中, 只有含有单词 FAMILY 的行会被写入第二个文件。
;取消第一行的注释的话, 将会覆盖现有的文件, 而不是将内容追加到现有的文件中。
;FileDelete, C:\Docs\Family Addresses.txt
```

```
Loop, read, C:\Docs\Address List.txt, C:\Docs\Family Addresses.txt
{
    IfInString, A_LoopReadLine, family, FileAppend, %A_LoopReadLine%`n
}
```

;示例 2：读取一个文本文件的最后一行。

```
Loop, read, C:\Log File.txt
    last_line := A_LoopReadLine ;当循环结束时，这个变量将会保存最后一行的内容。
```

;示例 3：从一个文本文件或者 HTML 文件中提取所有的 FTP 和 HTTP URL。

```
FileSelectFile, SourceFile, 3,, 选择一个文本文件或者 HTML 文件来分析。
if SourceFile =
    return ;未选择文件则退出。

SplitPath, SourceFile,, SourceFilePath,, SourceFileNoExt
DestFile = %SourceFilePath%\%SourceFileNoExt% Extracted Links.txt

IfExist, %DestFile%
{
    MsgBox, 4,, 覆盖已存在的链接文件？按否则是追加。`n`n文件: %DestFile%
    IfMsgBox, Yes
        FileDelete, %DestFile%
}

LinkCount = 0
Loop, read, %SourceFile%, %DestFile%
{
    URLSearchString = %A_LoopReadLine%
    Gosub, URLSearch
```

```

}

MsgBox 查找到 %LinkCount% 个链接, 已写入 "%DestFile%"。

return


URLSearch:

;下面这么写是因为一些 URL 中嵌入了其它的 URL:

StringGetPos, URLStart1, URLSearchString, http://

StringGetPos, URLStart2, URLSearchString, ftp://

StringGetPos, URLStart3, URLSearchString, www.

;

;找到最左边的开始位置:

URLStart = %URLStart1% ;设置默认的开始位置。

Loop

{

;通过只解析 "URLStart%A_Index%" 一次来提高效率 (至少在一个含有许多变量的脚本中能提高):

ArrayElement := URLStart%A_Index%

if ArrayElement = -1 ;抵达数组的末尾。

break

if ArrayElement = -1 ;取消这个元素。

continue

if URLStart = -1

    URLStart = %ArrayElement%

else ; URLStart 中包含有效的起始位置, 所以和 ArrayElement 比较。

{

if ArrayElement <> -1

    if ArrayElement < %URLStart%


        URLStart = %ArrayElement%
}
}

```

```

}

}

if URLStart = -1 ; URLSearchString 中没有 URL。
    return

;否则，提取这个 URL:

StringTrimLeft, URL, URLSearchString, %URLStart% ;省略开始部分/不相关的内容。
Loop, parse, URL, %A_Tab%%A_Space%<> ;查找第一个空格, Tab, 或者尖括号 (如果有的话)。
{
    URL = %A_LoopField%
    break ;即只循环一次来获取首个“片段”。
}
;如果没有进入上面的循环，表示没找到结束符，将不处理 URL 变量的内容。

;如果 URL 以双引号结尾，移除它。目前使用了 StringReplace,
;但是要注意，由于双引号也可以出现在 URL 中，所以这个操作可能会破坏它们:
StringReplace, URLCleansed, URL, ",, All
FileAppend, %URLCleansed%`n
LinkCount += 1

;检查这一行中是否还有其它 URL:
StringLen, CharactersToOmit, URL
CharactersToOmit += %URLStart%
StringTrimLeft, URLSearchString, URLSearchString, %CharactersToOmit%
Gosub, URLSearch ;循环调用自身。
return

```

翻译: okey3m 修正: 天堂之门 menk33@163.com 2009 年 1 月 7 日

16.33 SetWorkingDir

更改脚本当前工作目录。

SetWorkingDir, DirName

参数

DirName	工作目录名。如果不使用绝对路径，则被认为是当前工作目录 %A_WorkingDir% (See 9.) 的子目录。
---------	---

ErrorLevel

如果遇到问题，[ErrorLevel/错误级别](#)(See 30.8) 被设置为 1，否则为 0。

注意

脚本的工作目录是脚本存取文件和目录时候使用的默认路径。如果存取文件或目录的时候未指定绝对路径，则使用默认路径。例如下面的例子中，文件 *My Filename.txt* 被存储在 [%A_WorkingDir%](#) 指定的目录中：[FileAppend](#)(See 16.4), A Line of Text, My Filename.txt

一个脚本的初始工作路径取决于它的启动方式。例如，通过“开始”菜单的快捷方式运行一个脚本的话，它的初始工作目录就是快捷方式属性中“起始位置”所显示的目录。

要让脚本无条件使用它所在的目录为工作目录的话，在脚本第一行写上：

SetWorkingDir %A_ScriptDir%

一旦更改了工作目录，它将立即生效并且影响所有的脚本。对所有的中断，[暂停](#)(See 17.18)，以及新运行的[线程](#)(See 30.19) 都有效，包括[定时器](#)(See 17.21)。

相关命令

[%A_WorkingDir%](#)(See 9.), [%A_ScriptDir%](#)(See 9.), [FileSelectFolder](#)(See 16.24)

示例

```
SetWorkingDir, D:\My Folder\Temp
```

16.34 SplitPath

将一个文件名或 URL (统一资源定位符)分解成它的名称、目录、扩展名和驱动器。

SplitPath, InputVar [, OutFileName, OutDir, OutExtension, OutNameNoExt, OutDrive]

参数

InputVar	包含用来被分析的文件名的变量名称。
----------	-------------------

OutFileName	储存不带路径的文件名的变量名称。文件的扩展名包括在内。
OutDir	储存文件的目录，包括驱动器字母或共享名称(如果有的话)的变量名称。最后的反斜线不被包括在内，即使文件定位在一个驱动器的根目录。
OutExtension	储存文件的扩展名(例如 TXT、DOC 或 EXE)的变量名称。圆点不被包括在内。
OutNameNoExt	储存不带路径、圆点和扩展名的文件名的变量名称。
OutDrive	储存文件的驱动器字母或服务器名称的变量名称。如果文件在一个本地或映射驱动器，变量将被设为驱动器字母跟一个冒号(无反斜线)。如果文件在一个网络路径(UNC 通用命名规则)，变量将被设为共享名，例如 \\Workstation01

说明

如果相应的信息不需要，任何一个输出变量可被省略。

如果 *InputVar* 包含一个缺少驱动器字母的文件名(即它不含路径或仅有一个相对路径)，*OutDrive* 将为空，但所有其他输出变量将设置正确。同样地，如果不存在路径，*OutDir* 将为空；如果有一个路径但不存在文件名，*OutFileName* 和 *OutNameNoExt* 将为空。

在文件系统里实际的文件和目录不通过此命令核查。它简单地分析在 *InputVar* 里给出的字串。

在文件名里不合法的通配符(*) 和 (?) 以及其他字符将与合法字符一样来对待，除了冒号、反斜线和句号(圆点)，是根据它们在划分驱动器字母、目录和文件扩展名的界限性质来处理的。

对 URLs 的支持：如果 *InputVar* 包含一个双斜线，例如 http://domain.com 或 ftp://domain.com，*OutDir* 将设为协议前缀 + 域名 + 目录 (例如 http://domain.com/images)，*OutDrive* 将设为协议前缀 + 域名 (例如 http://domain.com)。所有其他变量将根据它们上面的定义来设置。

相关命令

[A_LoopFileExt](#)(See 16.31), [StringSplit](#)(See 27.21), [StringGetPos](#)(See 27.14), [StringMid](#)(See 27.19), [StringTrimLeft](#)(See 27.22), [StringLeft](#)(See 27.15), [FileSelectFile](#)(See 16.23), [FileSelectFolder](#)(See 16.24)

范例

```
FullPathName = C:\My Documents\Address List.txt
```

;仅从上面取出单单文件名:

```
SplitPath, FullFileName, name
```

; 仅取出它的目录:

```
SplitPath, FullFileName,, dir
```

```
; 取出所有信息:
SplitPath, FullFileName, name, dir, ext, name_no_ext, drive
```

```
; 上面这行会将变量如下设置:
; name = Address List.txt
; dir = C:\My Documents
; ext = txt
; name_no_ext = Address List
; drive = C:
```

翻译：天堂之门 menk33@163.com 2008 年 7 月 23 日

17. 控制流

17.1 #Include/#IncludeAgain

使脚本表现得好像指定文件的内容就出现在这个位置。

```
#Include FileOrDirectoryName
#includeagain FileOrDirectoryName
```

参数

FileOrDirectoryName	<p>File: 要包含的文件名称，如果未指定绝对路径，它将假设在启动/工作目录(除了 ahk2exe(See 8.))，它会假设文件在脚本自己的目录)。文件名称不能包含变量引用(除了 %A_ScriptDir%(See 9.), %A_AppData%(See 9.) 和 %A_AppDataCommon%(See 9.))、双引号或者通配符。转义顺序(See 29.5)除了分号 (`;) 外都不能被使用，它们也不需要，因为字符比如百分号是原义对待的。注意：SetWorkingDir(See 16.33) 对 <code>#Include</code> 不起作用，因为 <code>#Include</code> 在脚本开始执行之前就已做处理。</p> <p>Directory: 指定一个目录而不是文件从而改变后面出现的所有 <code>#Include</code> 和 <code>FileInstall(See 16.15)</code> 使用的工作目录。目录名称不能包含变量，除了 %A_ScriptDir%(See 9.), %A_AppData%(See 9.) 和 %A_AppDataCommon%(See 9.)。注意：以这种方式改变工作目录不会影响脚本开始运行时的初始工作目录(A_WorkingDir(See 9.))。要改变那个初始工作目录，请在脚本的顶部使用 SetWorkingDir(See 16.33)。</p>
---------------------	---

注意

脚本表现得好像被包含的文件内容就在 `#Include` 指令的位置出现(就好像从包含的文件里复制粘贴过来一样)。因此，它通常不能将两个孤立的脚本合并成一个可运作的脚本(要实现的话，请看 www.autohotkey.com/forum/topic18545.html)。

`#Include` 会确保 *FileName* 仅被包含一次，即使它遇到多次重复的包含。相比之下，`#IncludeAgain` 则允许同个文件的多次包含，而在所有其他的方面和 `#Include` 一样。

可以在 *FileName* 参数前放置 `*i` 和单个空格，这样会使程序在加载被包含文件时忽略任何产生的故障。例如：`#Include *i SpecialOptions.ahk`。这个选项应该仅在被包含文件的内容对主脚本的操作是可有可无的情况下才去使用。

通过 [ListLines](#)(See 22.11) 或者菜单 View->Lines 显示在主窗口里的那些行总是按照它们在文件内部的物理顺序编号的。换句话说，包含一个新的文件将仅仅改变主脚本文件的一行编号，就是 `#Include` 这行(除了[已编译的脚本](#)(See 8.)，它会在编译时把它们包含的文件合并为一个大的脚本)。

`#Include` 常用来加载在一个外部文件里定义的[函数](#)(See 10.)。和子程序标签不同，[函数](#)(See 10.)可以被包含在脚本的最顶部而不影响[自动执行部分](#)(See 8.)。

和其他 `#` 开头的指令一样，`#Include` 不能被有条件地执行。换句话说，下例不起作用：

```
if x = 1

#Include SomeFile.ahk ;不管 x 的值是什么，这行都会生效。

y = 2 ;而此行是属于上面的 IF 语句的，因为 # 开头的指令不属于 IF 语句。
```

按函数名称在一个[函数库](#)(See 10.)里调用，可以自动地将文件包含进来(也就是不需要使用 `#Include`)。

相关命令

[Libraries of Functions](#)(See 10.), [Functions](#)(See 10.), [FileInstall](#)(See 16.15)

示例

```
#Include C:\My Documents\Scripts\Utility Subroutines.ahk

#Include %A_ScriptDir% ;为下面的 #Include 和 FileInstall 改变工作目录。

#Include C:\My Scripts ;同上，不过换成了一个明确命名的目录。
```

翻译：天堂之门 menk33@163.com 2008 年 11 月 6 日

17.2 Block

一对大括号表示一个区块。区块通常和[函数](#)(See 10.)、[Else](#)(See 17.5)、[Loop](#)(See 17.12)、[While-loop](#)(See 17.24) 和 IF 命令一起使用。

{

空行或多行命令

}

注意

区块被用来绑定两行或多行命令。它也能用来改变一个 [ELSE\(See 17.5\)](#) 从属于哪个 [IF](#) 语句，例如在此例中，区块迫使 [ELSE\(See 17.5\)](#) 从属于第一个 [IF](#) 语句而不是第二个：

```
if var1 = 1
{
    if var2 = abc
        sleep, 1
}
else
    return
```

虽然区块能被用在任何地方，不过目前它们只能和[函数\(See 10.\)](#)、[Else\(See 17.5\)](#)、[Loop\(See 17.12\)](#)或一个 [IF](#) 类型的命令例如 [IfEqual\(See 17.10\)](#) 或 [IfWinExist\(See 28.7\)](#) 一起使用时才有意义。

如果 [IF\(See 17.10\)](#)、[ELSE\(See 17.5\)](#)、[Loop\(See 17.12\)](#) 或 [While-loop\(See 17.24\)](#) 仅有一行命令，此命令不需要被装入区块。不过，这样做可以提高脚本的可读性或可维护性。

区块可以为空(不含任何命令)，当你想要注解掉区块的内容而不移除它本身时会变得很有用。

单个正确的大括号 (OTB, K&R 类型): OTB 类型可随意地使用在后面这些地方：[IF 语句的表达式\(See 17.11\)](#)、"[else\(See 17.5\)"](#) 关键字、[while-loops\(See 17.24\)](#)、[标准循环\(See 17.12\)](#)、[函数定义\(See 10.\)](#)。这种类型将区块的开始大括号放在区块控制语句的同一行而不是它的下一行上。例如：

```
if (x < y) {
    ...
} else {
    ...
}
While x < y {
    ...
}
Loop %RepeatCount% {
    ...
}
MyFunction(x, y) {
```

{

类似地，一个命令或其他动作可以存在于一个大括号的右边(除了单个正确的大括号类型的开始大括号外)。例如：

```
if x = 1
{ MsgBox 这行出现在一个开始大括号的右边。它在 IF 语句为真时执行。
    MsgBox 这是下一行。
} MsgBox 这行出现在一个结束大括号的右边。它无条件地执行。
```

相关命令

[Functions](#)(See 10.), [While-loop](#)(See 17.24), [Loop](#)(See 17.12), [Else](#)(See 17.5), [If](#)(See 17.10), [If\(Expression\)](#)(See 17.11)

示例

```
if x = 1
{
    MsgBox, 测试 1
    Sleep, 5
}
else
    MsgBox, 测试 2
```

翻译：天堂之门 menk33@163.com 2008 年 11 月 5 日

17.3 Break

退出（终止）某个 [loop](#)(See 17.12)。仅在 [loop](#)(See 17.12) 中有效。

Break

Break 语句用在封装了它的最内层循环。更鼓励使用 **Break** 和 [Continue](#)(See 17.4) 命令而不是 [goto](#)(See 17.9)，因为它们通常使脚本更具可读性和可维护性。

相关命令

[Continue](#)(See 17.4), [Loop](#)(See 17.12), [While-loop](#)(See 17.24), [Blocks](#)(See 17.2)

示例

```
loop
{
    ...
}
```

```

if var > 25
    break
...
if var <= 5
    continue
}

```

翻译: helfee 修正: 天堂之门 menk33@163.com 2008年11月6日

17.4 Continue

跳过当前 [Loop\(See 17.12\)](#)(循环) 重复的剩余部分并开始一个新的重复。在任何种类的 [Loop\(See 17.12\)](#) 内部都有效。

Continue

Continue 命令应用于它被装入的循环的最内层。它与到达循环的闭括号的效果是一样的:

1. [A_Index](#) 增加 1。
2. 它跳过了在它之后循环的其余的部分。
3. 循环的条件 (如果有的话) 会被判断是否满足。如果满足, 新的循环开始; 否则循环结束。

鼓励使用 [Break\(See 17.3\)](#) 和 [Continue](#) 命令胜于 [goto\(See 17.9\)](#) 命令, 由于它们通常使脚本更具可读性和可维护性。

相关命令

[Break\(See 17.3\)](#), [Loop\(See 17.12\)](#), [While-loop\(See 17.24\)](#), [Blocks\(See 17.2\)](#)

示例

```

; 此例显示了 5 个消息框, 6 到 10 的每个数字对应一个。
; 注意循环的前 5 次重复, "continue" 命令让循环在到达 MsgBox 这行前重新开始。
Loop, 10
{
if A_Index <= 5
    continue
    MsgBox %A_Index%
}

```

翻译: 天堂之门 menk33@163.com 2008年8月6日

17.5 Else

假如一个 **IF** 语句得出 **FALSE**, 则执行指定的命令。当出现多个命令时, 将它们括入一个**区块**(See 17.2)(大括号)。

Else

说明

每次使用 **ELSE** 都要附属于(相关联)它上面的一个 **IF** 语句。一个 **ELSE** 总是附属于上面那个距离它最近的未认领的 **IF** 语句, 除非使用了一个**区块**(See 17.2)来改变这种关联。

一个 **ELSE** 后面能立即被在同一行的其他单个命令跟着。其最常被"else if"梯形结构所使用(详见下面的范例)。

当一个 **IF**(See 17.10) 或一个 **ELSE** 命令拥有多行, 那些行必须被括入大括号。但是如果只有一行属于一个 **IF** 或一个 **ELSE** 命令, 那么大括号是可选的。例如:

```
if count > 0 ;不需要在下一行周围使用大括号, 因为它只有一行。
    MsgBox 按下确定开始此进程。
else ;一定要在下面这部分的周围使用大括号, 因为它由多行组成。
{
    WinClose 无标题 - 记事本
    MsgBox 当前没有内容。
}
```

单个正确的大括号(OTB)类型(See 17.2)可以选择性地被使用在一个"else"周围。例如:

```
if IsDone {
    ...
} else if (x < y) {
    ...
}
else {
    ...
}
```

相关命令

详见**区块**(See 17.2)。而且每个 **IF** 语句都能使用 **ELSE**, 包括 **IfWinActive**(See 28.6), **IfWinExist**(See 28.7), **IfMsgBox**(See 19.9), **IfInString**(See 27.3), **IfBetween**(See 21.7), **IfIn**(See 27.4), **IF**(See 17.10) 和 **IF (expression)**(See 17.11)。

范例

```
IfWinExist, 无标题 - 记事本
{
    WinActivate
    Send This is a test.{Enter}
}
else
{
    WinActivate, Some Other Window
    MouseClick, left, 100, 200
}

if x = 1
    Gosub, a1
else if x = 2 ; "else if" 类型
    Gosub, a2
else IfEqual, x, 3 ;交替的类型
{
    Gosub, a3
    Sleep, 1
}
else Gosub, a4 ;也就是说任何单个命令都能和一个 ELSE 命令在同一行。

;也可以这样:
IfEqual, y, 1, Gosub, b1
else {
    Sleep, 1
    Gosub, b2
}
```

翻译：天堂之门 menk33@163.com 2008 年 11 月 6 日

17.6 Exit

退出 [current thread](#)(See 30.19)(当前线程) 或(如果脚本非 [persistent](#)(See 29.21) 且未包含热键) 整个脚本。

Exit [, ExitCode]

参数

ExitCode	一个 integer (整数) (也就是负数、正数、零或者 expression (See 9.)(表达式)) 在脚本退出时被返回给它的调用者。此退出代码可被任何调用脚本的程序使用, 例如另一个脚本(通过 RunWait) 或一个 batch (.bat) file (批处理文件)。如果省略, ExitCode 默认为 0。0 通常被用来表示成功。注意: Windows 95 可能限制 ExitCode 的大小。
----------	---

注意

如果脚本没有热键, 也不是 [persistent](#)(See 29.21), 并且没有要求 Num/Scroll/CapsLock 键保持一直开启或一直关闭, 那么当遇到 Exit 时它会立即终止(除非它有一个 [OnExit](#)(See 17.17) 子程序)。

否则, Exit 命令只终止 [current thread](#)(See 30.19)(当前线程)。换句话说, 通过 [menu](#)(See 22.13)、[timer](#)(See 17.21) 或[热键](#)(See 4.)子程序来直接或间接地调用 [stack of subroutines](#) (子程序堆栈), 将好像各自直接地遇上一个 [Return](#)(See 17.19) 那样全部被返回。如果直接在这样的子程序内部使用 Exit -- 而不是在子程序内间接地被脚本调用 -- Exit 等同于 [Return](#)(See 17.19)。

使用 [ExitApp](#)(See 17.7) 来完全终止一个 [persistent](#)(See 29.21) 或包含热键的脚本。

相关命令

[ExitApp](#)(See 17.7), [OnExit](#)(See 17.17), [Functions](#)(See 10.), [Gosub](#)(See 17.8), [Return](#)(See 17.19), [Threads](#)(See 30.19), [#Persistent](#)(See 29.21)

示例

```
#z::  
Gosub, Sub2  
MsgBox, 由于 EXIT, 这个消息框将不会弹出。  
return  
  
Sub2:  
Exit ; 终止子程序, 也终止了来调用的热键子程序。
```

翻译: Xianggee 修正: 天堂之门 menk33@163.com 2008年8月8日

17.7 ExitApp

无条件地终止脚本。

ExitApp [, ExitCode]

参数

ExitCode	当脚本退出时返回一个整数(相当于负数、正数或者是零)给它的调用程序。此代码能被任何生成脚本的程序运用，例如另一个脚本(通过 RunWait)或者一个批处理(.bat)文件。如果省略， <code>ExitCode</code> 默认为零。零通常用来表示成功。注意：Windows 95 可能限制了 <code>ExitCode</code> 的大小。
----------	---

说明

脚本将立即被终止，除非它有一个 [OnExit\(See 17.17\)](#) 子程序。这相当于从脚本的托盘菜单或主菜单选择 "Exit"。

当脚本没有包含热键，不是[持续的\(See 29.21\)](#)，并且没有要求 Num/Scroll/Capslock 键保持一直打开或者一直关闭时，[Exit\(See 17.6\)](#) 和 [ExitApp](#) 作用相同。

如果脚本有一个 [OnExit\(See 17.17\)](#) 子程序，它将响应 [ExitApp](#) 而运行。

相关命令

[Exit\(See 17.6\)](#), [OnExit\(See 17.17\)](#), [#Persistent\(See 29.21\)](#)

范例

```
#x::ExitApp ;指定一个热键来终止此脚本。
```

翻译：天堂之门 menk33@163.com 2008 年 5 月 17 日

17.8 Gosub

跳到指定的标签并且继续执行，直到碰到 [Return\(See 17.19\)](#)。

`Gosub, Label`

参数

Label	跳转的标签名称， 热键标签(See 4.) 或者 热字串标签(See 5.) ，它执行 <code>Label</code> 下面的命令，直到碰到一个返回或退出。 "Return"(See 17.19) 让脚本跳回到 <code>Gosub</code> 下面的首个命令并且在那继续执行。 "Exit"(See 17.6) 终止 当前的线程(See 30.19) 。
-------	---

说明

和几乎所有其他命令的参数一样，`Label` 可以是一个[变量\(See 9.\)](#)引用，例如 `%MyLabel%`，这时储存在变量中的名称被用来作为目标。然而，性能有略微地损失，因为目标 `Label` 每次都要“查找”，而不是仅在脚本首次运行时查找一次。

当使用一个动态 `Label`，例如 `%MyLabel%`，如果 `Label` 不存在，将显示一个错误对话框。要避免这种情况，可提前调用 [IsLabel\(\)\(See 10.\)](#)。例如：

```
if IsLabel(VarContainingLabelName)
```

Gosub %VarContainingLabelName%

虽然 **Gosub** 可用于简单、多用途的子程序，但考虑为更复杂的用途而使用 [函数\(See 10.\)](#)。

相关命令

[Return\(See 17.19\)](#), [Functions\(See 10.\)](#), [IsLabel\(\)\(See 10.\)](#), [Blocks\(See 17.2\)](#), [Loop\(See 17.12\)](#), [Goto\(See 17.9\)](#)

范例

```
Gosub, Label1
 MsgBox, Label1 子程序已经返回（它已结束）。
return

Label1:
 MsgBox, Label1 子程序正在运行。
return
```

翻译：天堂之门 menk33@163.com 2008 年 11 月 8 日

17.9 Goto

跳转到指定的标签并继续执行。

Goto, Label

参数

Label	要跳转到的标签名称。
-------	------------

注意

当使用一个动态标签比如 `%MyLabel%` 时，如果标签不存在，将会显示一个错误对话框。要避免这种情况，可事先调用 [IsLabel\(\)\(See 10.\)](#)。例如：

```
if IsLabel(VarContainingLabelName)
  Goto %VarContainingLabelName%
```

不鼓励使用 **Goto**，因为它通常使脚本不太易读并且难以维护。可以考虑使用 [Else\(See 17.5\)](#), [Blocks\(See 17.2\)](#), [Break\(See 17.3\)](#) 和 [Continue\(See 17.4\)](#) 来代替 **Goto**。

相关命令

[Gosub](#)(See 17.8), [Return](#)(See 17.19), [IsLabel\(\)](#)(See 10.), [Else](#)(See 17.5), [Blocks](#)(See 17.2),
[Break](#)(See 17.3), [Continue](#)(See 17.4)

示例

```
Goto, MyLabel
...
MyLabel:
Sleep, 100
...
```

翻译: [Kookeee](#) 修正: 天堂之门 menk33@163.com 2008 年 9 月 24 日

17.10 If

指定一个[变量](#)(See 9.)和一个值比较得出 TRUE 时要执行的命令。当存在多个命令时，将它们括入一个[区块](#)(See 17.2)。

IfEqual, var, value (同: if var = value)
IfNotEqual, var, value (同: if var <> value) (!= 能用来替代 <>)
IfGreater, var, value (同: if var > value)
IfGreaterOrEqual, var, value (同: if var >= value)
IfLess, var, value (同: if var < value)
IfLessOrEqual, var, value (同: if var <= value)
If var ;如果变量的内容为空或为 0，那么它被视为 false。否则，它为 true。

另外可见: [IfInString](#)(See 27.3)

参数

var	变量(See 9.)名称。
value	原义的字串、数字或者变量引用(例如 %var2%)。Value 可以被省略，如果你想要将 var 与一个空字符串(空白的)相比较。

说明

如果 *var* 和 *value* 都是纯粹的数值型，它们将被作为数字比较，而不是作为字符串。否则，它们将作为字符串按字母顺序来比较（就是说，字母的次序将决定 *var* 是否小于、等于或者大于 *value*）。**译注: A<B**

当一个 IF 或者 ELSE(See 17.5) 拥有多行时，那些行必须被括入大括号。例如:

```
if count <= 0
{
    WinClose Untitled - Notepad
    MsgBox There are no items present.
```

```
}
```

然而，如果仅有一行属于 **IF** 或者 **ELSE**，大括号就是可选的。

如果你使用命令名称类型，那么另一个命令就只能出现在 **IF** 语句的同一行。换句话说，这些是有效的：

```
IfEqual, x, 1, Sleep, 1
IfGreater, x, 1, EnvAdd, x, 2
```

但是这些是无效的：

```
if x = 1 Sleep 1
IfGreater, x, 1, x += 2
```

单个正确的大括号(OTB)类型(See 17.11)不能和这些类型的 **IF** 语句一起使用。它只能和表达式类型的 **IF** 语句一起使用。

做一个相关的提示，“**if var [not] between LowerBound and UpperBound**(See 21.7)” 命令检查一个变量是否在两个 **values** 之间，而 “**if var [not] in value1,value2**(See 27.4)” 能用来检查一个变量的内容是否在 **values** 列表里存在。

相关命令

[IF \(expression\)](#)(See 17.11), [StringCaseSense](#)(See 27.13), [Assign expression \(:=\)](#)(See 21.12), [if var in/contains MatchList](#)(See 27.4), [if var between](#)(See 21.7), [IfInString](#)(See 27.3), [Blocks](#)(See 17.2), [Else](#)(See 17.5)

范例

```
if counter >= 1
    Sleep, 10

if counter >=1      ;如果一个 IF 有多行，将那些行插入大括号：
{
    WinClose, Untitled - Notepad
    Sleep 10
}

if MyVar = %MyVar2%
    MsgBox MyVar 和 MyVar2 的内容是相同的。
else if MyVar =
```

```
{
    MsgBox, 4,, MyVar 是空的/空白的。继续?

    IfMsgBox, No

        Return

}

else if MyVar <> ,

    MsgBox value 在 MyVar 里不是一个逗号。

else

    MsgBox value 在 MyVar 里是一个逗号。

if Done

    MsgBox 变量 Done 即不为空也不为零。
```

翻译: 天堂之门 menk33@163.com 2008 年 11 月 14 日

17.11 If (expression)

注意

包含表达式的 `if` 语句不同于[传统的 if\(See 17.10\)](#) 例如 `If FoundColor <> Blue`, 其在单词 "if" 后面的字符是左括号。尽管通常情况下会将整个表达式围在圆括号内, 不过也可以这样 `if(x > 0) and (y > 0)`。此外, 如果紧跟 "if" 后的是一个[函数调用\(See 10.\)](#)或者是一个运算符例如 "not" 或 "!" , 那么左括号将被整个省略掉。

如果 `if` 的条件语句为真(除过空字符串或数字 0 之外的其他结果), 下面的一行或者 [block\(See 17.2\)](#) (语句块) 将会被执行。否则如果有相应的 `ELSE`, 那么将执行 `ELSE` 下面的一行或者语句块。

当一个 `IF` 或 `ELSE(See 17.5)` 语句有许多行时, 那些行要用 [braces\(See 17.2\)](#) (大括号) 围起来。然而, 如果 `IF` 或 `ELSE` 语句只有一行, 那么大括号就可以不用。请看本页末尾的例子。

单个正确的大括号(OTB)类型可以和表达式 `if` 语句一起用(但不能用在[传统的 if 语句\(See 17.10\)](#))。例如:

```
if (x < y) {

    ...

}

if WinExist("无标题 - 记事本") {

    WinActivate
```

```

}

if IsDone {

    ...

}

else {

    ...

}

```

不像 "else" 语句，它的右边支持紧跟任意类型的语句。if 语句的右边仅支持一个 "}"。

做个相关的提示，"if var [not] between LowerBound and UpperBound(See 21.7)" 命令检查变量的值是否处在两个值之间，而 "if var [not] in value1,value2(See 27.4)" 命令能检查变量的内容是否在一个值的列表中存在。

相关命令

[Expressions](#)(See 9.), [Assign expression \(:=\)](#)(See 21.12), [if var in/contains MatchList](#)(See 27.4),
[if var between](#)(See 21.7), [IfInString](#)(See 27.3), [Blocks](#)(See 17.2), [Else](#)(See 17.5),
[While-loop](#)(See 17.24)

示例

```

if (A_Index > 100 or Done)
    return

if (A_TickCount - StartTime > 2*MaxTime + 100)
{
    MsgBox 经过太长时间了。
    ExitApp
}

if (Color = "Blue" or Color = "White")
{
    MsgBox 颜色是允许值之一。
    ExitApp
}
else if (Color = "Silver")
{
    MsgBox 银色不是允许的颜色。
    return
}
else
{
    MsgBox 该颜色不可识别。
}

```

```
ExitApp
}
```

翻译: yugi 修正: 天堂之门 menk33@163.com 2008 年 11 月 14 日

17.12 Loop

重复地执行一系列命令：可以是指定了数字的重复次数或直到遇上了 [break\(See 17.3\)](#)。

`Loop [, Count]`

参数

Count	<p>执行 <code>loop</code>(循环)多少次(反复)。如果省略，循环无限期地继续下去，直到遇上一个 break(See 17.3) 或 return(See 17.19)。</p> <p>如果 <code>Count</code> 是一个引用变量例如 <code>%ItemCount%</code>，每当变量为空或包含的数字小于 1，循环将被整个地跳过。</p> <p>由于必须支持 file-pattern loops(See 16.31)(文件类型循环)，<code>Count</code> 不可以是一个表达式。不过，和所有非表达式参数一样，一个表达式可以通过在其前面加上一个 % 和一个空格来强制地被使用。例如：<code>Loop % Count + 1</code>。在此例中，表达式仅被计值一次，就在循环开始前。</p>
-------	--

注意

循环命令通常后跟一个 [block\(See 17.2\)](#) (区块)，它是一些语句的集合，组成循环部分。不过，循环仅有 一条语句时，不需要一个区块(一个 "if" 以及它的 "else" 复合语句在这种用途时算作单独一个命令)。

此命令通常的用法是一个无限循环，其在循环部分的某处使用 [break\(See 17.3\)](#) 命令来决定何时停止循环。

在一个循环里鼓励使用 [break\(See 17.3\)](#) 和 [continue\(See 17.4\)](#) 来代替 [goto\(See 17.9\)](#)，由于它们通常使一个脚本更加易懂和更具可维护性。要创建一个" While" loop，让一个 IF 语句作为循环部分的首行语句从而有条件地分配 [break\(See 17.3\)](#) 命令。要创建一个" Do...While" loop，除了要把 IF 语句放在循环部分语句的末尾外，使用的是相同的手法。

内置变量 **A_Index** 包含当前循环重复的次数。循环的 `body`(循环体) 第一次执行时，它包含 1。第二次时，它包含 2；以此类推。如果一个内部循环装在一个外部循环中，内部循环拥有优先权使用 **A_Index**。
A_Index 能用在所有类型的循环内部，包括 [file-loops\(See 16.31\)](#) 和 [registry-loops\(See 17.16\)](#)；但是 **A_Index** 在循环外部包含 0。

[One True Brace \(OTB\) style\(See 17.2\)](#)(单个正确的大括号类型)可以随意地和标准循环一起使用(但不可以是例如 [file-pattern\(See 16.31\)](#)(文件类型) 和 [parsing\(See 17.14\)](#)(分解) 这种特化的循环)。例如：

```
Loop {
...
}
```

```
Loop %RepeatCount% {
    ...
}
```

特化的循环：循环能被用来自动地获取文件、文件夹或注册表项(一次一个)。详见 [file-loop\(See 16.31\)](#) 和 [registry-loop\(See 17.16\)](#)。此外，[file-reading loops\(See 16.32\)](#) 能对一整个文件的内容进行操作，一次一行。最后，[parsing loops\(See 17.14\)](#) 能对一个限定的字串内部的独立部分进行操作。

相关命令

[While-loop\(See 17.24\)](#), [Files-and-folders loop\(See 16.31\)](#), [Registry loop\(See 17.16\)](#),
[File-reading loop\(See 16.32\)](#), [Parsing loop\(See 17.14\)](#), [Break\(See 17.3\)](#), [Continue\(See 17.4\)](#),
[Blocks\(See 17.2\)](#)

示例

```
Loop, 3
{
    MsgBox, 重复的次数是 %A_Index%. ; A_Index 将是 1 、 2 ， 然后是 3
    Sleep, 100
}
Loop
{
    if a_index > 25
        break ; 终止循环
    if a_index < 20
        continue ; 跳过以下部分并开始一次新的循环
    MsgBox, a_index = %a_index% ; 此对话框将仅在 A_Index 是数字 20 到 25 时显示
}
```

翻译：天堂之门 menk33@163.com 2008 年 8 月 5 日

17.13 Loop (files & folders)

获取指定的文件或文件夹，一次一个。

Loop, FilePattern [, IncludeFolders?, Recurse?]

参数

FilePattern	<p>单个文件或文件夹的名称，或一个通配符类型例如 C:\Temp*.tmp 。如果未指定绝对路径将假设 <i>FilePattern</i> 在 %A_WorkingDir%(See 9.) 中。</p> <p>星号和问号标记都被支持作为通配符。当文件类型出现在文件的长/普通名称或它的 8.3 短名称 中时都将获得匹配。</p>
-------------	---

	如果此参数是单个文件或文件夹（即没有通配符）并且 <i>Recurse</i> 设为 1，如果指定的文件名称出现在多个被搜索的文件夹中时，将找到多个匹配。
IncludeFolders?	使用下面的数字，或留空而使用默认： 0 (默认) 不获取文件夹 (仅文件)。 1 获取所有匹配通配符类型的文件和文件夹。 2 只获取文件夹 (没有文件)。
Recurse?	使用下面的数字，或留空而使用默认： 0 (默认) 子文件夹不被 <i>recessed into</i> (搜索)。 1 搜索子文件夹以便获取包含在其中匹配 <i>FilePattern</i> 的文件和文件夹。所有的子文件夹将被搜索，不仅仅是那些名称匹配 <i>FilePattern</i> 的。

可用在 文件-Loop 里的特有变量

下面的变量存在于任何 `文件-loop` 中。如果一个内部的 `文件-loop` 封在一个外部的 `文件-loop` 中，最里面 `loop` 的文件将享有优先权：

A_LoopFileName	当前取得的文件或文件夹名称（不带路径）。
A_LoopFileExt	文件的扩展名（例如 <code>TXT</code> , <code>DOC</code> , 或 <code>EXE</code> ）。不包括点号（ <code>.</code> ）。
A_LoopFileFullPath	当前取得的文件/文件夹的完全路径和名称。不过，如果 <i>FilePattern</i> 包含了一个相对路径而不是一个绝对路径，那么这里的路径也将是相对的。此外，任何在 <i>FilePattern</i> 里短（8.3）名称的文件夹也仍是短的（请看下一行来获取长版名称）。
A_LoopFileLongPath	此变量在以下方面和 <code>A_LoopFileFullPath</code> 不同：1) 它总是包含文件的绝对 / 完全路径，即使 <i>FilePattern</i> 包含的是一个相对路径；2) 任何本身在 <i>FilePattern</i> 中是文件夹的短（8.3）名称的，将转换为它们的长名称；3) 在 <i>FilePattern</i> 里的字符被转化为大写或小写，用来匹配存在文件系统里的大小写。将文件名称转换为它们被 <i>Explorer</i> （资源管理器）显示的确切路径名称，这会很有用 -- 例如那些传递给一个脚本作为命令行的参数。
A_LoopFileShortPath	<p>当前取得的文件/文件夹的 8.3 短路径和名称。例如： <code>C:\MYDOCU~1\ADDRES~1.txt</code>。不过，如果 <i>FilePattern</i> 包含一个相对路径而不是一个绝对路径，那么这里的路径也将是相对的。</p> <p>要获取单个文件或文件夹完整的 8.3 路径和名称，像在这个例子中那样为 <i>FilePattern</i> 指定它的名称：</p> <pre>Loop, C:\My Documents\Address List.txt ShortPathName = %A_LoopFileShortPath%</pre> <p>注意：如果文件没有短名称，此变量将为空，这会在那些注册表里设置了 <code>NtfsDisable8dot3NameCreation</code> (<code>Ntfs</code> 禁用 8 点 3 名称创建) 的操作系统上发生。如果 <i>FilePattern</i> 包含一个相对路径并且 <code>loop</code> 内部使用 SetWorkingDir (See 16.33) 从对 <code>loop</code> 自身起作用的工作目录切换出来，</p>

	它也将为空。
A_LoopFileName	8.3 短名称，或文件替换名称。如果文件没有此名称（由于长名称比 8.3 格式更短或者大概因为短名称在一个 NTFS 文件系统上被禁止创建），A_LoopFileName 将代替它被获得。
A_LoopFileDir	A_LoopFileName 所属目录的完全路径。不过，如果 FilePattern 包含一个相对路径而不是绝对路径，那么这里的路径也将是相对的。根目录不包含结尾反斜线。例如： C:
A_LoopFileTimeModified	文件被最后修改的时间。格式 YYYYMMDDHH24MISS(See 16.26)。
A_LoopFileTimeCreated	文件被创建的时间。格式 YYYYMMDDHH24MISS(See 16.26)。
A_LoopFileTimeAccessed	文件被最后访问的时间。格式 YYYYMMDDHH24MISS(See 16.26)。
A_LoopFileAttrib	当前获取的文件的 属性(See 16.10)。
A_LoopFileSize	当前获取的文件的字节大小。同样支持大于 4 G 的文件。
A_LoopFileSizeKB	当前获取的文件的千字节大小，四舍五入为整数。
A_LoopFileSizeMB	当前获取的文件的兆字节大小，四舍五入为整数。

注意

当你想要对收集的文件和/或文件夹一次一个进行操作时，文件-loop 是很有用的。

所有匹配的文件会被获取，包括隐藏文件。相比之下，操作系统的特性例如 DIR 命令默认省略隐藏文件。要避免处理隐藏、系统和/或只读文件，在 loop 里使用一些像下面这样的命令：

if A_LoopFileAttrib contains H,R,S ; 跳过任何具有 H (隐藏), R (只读), 或 S (系统)的文件。

注意：在 "H,R,S" 之间无空格。

continue ; 跳过此文件并继续下一个。

要在一个递归搜索期间获得文件的相对路径而不是绝对路径，使用 SetWorkingDir(See 16.33) 来改变在 loop 之前所基于的文件夹，并在之后的 Loop 里省略路径（例如 Loop, *.*, 0, 1）。这将导致 A_LoopFullPath 包含的文件路径相对于它所基于文件夹。

如果一个文件-loop 在它自己的范围内创建或重命名文件或文件夹，它会搅乱它自己。例如，如果它通过 FileMove(See 16.16) 或其他方法重命名文件，每个这样的文件可能被找到两次：一次以它的旧名称，再一次以它的新名称。要绕弯解决这个问题，在创建一个文件的列表之后重命名它们。例如：

```
FileDialog =
Loop *.jpg
FileDialog = %FileDialog%%A_LoopFileName%`n
Loop, parse, FileDialog, `n
FileMove, %A_LoopField%, renamed_%A_LoopField%
```

在一个 NTFS 文件系统上，文件可能总会以字母顺序来获取。在其他文件系统里，文件没有以特别的顺序来获取。要确保一个特别的顺序，以 Sort(See 27.12) 命令示例部分下面显示的那样来使用。

文件和文件夹的完整路径名称长于 259 字符的部分将被略过, 就像它们不存在一样。这样的文件相当罕见, 因为通常操作系统不允许它们创建。

请看 [Loop\(See 17.12\)](#) 关于 [Blocks\(See 17.2\)](#), [Break\(See 17.3\)](#), [Continue\(See 17.4\)](#), 和 [A_Index](#) 变量 (其存在于每种类型的 loop 中) 的信息。

相关命令

[Loop\(See 17.12\)](#), [Break\(See 17.3\)](#), [Continue\(See 17.4\)](#), [Blocks\(See 17.2\)](#), [SplitPath\(See 16.34\)](#), [FileSetAttrib\(See 16.25\)](#), [FileSetTime\(See 16.26\)](#)

示例

```
; 例子 #1:

Loop, %A_ProgramFiles%\*.txt, , 1 ; 在子文件夹中搜索。

{
    MsgBox, 4, , 文件名称 = %A_LoopFilePath%`n`n继续?

    IfMsgBox, No
        break
}
```

```
; 例子 #2: 计算一个文件夹的大小, 包括它所有子文件夹里的文件:

SetBatchLines, -1 ; 让操作以最快的速度运行。

FolderSizeKB = 0

FileSelectFolder, WhichFolder ; 告诉用户去挑选一个文件夹。

Loop, %WhichFolder%\*.*,, 1
    FolderSizeKB += %A_LoopFileSizeKB%

MsgBox %WhichFolder% 的大小是 %FolderSizeKB% KB 。
```

```
; 例子 #3: 获得以名称来分类的文件名 (请看下一个例子来以日期分类):

FileList = ; 初始化为空。

Loop, C:\*.*

    FileList = %FileList%%A_LoopFileName%`n

Sort, FileList, R ; R 选项以反方向分类。其他选项请看 Sort\(See 27.12\) 命令页面。
```

```

Loop, parse, fileList, `n
{
    if A_LoopField = ; 忽略在列表底部的空白。
        continue

    MsgBox, 4,, 文件编号 %A_Index% 是 %A_LoopField% 。继续?
    IfMsgBox, No
        break
}

```

; 例子 #4: 获得以修改日期分类的文件名称:

```

fileList =
Loop, %A_MyDocuments%\Photos\*.* , 1
    fileList = %fileList%%A_LoopFileTimeModified%`t%A_LoopFileName%`n
Sort, fileList ; 以日期分类。

Loop, parse, fileList, `n
{
    if A_LoopField = ; 省略在列表尾部的最后一个换行 (空项)。
        continue

    StringSplit, fileItem, A_LoopField, %A_Tab% ; 以 tab 符号分为两部分。
    MsgBox, 4,, 下一个文件 (修改于 %fileItem1%) 是: `n%fileItem2%`n`n继续?
    IfMsgBox, No
        break
}

```

; 例子 #5: 仅复制比它们目标文件更新的源文件:

```

CopyIfNewer:
; 调用方已为我们设置了变量 CopySourcePattern 和 CopyDest 。
Loop, %CopySourcePattern%
{

```

```

copy_it = n

IfNotExist, %CopyDest%\%A_LoopFileName% ; 如果目标文件还未存在，将总去复制。

copy_it = y

else

{

    FileGetTime, time, %CopyDest%\%A_LoopFileName%

    EnvSub, time, %A_LoopFileTimeModified%, seconds ; 目标文件的时间减去源文件
    的时间。

    if time < 0 ; 源文件比目标文件更新。

        copy_it = y

}

if copy_it = y

{

    FileCopy, %A_LoopFileFullPath%, %CopyDest%\%A_LoopFileName%, 1 ; 确定
    覆盖地复制

    if ErrorLevel

        MsgBox, 无法复制 "%A_LoopFileFullPath%" 到
        "%CopyDest%\%A_LoopFileName%" 。

}

}

Return

```

; 例子 #6: 像存在文件系统中那样把通过命令行参数传递进来的文件名称转变为长文件名、完整路径
以及正确的大写/小写字符。

```

Loop %0% ; 将每个文件拖到脚本 (或以一个参数传递)。

{

    GivenPath := %A_Index% ; 获取下一命令行参数。
    Loop %GivenPath%, 1
        LongPath = %A_LoopFileLongPath%
    MsgBox 大小写正确的长路径文件名`n%GivenPath%`n 是: `n%LongPath%
}

```

翻译：天堂之门 menk33@163.com 2008 年 8 月 30 日

17.14 Loop (parse a string)

根据分隔符，从一个字符串中循环获取子字符串(片段)，一次获取一段。

`Loop, Parse, InputVar [, Delimiters, OmitChars]`

参数

Parse	这个参数必须使用单词 <code>PARSE</code> ，而且和其它的循环命令不同，这里不能引用变量的值。
InputVar	需要被解析的变量。不要用百分号将变量名包起来，除非该变量的 内容 就是需要被解析的变量名。
Delimiters	<p>分隔符。如果这个参数为空或省略，循环将依次获取 <code>InputVar</code> 中的每个字符。</p> <p>如果这个参数使用 <code>CSV</code>，<code>InputVar</code> 将按照标准的逗号分隔格式进行解析。这里有一个 MS Excel 生成的 CSV 文件的示例: "first field",SecondField,"the word ""special"" is quoted literally","","last field, has literal comma"</p> <p>其它情况下，<code>Delimiters</code> 中可以包含一个或多个字符(区分大小写)，每一个字符都会作为分隔符，用来将 <code>InputVar</code> 中的字符串分隔为相应的子字符串。</p> <p>分隔符将不会出现在被分隔出来子字符串中。另外，如果在 <code>InputVar</code> 中两个分隔符之间没有任何内容，则相应解析出来的子字符串为空。</p> <p>例如：`，(经过转义的逗号) 会按照字符串中逗号的位置将字符串分隔为多个子字符串。相似的，%A_Tab%%A_Space% 则会按照字符串中空格和制表符的位置将字符串分隔。</p> <p>如果需要使用一个字符串作为分隔符，可以先使用 StringReplace(See 27.20) 命令将 <code>InputVar</code> 中相应的字符串替换为一个在 <code>InputVar</code> 中从未使用的字符，例如这些特殊字符: ¢¤¥¦§©¤®µ¶，然后再使用这些特殊字符作为分隔符进行解析。参考下面这个例子，使用字符串
 作为分隔符：</p> <pre> StringReplace, NewHTML, HTMLString,
, ¢, All Loop, parse, NewHTML, ¢ ; 使用 ¢ 解析字符串。 { ... } </pre>
OmitChars	<p>忽略字符。这个可选参数中可以包含一个或多个字符(区分大小写)，这些字符会从解析出来的子字符串的开头和结尾部分移除。例如，<code>OmitChars</code> 参数使用 %A_Space%%A_Tab%，则如果解析出来的子字符串的开头或结尾部分有空格或制表符的话，这些空格和制表符会被删除(在子字符串中间的空格和制表符不会被删除)。</p> <p>如果 <code>Delimiters</code> 参数留空，<code>OmitChars</code> 参数指定的字符将不会出现在循环中。</p>

和其它大多数命令最后一个参数不同，在 *OmitChars* 参数中，逗号必须进行转义(`,)。

注意

当需要对一个字符串的各个片段进行逐个操作的时候，经常使用字符串解析循环。字符串解析循环也比 [StringSplit\(See 27.21\)](#) 命令更省内存(因为 `StringSplit` 命令创建了一个持久的数组)，而且在大多数情况下更加易用。

内置变量 **A_LoopField** 存在于任何一种解析循环中，它包含了从 *InputVar* 解析出来的当前子字符串(或片段)的内容。如果一个内部解析循环被装在一个外部解析循环中，则最里层循环中解析出来的片段将享有优先。

InputVar 或它的片段的大小没有限制。另外，即使在循环过程中 *InputVar* 的内容改变了，`Loop` 也会“无视”这些改变，因为它操作的是原始内容的临时副本。

要在解析之前将字符串中的片段进行排序，请用 [Sort\(See 27.12\)](#) 命令。

请看 [Loop\(See 17.12\)](#) 命令获取 [Blocks\(See 17.2\)](#), [Break\(See 17.3\)](#), [Continue\(See 17.4\)](#) 以及内置变量 **A_Index** (存在于任何类型的循环中) 的相关信息。

相关命令

[StringSplit\(See 27.21\)](#), [file-reading loop\(See 16.32\)](#), [Loop\(See 17.12\)](#), [Break\(See 17.3\)](#), [Continue\(See 17.4\)](#), [Blocks\(See 17.2\)](#), [Sort\(See 27.12\)](#), [FileSetAttrib\(See 16.25\)](#), [FileSetTime\(See 16.26\)](#)

示例

```
;示例 1:

Colors = red,green,blue

Loop, parse, Colors, `,
{
    MsgBox, 第 %A_Index% 种颜色是 %A_LoopField%。
}
```

;示例 2：逐行读取一个变量中的内容(类似文件读取循环([See 16.32](#))。

;使用 [FileRead\(See 16.19\)](#) 命令可以将一个文件读取到一个变量中:

Loop, parse, FileContents, `n, `r ;将 `n 写在 `r 前面以保证 Windows 和 Unix 这两者的文件都能被正常解析。

{

```

    MsgBox, 4, , 第 %A_Index% 行是 %A_LoopField%。`n`n继续?
    IfMsgBox, No, break
}

```

;示例 3: 这个示例和上面的一样, 只不过解析对象是剪贴板。

;当剪贴板中包含文件时很有用, 例如从一个打开的资源管理器窗口中复制的文件(程序会自动将这些文件转换为它们完整的路径):

```

Loop, parse, clipboard, `n, `r
{
    MsgBox, 4, , 第 %A_Index% 个文件是 %A_LoopField%。`n`n继续?
    IfMsgBox, No, break
}

```

;示例 4: 解析一个逗号分隔(CSV)格式的文件:

```

Loop, read, C:\Database Export.csv
{
    LineNumber = %A_Index%
    Loop, parse, A_LoopReadLine, CSV
    {
        MsgBox, 4, , 片段 %LineNumber%-%A_Index% 是: `n%A_LoopField%`n`n继续?
        IfMsgBox, No
            return
    }
}

```

翻译: okey3m 修正: 天堂之门 menk33@163.com 2009 年 1 月 7 日

17.15 Loop (read file contents)

逐行读取一个文本文件的内容 (比 [FileReadLine](#)(See 16.18) 执行地更好)。

Loop, Read, InputFile [, OutputFile]

参数

Read	此参数必须是单词 READ。
InputFile	文本文件的名称，它的内容将被 loop 读取，如果未指定绝对路径，则假设位于 %A_WorkingDir%(See 9.) 。支持 Windows 和 Unix 格式的文件；就是说，文件行可以是回车加换行符 (`r`n) 结尾或者只用换行符 (`n) 结尾。
OutputFile	<p>(可选) 输出文件的名称，它在循环过程中会保持打开状态，如果未指定绝对路径，则假设位于 %A_WorkingDir%(See 9.)。</p> <p>在循环体内，请用只带一个参数(要写入的文本)的 FileAppend(See 16.4) 命令来将文本追加到此文件。以这种方式追加到一个文件比使用 2 个参数模式的 FileAppend(See 16.4) 执行地更好，因为每次操作不需要关闭文件再重新打开。如果需要，记得在写入内容后跟一个换行符 (`n)。</p> <p>如果没有写入操作，该文件不会被打开。例如循环次数是零，或者没有使用 FileAppend(See 16.4) 命令。</p> <p>二进制模式：要以二进制模式而不是文本模式追加的话，在文件名前加一个星号。这会使每个换行符 (`n) 作为一个单独的换行 (LF) 写入，而不是 Windows 标准的回车加换行 (CR+LF)。例如：*C:\My Unix File.txt。即使不用星号，如果循环第一次使用 FileAppend(See 16.4) 命令的时候写入了任何成对的回车和换行符 (`r`n)，二进制模式也会自动生效。</p> <p>标准输出 (stdout)：给 OutputFile 指定一个星号 (*) 可以将任何由 FileAppend(See 16.4) 写入的文本发送到标准输出 (stdout)。这样的文本可以重定向到一个文件、传送到另一个 EXE 程序或被 高级的文本编辑器(See 29.4) 获取。不过，以标准形式输出的文本并不会在执行它的命令提示符中出现。详见 FileAppend(See 16.4)。</p> <p>逗号转义：和其它大多数命令的最后一个参数不同，在 OutputFile 参数中，逗号必须进行转义 (`,)。</p>

注意

当你想对一个文本文件中的内容逐行进行操作的时候，文件读取循环十分有用。它比使用 [FileReadLine\(See 16.18\)](#) 执行地更好，因为：1) 文件在整个操作过程中可以保持打开状态；2) 不必每次都重新扫描文件来查找请求的行数。

内置变量 **A_LoopReadLine** 存在于任何文件读取循环中。它包含了当前行的内容，行尾的回车和换行符 (`r`n) 标记除外。如果一个内层文件读取循环被装在一个外层文件读取循环中，则最里层循环的文件行将享有优先。

每行最多可以读取 65,534 个字符。如果某行的长度有超出，那它剩余的字符将在下次循环中被读取。

在文件读取循环中，经常使用 [StringSplit\(See 27.21\)](#) 或者 [字符串解析循环\(See 17.14\)](#) 来解析从 InputFile 中读取的每一行内容。例如，如果 InputFile 中的每一行内容都是由 tab 分隔的一系列片段，那么这些片段可以用下面这段示例逐个读取：

```
Loop, read, C:\Database Export.txt
{
    Loop, parse, A_LoopReadLine, %A_Tab%
    {
        MsgBox, 第 %A_Index% 个片段是 %A_LoopField%。
    }
}
```

要将整个文件内容读取到一个变量中的话，请用 [FileRead](#)(See 16.19)，因为它比一个循环执行地好得多(特别是读取大文件的时候)。

要同时打开多个文件，请用 [DIIICall\(\)](#)，参考[这个例子](#)(See 18.3)。

请看 [Loop](#)(See 17.12) 命令获取 [Blocks](#)(See 17.2), [Break](#)(See 17.3), [Continue](#)(See 17.4) 以及内置变量 [A_Index](#) (存在于任何类型的循环中) 的相关信息。

相关命令

[FileRead](#)(See 16.19), [FileReadLine](#)(See 16.18), [FileAppend](#)(See 16.4), [Sort](#)(See 27.12), [Loop](#)(See 17.12), [Break](#)(See 17.3), [Continue](#)(See 17.4), [Blocks](#), ([See 17.2](#))[FileSetAttrib](#)(See 16.25), [FileSetTime](#)(See 16.26)

示例

```
;示例 1：第一个文件中，只有含有单词 FAMILY 的行会被写入第二个文件。
;取消第一行的注释的话，将会覆盖现有的文件，而不是将内容追加到现有的文件中。
;FileDelete, C:\Docs\Family Addresses.txt

Loop, read, C:\Docs\Address List.txt, C:\Docs\Family Addresses.txt
{
    IfInString, A_LoopReadLine, family, FileAppend, %A_LoopReadLine%`n
}
```

```
;示例 2：读取一个文本文件的最后一行。
Loop, read, C:\Log File.txt
    last_line := A_LoopReadLine ;当循环结束时，这个变量将会保存最后一行的内容。
```

;示例 3：从一个文本文件或者 HTML 文件中提取所有的 FTP 和 HTTP URL。

```
FileSelectFile, SourceFile, 3,, 选择一个文本文件或者 HTML 文件来分析。  
if SourceFile =  
    return ;未选择文件则退出。  
  
SplitPath, SourceFile,, SourceFilePath,, SourceFileNoExt  
DestFile = %SourceFilePath%\%SourceFileNoExt% Extracted Links.txt  
  
IfExist, %DestFile%  
{  
    MsgBox, 4,, 覆盖已存在的链接文件？按否则是追加。`n`n文件: %DestFile%  
    IfMsgBox, Yes  
        FileDelete, %DestFile%  
    }  
  
LinkCount = 0  
Loop, read, %SourceFile%, %DestFile%  
{  
    URLSearchString = %A_LoopReadLine%  
    Gosub, URLSearch  
}  
 MsgBox 查找到 %LinkCount% 个链接，已写入 "%DestFile%"。  
return  
  
  
URLSearch:  
;下面这么写是因为一些 URL 中嵌入了其它的 URL:  
StringGetPos, URLStart1, URLSearchString, http://
```

```

StringGetPos, URLStart2, URLSearchString, ftp://
StringGetPos, URLStart3, URLSearchString, www.

;找到最左边的开始位置:
URLStart = %URLStart1% ;设置默认的开始位置。

Loop
{
    ;通过只解析 "URLStart%A_Index%" 一次来提高效率 (至少在一个含有许多变量的脚本中能
    ;提高):
    ArrayElement := URLStart%A_Index%
    if ArrayElement = -1 ;抵达数组的末尾。
        break
    if ArrayElement = -1 ;取消这个元素。
        continue
    if URLStart = -1
        URLStart = %ArrayElement%
    else ; URLStart 中包含有效的起始位置, 所以和 ArrayElement 比较。
    {
        if ArrayElement <> -1
            if ArrayElement < %URLStart%
                URLStart = %ArrayElement%
    }
}

if URLStart = -1 ; URLSearchString 中没有 URL。
    return

;否则, 提取这个 URL:
StringTrimLeft, URL, URLSearchString, %URLStart% ;省略开始部分/不相关的内容。

```

```

Loop, parse, URL, %A_Tab%%A_Space%<> ;查找第一个空格, Tab, 或者尖括号 (如果有的话)。

{
    URL = %A_LoopField%

    break ;即只循环一次来获取首个“片段”。

}

;如果没有进入上面的循环, 表示没找到结束符, 将不处理 URL 变量的内容。

;如果 URL 以双引号结尾, 移除它。目前使用了 StringReplace,
;但是要注意, 由于双引号也可以出现在 URL 中, 所以这个操作可能会破坏它们:

StringReplace, URLCleansed, URL, ",, All
FileAppend, %URLCleansed%`n
LinkCount += 1

;检查这一行中是否还有其它 URL:
StringLen, CharactersToOmit, URL
CharactersToOmit += %URLStart%
StringTrimLeft, URLSearchString, URLSearchString, %CharactersToOmit%
Gosub, URLSearch ;循环调用自身。
return

```

翻译: okey3m 修正: 天堂之门 menk33@163.com 2009年1月7日

17.16 Loop (registry)

Retrieves the contents of the specified registry subkey, one item at a time.

Loop, RootKey [, Key, IncludeSubkeys?, Recurse?]

参数

RootKey	Must be either HKEY_LOCAL_MACHINE (or HKLM), HKEY_USERS (or HKU), HKEY_CURRENT_USER (or HKCU), HKEY_CLASSES_ROOT (or HKCR), or HKEY_CURRENT_CONFIG (or HKCC). To access a remote registry, prepend the computer name and a colon, as in
---------	--

	this example: \\workstation01:HKEY_LOCAL_MACHINE
Key	The name of the key (e.g. Software\SomeApplication). If blank or omitted, the contents of <i>RootKey</i> will be retrieved.
IncludeSubkeys?	0 (default) Subkeys contained within <i>Key</i> are not retrieved (only the values). 1 All values and subkeys are retrieved. 2 Only the subkeys are retrieved (not the values).
Recurse?	0 (default) Subkeys are not recursed into. 1 Subkeys are recursed into, so that all values and subkeys contained therein are retrieved.

注意

A registry-loop is useful when you want to operate on a collection registry values or subkeys, one at a time. The values and subkeys are retrieved in reverse order (bottom to top) so that [RegDelete](#)(See 25.2) can be used inside the loop without disrupting the loop.

The following variables exist within any registry-loop. If an inner registry-loop is enclosed by an outer registry-loop, the innermost loop's registry item will take precedence:

A_LoopRegName	Name of the currently retrieved item, which can be either a value name or the name of a subkey. Value names displayed by Windows RegEdit as "(Default)" will be retrieved if a value has been assigned to them, but A_LoopRegName will be blank for them.
A_LoopRegType	The type of the currently retrieved item, which is one of the following words: KEY (i.e. the currently retrieved item is a subkey not a value), REG_SZ, REG_EXPAND_SZ, REG_MULTI_SZ, REG_DWORD, REG_QWORD, REG_BINARY, REG_LINK, REG_RESOURCE_LIST, REG_FULL_RESOURCE_DESCRIPTOR, REG_RESOURCE_REQUIREMENTS_LIST, REG_DWORD_BIG_ENDIAN (probably rare on most Windows hardware). It will be empty if the currently retrieved item is of an unknown type.
A_LoopRegKey	The name of the root key being accessed (HKEY_LOCAL_MACHINE, HKEY_USERS, HKEY_CURRENT_USER, HKEY_CLASSES_ROOT, or HKEY_CURRENT_CONFIG). For remote registry access, this value will not include the computer name.
A_LoopRegSubKey	Name of the current SubKey. This will be the same as the <i>Key</i> parameter unless the <i>Recurse</i> parameter is being used to recursively explore other subkeys. In that case, it will be the full path of the currently retrieved item, not including the root key. For example: Software\SomeApplication\My SubKey

A_LoopRegTimeModified	The time the current subkey or any of its values was last modified. Format <code>YYYYMMDDHH24MISS</code> (See 16.26). This variable will be empty if the currently retrieved item is not a subkey (i.e. <code>A_LoopRegType</code> is not the word KEY) or if the operating system is Win9x (since Win9x does not track this info).
-----------------------	--

When used inside a registry-loop, the following commands can be used in a simplified way to indicate that the currently retrieved item should be operated upon:

RegRead (See 25.3), OutputVar	Reads the current item. If the current item is a key, ErrorLevel will be set to 1 and <i>OutputVar</i> will be made empty.
RegWrite (See 25.4) [, Value]	Writes to the current item. If <i>Value</i> is omitted, the item will be made 0 or blank depending on its type. If the current item is a key, ErrorLevel will be set to 1 and there will be no other effect.
RegDelete (See 25.2)	Deletes the current item. If the current item is a key, it will be deleted along with any subkeys and values it contains.

When accessing a remote registry (via the *RootKey* parameter described above), the following notes apply:

- The target machine must be running the Remote Registry service, or (for Windows Me/98/95) have the Microsoft Remote Registry service installed (this can be obtained from Microsoft).
- Access to a remote registry may fail if the target computer is not in the same domain as yours or the local or remote username lacks sufficient permissions (however, see below for possible workarounds).
- Depending on your username's domain, workgroup, and/or permissions, you may have to connect to a shared device, such as by mapping a drive, prior to attempting remote registry access. Making such a connection -- using a remote username and password that has permission to access or edit the registry -- may as a side-effect enable remote registry access.
- If you're already connected to the target computer as a different user (for example, a mapped drive via user Guest), you may have to terminate that connection to allow the remote registry feature to reconnect and re-authenticate you as your own currently logged-on username.
- For Windows Server 2003 and Windows XP Professional, MSDN states: "If the [registry server] computer is joined to a workgroup and the *Force network logons using local accounts to authenticate as Guest* policy is enabled, the function fails. Note that this policy is enabled by default if the computer is joined to a workgroup."
- For Windows XP Home Edition, MSDN states that "this function always fails". Home Edition lacks both the registry server and client, though the client can be extracted from one of the OS cab files.

See [Loop](#)(See 17.12) for information about [Blocks](#)(See 17.2), [Break](#)(See 17.3), [Continue](#)(See 17.4), and the A_Index variable (which exists in every type of loop).

相关命令

[Loop](#)(See 17.12), [Break](#)(See 17.3), [Continue](#)(See 17.4), [Blocks](#)(See 17.2), [RegRead](#)(See 25.3), [RegWrite](#)(See 25.4), [RegDelete](#)(See 25.2)

示例

```
; Example: Delete Internet Explorer's history of URLs typed by the user:
```

```
Loop, HKEY_CURRENT_USER, Software\Microsoft\Internet Explorer\TypedURLs
    RegDelete
```

```
; Example: A working test script:
```

```
Loop, HKEY_CURRENT_USER, Software\Microsoft\Windows, 1, 1
```

```
{
    if a_LoopRegType = key
        value =
    else
        {
            RegRead, value
            if ErrorLevel
                value = *error*
        }
    MsgBox, 4, , %a_LoopRegName% = %value% (%a_LoopRegType%)`n`nContinue?
    IfMsgBox, NO, break
}
```

```
; Example: A working example to recursively search the entire
; registry for particular value(s).
```

```
SetBatchLines -1 ; Makes searching occur at maximum speed.
```

```
RegSearchTarget = Notepad ; Tell the subroutine what to search for.
```

```
Gosub, RegSearch

return

RegSearch:

ContinueRegSearch = y

Loop, HKEY_LOCAL_MACHINE, , 1, 1

{

    Gosub, CheckThisRegItem

    if ContinueRegSearch = n ; It told us to stop.

        return

}

Loop, HKEY_USERS, , 1, 1

{

    Gosub, CheckThisRegItem

    if ContinueRegSearch = n ; It told us to stop.

        return

}

Loop, HKEY_CURRENT_CONFIG, , 1, 1

{

    Gosub, CheckThisRegItem

    if ContinueRegSearch = n ; It told us to stop.

        return

}

; Note: I believe HKEY_CURRENT_USER does not need to be searched if HKEY_USERS
; is being searched. The same might also be true for HKEY_CLASSES_ROOT if
; HKEY_LOCAL_MACHINE is being searched.

return

CheckThisRegItem:
```

```

if A_LoopRegType = KEY ; Remove these two lines if you want to check key names too.

    return

RegRead, RegValue

if ErrorLevel

    return

IfInString, RegValue, %RegSearchTarget%

{

    MsgBox, 4, , The following match was
found: `n%A_LoopRegKey%\%A_LoopRegSubKey%\%A_LoopRegName%` nValue
= %RegValue%`n`nContinue?

    IfMsgBox, No

        ContinueRegSearch = n ; Tell our caller to stop searching.

}

return

```

17.17 OnExit

Specifies a [subroutine](#)(See 17.8) to run automatically when the script exits.

`OnExit [, Label]`

参数

Label	If omitted, the script is returned to its normal exit behavior. Otherwise, specify the name of the label whose contents will be executed (as a new thread (See 30.19)) when the script exits by any means.
-------	--

IMPORTANT: Since the specified subroutine is called instead of terminating the script, that subroutine must use the [ExitApp](#)(See 17.7) command if termination is desired. Alternatively (as in the Examples section below), the OnExit subroutine might display a [MsgBox](#)(See 19.11) to prompt the user for confirmation -- and only if the user presses YES would the script execute ExitApp to close itself.

注意

The OnExit subroutine is called when the script exits by any means (except when it is killed by something like "End Task"). It is also called whenever the [#SingleInstance](#)(See 22.2) and [Reload](#)(See 20.1.13) commands ask a previous instance to terminate.

A script can detect and optionally abort a system shutdown or logoff via [OnMessage\(0x11, "WM_QUERYENDSESSION"\)](#)(See 18.11).

The OnExit [thread](#)(See 30.19) does not obey [#MaxThreads](#)(See 20.1.6) (it will always launch when needed). In addition, while it is running, it cannot be interrupted by any [thread](#)(See 30.19), including [hotkeys](#)(See 4.), [custom menu items](#)(See 22.13), and [timed subroutines](#)(See 17.21). However, it will be interrupted (and the script will terminate) if the user chooses Exit from the tray menu or main menu, or the script is asked to terminate as a result of [Reload](#)(See 20.1.13) or [#SingleInstance](#)(See 22.2). Because of this, the OnExit subroutine should be designed to finish quickly unless the user is aware of what it is doing.

If the OnExit [thread](#)(See 30.19) encounters a failure condition such as a runtime error, the script will terminate. This prevents a flawed OnExit subroutine from making a script impossible to terminate.

If the OnExit subroutine was launched due to an [Exit](#)(See 17.6) or [ExitApp](#)(See 17.7) command that specified an exit code, that code is ignored and no longer available. A new exit code can be specified by the OnExit subroutine if/when it calls [ExitApp](#)(See 17.7).

Whenever the OnExit subroutine is called by an exit attempt, it starts off fresh with the default values for settings such as [SendMode](#)(See 20.13). These defaults can be changed in the [auto-execute section](#)(See 8.).

The built-in variable **A_ExitReason** is blank unless the OnExit subroutine is currently running or has been called at least once by a prior exit attempt. If not blank, it is one of the following words:

Logoff	The user is logging off.
Shutdown	The system is being shut down or restarted, such as by the Shutdown (See 24.7) command.
Close	The script was sent a WM_CLOSE or WM_QUIT message, had a critical error, or is being closed in some other way. Although all of these are unusual, WM_CLOSE might be caused by WinClose (See 28.14) having been used on the script's main window. To prevent this, dismiss the main window with <i>Send</i> , !{F4}.
Error	A runtime error occurred in a script that has no hotkeys and that is not persistent (See 29.21). An example of a runtime error is Run/RunWait (See 24.5) being unable to launch the specified program or document.
Menu	The user selected Exit from the main window's menu or from the standard tray menu.
Exit	The Exit (See 17.6) or ExitApp (See 17.7) command was used (includes custom menu items (See 22.13)).
Reload	The script is being reloaded via the Reload (See 20.1.13) command or menu item.
Single	The script is being replaced by a new instance of itself as a result of #SingleInstance (See 22.2).

相关命令

[OnMessage\(\)](#)(See 18.11), [RegisterCallback\(\)](#)(See 18.14), [OnClipboardChange](#)(See 30.6),
[ExitApp](#)(See 17.7), [Shutdown](#)(See 24.7), [#Persistent](#)(See 29.21), [Threads](#)(See 30.19),
[Gosub](#)(See 17.8), [Return](#)(See 17.19), [Menu](#)(See 22.13)

示例

```
#Persistent ; For demonstration purposes, keep the script running if the user chooses
"No".

OnExit, ExitSub

return


ExitSub:

if A_ExitReason not in Logoff,Shutdown ; Avoid spaces around the comma in this line.

{

    MsgBox, 4, , Are you sure you want to exit?

    IfMsgBox, No

        return

}

ExitApp ; The only way for an OnExit script to terminate itself is to use ExitApp in the
OnExit subroutine.
```

17.18 Pause

暂停脚本的 [current thread](#)(See 30.19)(当前线程)。

Pause [, On|Off|Toggle, OperateOnUnderlyingThread?]

参数

On Off Toggle	<p>如果留空或省略，它默认为 Toggle。否则，指定为下列一个单词：</p> <p>Toggle: 暂停 current thread(See 30.19) 除非它下面的线程已经暂停，这种情况下将反暂停下面的线程。</p> <p>On: 暂停当前线程。</p> <p>Off: 如果当前线程下面的线程已经暂停，当它恢复时将成为一种反暂停</p>
----------------------	---

	状态。反之，命令无效。
OperateOnUnderlyingThread?	<p>此参数被 "Pause Off" 忽略。对上面参数的其他两种而言，它也被忽略除非暂停已经被开启(包括凭借 Toggle 开启的)。</p> <p>指定下面的一个数字：</p> <p>0 (或者省略): 命令暂停当前线程；也就是，正在运行暂定命令的线程。</p> <p>1: 命令标记当前线程下面的线程为暂定，以便当它恢复时，完成它运行的命令(如果有的话)并且之后进入一个暂停状态。如果当前线程下面没有线程，脚本自己会暂停，这将阻止 timers(See 17.21) 运行(当脚本没有线程时，这个效果和使用了菜单项 "Pause Script" 一样)。</p>

注意

和 [Suspend](#)(See 17.23) 不同 -- 它禁用了 [hotkeys](#)(See 4.) 和 [hotstrings](#)(See 5.) -- pause 将冻结 [current thread](#)(See 30.19)。作为一个副作用，任何当前线程下面被中断的线程也将潜伏着。

在任何线程被暂停时，[timers](#)(See 17.21) 也不会运行。相比之下，明确地启动的线程例如 [hotkeys](#)(See 4.) 和 [menu items](#)(See 22.13) 仍能被运行；但当它们的 [threads](#)(See 30.19) 结束时，下面的线程仍将被暂停。换言之，每个独立于其他的线程能被暂停。

当脚本的 [current thread](#)(See 30.19) 处于一个暂停状态时，托盘图标的颜色从绿色转为红色。这个颜色的改变能够通过冻结图标来避免，其通过为 [Menu](#) 命令的最后一个参数指定 1 来获得。例如：

[Menu](#)(See 22.13), Tray, Icon, C:\My Icon.ico, , 1

要禁用 [timers](#)(See 17.21) 而不暂停脚本，使用 "Thread, NoTimers"(See 22.22)"。

[Pause](#) 命令在功能上和内置菜单项 "Pause Script" 相似。

当一个脚本显示任何种类的 [menu](#)(See 22.13) (tray menu, menu bar, GUI context menu 等等) 时，它总会停住(虽然不是正式地暂停)。

相关命令

[Suspend](#)(See 17.23), [Menu](#)(See 22.13), [ExitApp](#)(See 17.7), [Threads](#)(See 30.19), [SetTimer](#)(See 17.21)

示例

```
Pause::Pause ; 给 "pause" 键指定暂停切换功能...
```

```
#p::Pause ; ... 或为 Win+p 或者其他一些热键指定此功能。
```

翻译：天堂之门 menk33@163.com 2008 年 8 月 12 日

17.19 Return

从一个子程序返回至先前通过 [函数调用\(See 10.\)](#)、[Gosub\(See 17.8\)](#)、[热键\(See 4.\)](#) 激活、[GroupActivate\(See 28.2.1\)](#) 或者其他方法执行的跳转。

Return [, Expression]

参数

Expression	<p>除了当 <code>return</code> 被用在一个 函数(See 10.) 内的时候外，此参数应该被省略。</p> <p>由于此参数是一个 表达式(See 9.)，下面所有的都是有效的例子：</p> <pre>return 3 return "literal string" return MyVar return i + 1 return true ; 返回数字 1 来表示 "true"。 return ItemCount < MaxItems ; 返回一个 true(真) 或者 false(假) 的值。 return FindColor(TargetColor)</pre> <p>已知的限制：为了反向兼容性和易用性，下面两个例子是同等功能的：</p> <pre>return MyVar return %MyVar%</pre> <p>换句话说，单独括在百分号里的变量被看成一个非表达式。要解决这种情况， 通过将其用圆括号括起来使它明确地作为一个表达式； 例如： <code>return (%MyVar%)</code></p>
------------	---

说明

如果没有调用者可返回，`Return` 将做一个 [Exit\(See 17.6\)](#) 动作来代替。

相关命令

[Functions\(See 10.\)](#), [Gosub\(See 17.8\)](#), [Exit\(See 17.6\)](#), [ExitApp\(See 17.7\)](#), [GroupActivate\(See 28.2.1\)](#)

范例

```
#z:::
MsgBox Win-Z 热键已被按下。
```

```
Gosub MySubroutine
```

```
return
```

```
MySubroutine:
```

```
Sleep 1000
```

```
return
```

翻译：天堂之门 menk33@163.com 2008 年 8 月 9 日

17.20 SetBatchLines

决定脚本的执行速度（影响 cpu 占用）。

`SetBatchLines, 20ms`

`SetBatchLines, LineCount`

参数

20ms	(20ms 只是举一个例子。) 如果这个参数以 <code>ms</code> 结尾，它表示脚本每两次休眠之间的时 间间隔（每次休眠 10ms）。在下面的例子中，设置脚本每执行 20ms 之后休眠 10ms： <code>SetBatchLines, 20ms</code>
LineCount	两次休眠之间执行脚本的行数。这个参数最大可以到 9223372036854775807。但是， 这个参数和上面那个参数是互相冲突的；也就是说，同时只有一个参数有效。

注意

使用 `SetBatchLines -1` 让脚本无休眠地执行（换句话说，也就是让脚本全速运行）。

如果没有使用这个命令：

- AutoIt v2 (.aut) 的脚本默认使用 `SetBatchLines 1`，每执行一行脚本休眠一次。
- 其它类型的脚本（例如.ahk）默认使用 `SetBatchLines 10ms`。不过在 v1.0.16 之前的版本中，
默认使用的是 `SetBatchLines 10`。

不论是注重脚本执行速度还是注重 cpu 占用，都推荐使用带“ms”的参数。例如，在大多数的系统中，设置 10ms 的休眠间隔可以让脚本占用不超过 50% 的 cpu 资源。这样不但让脚本快速执行，也可以保留充分的 cpu 资源让其它任务使用，例如游戏或者视频捕捉、回放。

内置变量 **A_BatchLines** 保存了当前设置。

在特定的脚本中，脚本的执行速度同时还受这些命令影响：[SetWinDelay\(See 28.9\)](#)，
[SetControlDelay\(See 28.1.13\)](#)，[SendMode\(See 20.13\)](#)，[SetKeyDelay\(See 20.14\)](#)，
[SetMouseDelay\(See 23.8\)](#) 以及 [SetDefaultMouseSpeed\(See 23.7\)](#)。

每一个新运行的 [Thread/线程](#)(See 30.19) (例如一个 [hotkey/热键](#)(See 4.), [custom menu item/自定义菜单](#)(See 22.13), 或 [timed/定时器](#)(See 17.21) 事件) 会将该命令的设置重置为默认值。要更改该命令的默认值, 可以将该命令放在脚本的自动执行区域 (脚本的顶部)。

相关命令

[SetWinDelay](#)(See 28.9), [SetControlDelay](#)(See 28.1.13), [SendMode](#)(See 20.13),
[SetKeyDelay](#)(See 20.14), [SetMouseDelay](#)(See 23.8), [SetDefaultMouseSpeed](#)(See 23.7),
[Critical](#)(See 22.7)

示例

```
SetBatchLines, 10ms
SetBatchLines, 1000
```

17.21 SetTimer

以一个指定的时间间隔来自动重复地启动子程序。

`SetTimer, Label [, Period|On|Off, Priority]`

参数

Label	欲跳转的标签或 热键标签 (See 4.)的名称, 它使 <i>Label</i> 下面的命令被执行直到遇上 Return (See 17.19)或 Exit (See 17.6)。与几乎所有其他命令的参数一样, <i>Label</i> 可以是一个 变量 (See 9.)引用比如 %MyLabel%, 在这种情况下变量中存储的名称会被作为目标使用。
Period On Off	<p>On: 重新启用一个在它的先前 <i>period</i> (周期)中被禁用的定时器。如果不存在定时器, 则创建它(默认周期为 250)。如果存在定时器但先前被设置成了 run-only-once 模式, 它仍然只运行一次。</p> <p>Off: 禁用一个已存在的定时器。</p> <p>Period: 使用这个参数作为 <i>Label</i> 子程序上一次开始后必须经过的毫秒数来创建或更新一个定时器。在这段时间过后, 将再次运行 <i>Label</i> (除非它仍在继续上一次运行)。定时器将被自动启用。要防止这种情况, 可以随后立即再一次调用此命令, 将此参数指定为 OFF。</p> <p>如果此参数为空并且:</p> <ol style="list-style-type: none"> 1) 定时器不存在: 那么将创建一个周期为 250 的定时器。 2) 定时器已存在: 那么将启用定时器, 并重设它先前的 <i>period</i>, 除非指定了 <i>Priority</i>。 <p>Run only once [v1.0.46.16+]: 用一个负的 <i>Period</i> 来表示定时器只能运行一次。例如, 指定 -100 将在 100ms 后运行定时器, 然后就像使用了 <code>SetTimer, Label, Off</code> 那样禁用定时器。</p>

<p>在vim中之所以可以连续按键两次,是因为 本来所有按键都不是用来在400秒内都不是用来输入的,都是先拦截判断400毫秒内发生了什么事情,只有一个j,就传输一个j;如果是两个j,就命令了.所以输入的时候也不能输入两个这个可选参数是一个介于 -2147483648 和 2147483647 之间的整数(或一个表达式(See 9.)来表示此定时器的线程优先级。如果省略将视为 0。详见线程(See 30.19)。</p>
<p>Priority 要改变一个已有定时器的优先级而不在任何其他方面影响它,只要将此参数前面的那些参数都留空。</p>

注意

拦截了c,在定时子程序中中发送c,因为是另外一个线程了,就会捕获自己的这个案件;只有利用A_PriorHotKey才能在自己的线程中

定时器很有用,因为它们是异步运行的,这意味着即使当脚本正在等待一个窗口,显示一个对话框,或忙于其他任务时,定时器也将按指定的频率(间隔)运行。它们的许多应用实例包括当用户闲置时(如 [%A_TimeIdle%\(See 9.\)](#) 反映的那样)采取某些行动或者在不需要的窗口出现时关闭它们。

尽管定时器会给人以脚本同时运行多个任务的错觉,但实际并非如此。而是定时的子程序被作为其他线程来处理:它们能中断别的线程或被另一个线程打断,比如[热键子程序\(See 4.\)](#)。详见[线程\(See 30.19\)](#)。

翻译为 period时间后开始执行

无论一个定时器是被创建,重新启用,还是更新为一个新的 *period*,它都不会马上运行;它的 *period* 时间必须先到期。如果你想让定时器的首次执行马上开始,可用[Gosub\(See 17.8\)](#)来执行定时器的子程序(不过这种方式不会像定时器自身那样开启一个新线程;所以像[SendMode\(See 20.13\)](#)这样的设置就不会以它们的默认值启动了)。

如果 [SetTimer](#) 被用在一个已存在的定时器上且第二个参数是一个数字或单词 **ON**(或它被省略),定时器内部的“上一次运行时间”将被重置为当前时间;换句话说,必须用完它的整个周期后,定时器才能再次运行。

目前,定时器在 Windows XP/2000/NT 下不能快于 10 ms,Windows 9x 下约为 55 ms。指定小于该值的 *Period* 通常实际的间隔结果为 10 或 55(但此策略可能在将来版本中改变,故无需盲从)。

定时器在下列情况中可能无法按指定频率运行:

- 其它应用程序使 CPU 负担过重。
- 定时器的子程序本身花费了比它的周期更长的时间来运行,或有太多相抵触的定时器(调整[SetBatchLines\(See 17.20\)](#)可能会有所帮助)。
- 定时器被另一个线程(See 30.19),即另一个定时的子程序、热键子程序(See 4.)或者自定义菜单项(See 22.13)打断(这可以通过[Critical\(See 22.7\)](#)避免)。如果发生了这种情况且打断线程花了很长时间才结束,那么实际上被打断的定时器在这段时间中将被禁用。不过,其它的定时器对打断首个定时器的线程(See 30.19)进行中断,仍可继续运行。
- 脚本由于[Critical\(See 22.7\)](#)或[Thread Interrupt/Priority\(See 22.22\)](#)而不可中断。在此时间段内,定时器不会运行。之后,当脚本可再次中断时,延误的定时器会尽快运行一次然后恢复到它的正常运作。

由于定时器靠临时中断脚本的当前活动来运作,所以它们的子程序应保持简短(以便它们迅速完成),无论何时都不适宜有很长的中断。

一个脚本运行期间持续生效的定时器通常应在[自动执行部分\(See 8.\)](#)创建。相比之下,临时定时器经常会被它的子程序禁用(请看本页底部的示例)。

每当一个定时的子程序运行时,它的设置比如[SendMode\(See 20.13\)](#)都会以默认值重新开始。这些默认值可在[自动执行部分\(See 8.\)](#)修改。

虽然定时器会在脚本挂起(See 17.23)时运作，但如果当前线程(See 30.19)的 "Thread NoTimers(See 22.22)" 正生效或者每当线程被暂停(See 17.18)时，它们是不会运行的。另外，在用户浏览脚本的某个菜单时（如托盘图标菜单或菜单栏）定时器也不会运作。

如果热键(See 4.)的响应时间很重要（比如游戏中）且脚本包含的定时器子程序执行时间会超过 5 ms，那么可以使用如下命令来避免 15 ms 的延迟。否则如果热键正好在一个定时器线程处于它的不可中断周期时被按下，这个延迟就会发生：

`Thread(See 22.22), interrupt, 0 ;使所有线程始终可中断。`

如果定时器在它的子程序正运行时被禁用，该子程序会继续运行直到完成。

[KeyHistory\(See 20.9\)](#) 功能会显示存在多少个定时器，当前启用了多少个。

定时器的周期不可大于 4294967295 毫秒(49.7 天)。

要让脚本持续运行，比如那些只包含了定时器的脚本，请用 [#Persistent\(See 29.21\)](#)。

相关命令

[Gosub\(See 17.8\)](#), [Return\(See 17.19\)](#), [Threads\(See 30.19\)](#), [Thread \(command\)\(See 22.22\)](#),
[Critical\(See 22.7\)](#), [IsLabel\(\)\(See 10.\)](#), [Menu\(See 22.13\)](#), [#Persistent\(See 29.21\)](#)

示例

;例 1：当不需要的窗口出现时将其关闭：

```
#Persistent

SetTimer, CloseMailWarnings, 250

return

CloseMailWarnings:

WinClose, Microsoft Outlook, A timeout occurred while communicating
WinClose, Microsoft Outlook, A connection to the server could not be established

return
```

;例 2：等待一个特定的窗口出现，然后通知用户：

```
#Persistent

SetTimer, Alert1, 500

return
```

```

Alert1:

IfWinNotExist, Video Conversion, Process Complete

    return

;否则:

SetTimer, Alert1, Off ;即定时器在此处将自己关闭。

SplashTextOn, , , 视频转换已完成。

Sleep, 3000

SplashTextOff

return

```

;例 3: 检测热键的单击、双击和三次按击。

;这允许热键可以按你按键的次数来执行不同的操作:

```

#c:::

if winc_presses > 0 ; SetTimer 已经启动, 所以我们记录按键。

{

    winc_presses += 1

    return
}

;否则, 这是新一系列按键的首次按键。将计数设为 1 并启动定时器:

winc_presses = 1

SetTimer, KeyWinC, 400 ;在 400 毫秒内等待更多的按键。

return

;

KeyWinC:

SetTimer, KeyWinC, off

if winc_presses = 1 ;该键已按过一次。

{

    Run, m:\ ;打开一个文件夹。
}

```

```

else if winc_presses = 2 ;该键已按过两次。
{
    Run, m:\multimedia ;打开一个不同的文件夹。
}
else if winc_presses > 2
{
    MsgBox, 检测到三次或更多次点击。
}
;不论上面哪个动作被触发, 将计数复位以备下一系列的按键:
winc_presses = 0
return

```

翻译: bu45gande 修正: 天堂之门 menk33@163.com 2008 年 12 月 18 日

17.22 Sleep

在继续前等待指定的时间量。

Sleep, DelayInMilliseconds

参数

Delay	要停顿的时间量(以毫秒形式)在 0 和 2147483647 (24 天) 之间, 其可以是一个 表达式 (See 9.)。
-------	--

说明

由于操作系统的时间控制系统的间隔尺寸, *Delay* 典型地上舍入为最临近 10 的倍数。例如, 在大多数 Windows NT/2000/XP 系统上, 一个在 1 和 10 (包含的)之间的 *delay* 相当于 10。不过, 由于硬件差异, 一些系统将上舍入为一个不同的值像 15。

如果 CPU (中央处理器)处于负担状态, 实际延迟时间可能比它请求的要更久结束。这是因为操作系统在给脚本另一个时间片之前, 给每个有需要的进程一个 CPU 时间片(典型地有 20 毫秒)。

一个为 0 的 *delay* 让出脚本当前的时间片的剩余给任何其他需要它的进程(只要它们不在 [priority](#)(See 24.4)(优先权)上比脚本显著地较低)。因此, 一个为 0 的 *delay* 产出一个在 0 和 20ms (或更久)之间的实际延迟, 取决于有需要的进程的数量 (如果无有需要的进程, 也就根本没有 *delay*)。不过, 一个为 0 的 *Delay* 将总是比任何更长的 *Delay* 结束得更早。

当脚本停顿时, 新的 [threads](#)(See 30.19)(线程) 能通过 [hotkey](#)(See 4.)、[自定义菜单项](#)(See 22.13) 或 [timer](#)(See 17.21) 被运行。

"Sleep -1": 一个为 -1 的 **delay** 不会停顿，而是让脚本立即检查它的消息队列。这能被用来强制任何待定的 [中断\(See 30.19\)](#) 发生在一个特定的地方，而不是更随机的某处。详见 [Critical\(See 22.7\)](#)。

相关命令

[SetKeyDelay\(See 20.14\)](#), [SetMouseDelay\(See 23.8\)](#), [SetControlDelay\(See 28.1.13\)](#),
[SetWinDelay\(See 28.9\)](#), [SetBatchLines\(See 17.20\)](#)

范例

```
Sleep, 1000 ; 1 秒
```

翻译: 天堂之门 menk33@163.com 2008 年 7 月 24 日

17.23 Suspend

禁用或启用所有的或是选择的 [热键\(See 4.\)](#)。

Suspend [, Mode]

参数

Mode	<p>On: 挂起所有的热键，除了那些在下面注意部分做出解释的外。</p> <p>Off: 重新启用所有的热键。</p> <p>Toggle (默认): 变更为它先前的相反状态 (On 或 Off)。</p> <p>Permit: 除了把当前的子程序标记为免除挂起外，什么也不做。</p>
------	--

注意

任何第一行有 **Suspend** (除了 "Suspend On") 的热键子程序将被免于挂起。换句话说，热键在 **Suspend** 为 ON 的状态下仍将可用。这就允许了通过这样的一个热键来关掉挂起。

要根据当前窗口的类型自动地禁用选择的热键或热字符串，请用 [#IfWinActive/Exist\(See 20.1.4\)](#)。

挂起一个脚本的热键不会终止脚本已在运行的 [线程\(See 30.19\)](#) (如果有的话)；请用 [Pause\(See 17.18\)](#) 命令来终止。

当脚本的热键被挂起时，它的托盘图标变为字母 **S**。可以通过冻结图标来避免，只要将 **Menu** 命令的最后一个参数指定为 **1** 即可。例如：

```
Menu(See 22.13), Tray, Icon, C:\My Icon.ico, , 1
```

如果脚本被挂起，内置变量 **A_IsSuspended** 将包含 **1**，否则为 **0**。

相关命令

[#IfWinActive/Exist](#)(See 20.1.4), [Pause](#)(See 17.18), [Menu](#)(See 22.13), [ExitApp](#)(See 17.7)

示例

```
^!s::Suspend ; 给一个热键指定挂起的开关功能。
```

翻译：天堂之门 menk33@163.com 2008 年 9 月 8 日

17.24 While-loop

重复地执行一系列命令直到指定的 [expression](#)(See 9.)(表达式) 值为 `false`(假).

While Expression

参数

Expression	任何合法的 expression (See 9.)(表达式)。例如: <code>while x < y</code>
------------	--

注意

每次循环之前表达式的值都会被计算一次。如果表达式的值为 `true`(真) (即除了空字符串或者数值 0 之外的任何其他结果)，那么循环体将被执行；否则，脚本跳至循环体之后一行执行。

一个 `while-loop` 之后通常有一个 [block](#)(See 17.2)(块)，它是循环 *body*(体) 语句的集合。不过，循环仅有一条语句时不需要一个块(一个 "if" 以及它的 "else" 用作这种目的算作一条语句)。

One True Brace (OTB) 编码风格可以被使用，它允许开括号出现在语句的同一行而不是下面。例如: `while x < y {`

内置变量 **A_Index** 记录了当前循环执行的次数。循环表达式和循环体第一次执行时它为 1。第二次执行时它为 2；依次类推。如果一个内部循环在一个外部循环中，内部循环优先使用 **A_Index**。**A_Index** 在任何类型的循环内部工作，但是在循环外它被置为 0。

和所有循环相同，[break](#)(See 17.3) 可以被用来提前结束循环。还有，[Continue](#)(See 17.4) 可以被用来跳过当前循环的剩余部分。这时 **A_Index** 加 1，循环表达式被重新计算。如果它认为 `true`(真)，新的循环开始；否则循环结束。

专用循环: 循环可以被用于自动获取文件，文件夹或者注册表项(一次一个)。详见 [file-loop](#)(See 16.31) 和 [registry-loop](#)(See 17.16)。另外，[file-reading loops](#)(See 16.32) 能够操作一个文件的所有内容，一次一行。最后，[parsing loops](#)(See 17.14) 能够操作一个被分隔字符串中的独立部分。

相关命令

[Break](#)(See 17.3), [Continue](#)(See 17.4), [Blocks](#)(See 17.2), [Loop](#)(See 17.12), [Files-and-folders loop](#)(See 16.31), [Registry loop](#)(See 17.16), [File-reading loop](#)(See 16.32), [Parsing loop](#)(See 17.14), [If \(expression\)](#)(See 17.11)

示例

; 当用户拖拽鼠标时，一个 ToolTip(提示框) 在拖拽区域内显示区域的大小。

CoordMode, Mouse, Screen

~LButton::

```
MouseGetPos, begin_x, begin_y
while GetKeyState("LButton")
{
    MouseGetPos, x, y
    ToolTip, % begin_x ", " begin_y "`n" Abs(begin_x-x) " x " Abs(begin_y-y)
    Sleep, 10
}
ToolTip
return
```

18. 函数

- 介绍以及简单的例子
- 参数
- 可选参数
- 局部变量
- 优化布尔求值
- 在函数中使用子程序
- **Return, Exit** 和一般说明
- 用 **#Include** 在多个脚本中共享函数
- 函数库：标准库和用户库
- 内置函数

函数和子程序([Gosub](#)(See 17.8))相似，只不过它能从它的调用者那里接受参数(输入)。此外，函数能随意地向它的调用者返回一个值。参考下面这个简单的函数，它接受了两个数字并返回它们的和：

```
Add(x, y)
{
    return x + y ; "Return(See 17.19)" 需要一个表达式(See 9.)
}
```

上面的被称为**函数定义**, 因为它创建了一个名为 "Add"(不区分大小写)的函数, 并且确定无论谁要调用它都必须确切地提供两个参数(x 和 y)。要调用函数并将它的结果赋值给变量, 用 **:=(See 21.12)****运算符**(See 21.12)。例如:

```
Var := Add(2, 3) ;数字 5 将存储在变量 var 中。
```

而且, 可以调用函数而存储它的返回值:

```
Add(2, 3)
```

但是如果这样, 任何通过函数返回的值都将被抛弃; 因此除非函数除了它的返回值外还会产生其他一些效果, 不然此调用将无济于事。

一旦函数在**表达式**(See 9.)中被调用, 任何在它参数列表中的变量名都不能被附上百分号。相反, 原义字符串却要加上双引号。例如:

```
if InStr(MyVar, "fox")
    MsgBox MyVar 变量包含单词 fox。
```

在 v1.0.47.06 及之后版本里, 一个函数(甚至是**内置函数**)可以通过百分号被动态地调用。例如, **%Var%(x, "fox")** 将调用一个名称存储在变量 Var 里的函数。相似地, **Func%A_Index%()** 将调用名称存储在指定数组元素(See 30.5)中的函数。所调用的函数必须在脚本中明确地定义, 也可通过 **#Include**(See 17.1)或者非动态调用一个**函数库**。如果被调用的函数不存在, 或者给函数参数传递的值或类型不对, 那么包含此调用的表达式将产生一个空字符串。

最后, 函数可以被任何命令的参数调用 (除了像 **StringLen**(See 27.17) 命令中的那些 **OutputVar** 和 **InputVar** 参数)。不过, 那些不支持**表达式**(See 9.)的参数必须使用 "%"作为前缀, 例如下面这个例子:

```
MsgBox % "答案是: " . Add(3, 2)
```

"%"前缀也允许用在本来就支持表达式的参数里, 但它将被简单地忽略。

当一个函数被定义的时候, 它的参数都列在它的名称后面的括号中(它的名称和左括号之间不能有空格)。如果一个函数不接收任何参数, 就让括号空着; 例如: **GetCurrentTimestamp()**。

ByRef 参数: 从函数的观点来看, 参数本质上和**局部变量**(See 10.)一样, 除非它们像在下面这个例子中那样被定义为 **ByRef**:

```
Swap(ByRef Left, ByRef Right)
{
    temp := Left
    Left := Right
```

```
Right := temp
}
```

在上面的例子中，**ByRef** 的使用导致各个参数变成了从调用者传递进来的变量的一个别名，换句话说，参数和调用者的变量在内存中都引用了相同的内容。这就允许了 **Swap** 函数通过移动 **Left** 的内容给 **Right** 来改变调用者的变量，反过来也一样。

相比之下，如果在上面的例子中没有使用 **ByRef** 的话，**Left** 和 **Right** 将是调用者的变量的复制，因此 **Swap** 函数也将没有效果。

由于 **return**(See 17.19) 只能送回一个值给函数的调用者，所以 **ByRef** 可以用来送回更多的结果。这是由函数向调用者传递进来的变量(通常为空)储存一个值来实现的。

当向函数传递大量字符串的时候，**ByRef** 可以提高性能并且通过回避复制字符串的需求来节约内存。同样，使用 **ByRef** 送回一个长字符串给调用者往往比例如 **Return** 巨型字符串执行得更好。

已知限制：

- 不能向函数的 **ByRef** 参数传递剪贴板(See 30.6)，内置变量(See 9.)或者环境变量(See 9.)，甚至当 **#NoEnv**(See 29.19) 在脚本里不存在的时候。传递一个内置变量给一个 **ByRef** 参数将导致显示一个错误对话框。
- 尽管一个函数能递归地调用自身，但如果它传递它自己的一个局部变量(See 10.)或者非 **ByRef** 参数给它自身的 **ByRef**，新一层的 **ByRef** 参数将引用它自身那个局部变量的名称而不是之前的层的。不过，这个矛盾在一个函数传递一个全局变量，静态变量或者 **ByRef** 参数给它自身的时候不会发生。
- 如果一个参数在函数调用里被解析为一个变量（比如 **Var** 或 **++Var** 或 **Var* =**），它左边或右边的其他参数能在它被传递给函数前改变那个变量。比如，当 **Var** 最初为 **0** 时，即使当函数的首个参数不是 **ByRef** 时，**func(Var, Var++)** 仍会意外地传递 **1** 和 **0**。由于这种行为是违反常规的，所以可能在将来放出的版本中改变。

在定义一个函数时，它的一个或多个参数可以被标记为可选。这可以通过给它们添加一个等号和一个默认值来实现。下面的函数的参数 **Z** 已标记为可选：

```
Add(X, Y, Z = 0)
{
    return X + Y + Z
}
```

当调用者传递满三个参数给上面的函数时，**Z** 的默认值将被忽略。但是当调用者仅传递了两个参数时，**Z** 自动地获取默认值 **0**。

不能将可选参数孤立在参数列表的中间。换句话说，位于首个可选参数右边的所有参数也必须标记为可选。

在 v1.0.46.13 及之后版本中，**ByRef** 参数也可支持默认值；例如：**Func(ByRef p1 = "")**。每当调用者省略了这样一个参数，函数会创建一个局部变量并赋上默认值；换句话说，函数会表现得像没有 **ByRef** 这个关键词一样。

一个参数的默认值必须是下列形式之一: **true**, **false**, 一个原义的整数, 一个原义的浮点数, 或一个引用的/原义的字符串例如 "fox" 或 "" (但在 1.0.46.13+ 之前的版本只允许 "")。

局部变量

所有在函数内部引用或创建的变量默认都是**局部的** (除了 [Clipboard\(See 30.6\)](#), [ErrorLevel\(See 30.8\)](#) 和 [A_TimeIdle\(See 9.\)](#) 这些内置变量)。每个局部变量的内容都只能在函数内可见。所以, 一个局部变量可以和一个全局变量有着相同的名字却有着不同的内容。最后, 所有的局部变量每次在函数被调用时都以空值开始。

全局变量

要在在一个函数里引用一个存在的全局变量(或创建一个新的), 需要在使用它之前先声明此变量为 **global**。例如:

```
LogFile(TextToLog)
{
    global LogFileName ;这个全局变量先前已经在此函数外赋过值了。
    FileAppend, %TextToLog%`n, %LogFileName%
}
```

如果一个函数需要引用或创建大量的全局变量, 可以通过将它的首行设为单词"**global**"或者声明一个局部变量来假设它所有的变量都是全局的(它的参数除外)。例如:

```
SetDefaults()
{
    global ;如果在此函数的首行有比如"local MyVar"这样的词, 那么这个单词可以被省略。
    Var := 33 ;将 33 赋值给一个全局变量, 如果需要, 首次可创建变量。
    local x, y:=0, z ;局部变量必须用这种形式来声明, 不然它们会被假定为全局变量。
    ; ...等等。
}
```

这种假设的全局模式也可以被函数用来创建一个全局数组([See 30.5](#)), 例如一个赋值给 **Array%A_Index%** 的循环。

静态变量

一个变量可以被声明为 **static** 来使它的值在多次调用期间被记住。例如:

```
LogFile(TextToLog)
{
    static LineCount = 0
    LineCount += 1 ;保持自身的累加(它的值在多次调用期间能被记住)。
    global LogFileName
```

```
FileAppend, %LineCount%: %TextToLog%`n, %LogFileName%
}
```

静态变量一般都是隐式的局部变量。在 1.0.46 之前的版本，所有的静态变量都以空值开始；所以要检查一个静态变量首次被使用的唯一办法就是检查它是否为空值。在 v1.0.46 及之后的版本，一个静态变量可以初始化为除了 "" 外的东西，通过其后跟 := 或 = 以及下列之一：true, false, 一个原义的整数，一个原义的浮点数，或一个引用的/原义的字符串比如 "fox"。例如：static X:=0, Y:="fox"。每个静态变量都只初始化一次(在脚本执行之前)。

关于局部和全局的更多信息：

在下面例子中，通过逗号将多种变量分隔开从而在同一行声明：

```
global LogFileName, MaxRetries := 5
static TotalAttempts = 0, PrevResult
```

在 v1.0.46 及之后版本，一个局部或者全局变量可以在同一行被初始化，通过在它的声明后加上 := 或 = 以及任何[表达式\(See 9.\)](#)(在声明中运算符 = 和 := 作用相同)。在一个特定行有多个声明时，由于[性能原因\(See 9.\)](#)(不像普通的[逗号分隔语句\(See 9.\)](#))每个有初始化设定的声明都被作为单独的行执行。与[静态变量初始化设定](#)不同，局部变量和全局变量的初始化设定在每次函数被调用时都会执行，但是仅当控制流实际能到达它们这时。换句话说，在一行中写下 local x = 0 和在两个单独的行写下 local x 以及 x = 0 的作用是一样的。

因为单词 *local*, *global* 和 *static* 在脚本启动时会被立即处理，所以不能用 [IF 语句\(See 17.11\)](#)来有条件地声明变量。换句话说，在一个 IF 或 ELSE [块\(See 17.2\)](#)里的声明，声明与函数末尾大括号之间的所有行都会无条件地生效。同时注意目前还不可能声明一个动态变量比如 *global Array%i%*。

在一个函数里，任何动态变量引用比如 *Array%i%* 常常会解析为一个局部变量，除非那个名称的变量不存在，在这时才会使用全局变量如果它存在的话。如果两者都不存在，那么需要变量先被创建才能使用，它会被创建为一个局部变量，除非[假设全局模式](#)已生效。因此，仅当函数被定义为[假设全局](#)函数时，它才能够手动地创建一个全局数组([See 30.5](#))(比如使用 *Array%i% := A_Index* 这样的方法)。

对于创建数组([See 30.5](#))的命令(例如 [StringSplit\(See 27.21\)](#))，如果[假设全局模式](#)未生效或者数组的第一个元素已声明为局部变量(将函数的一个参数传递给它也可以 -- 即使那个参数是 [ByRef\(See 10.\)](#) 的 -- 因为参数和局部变量很相似)，那么得出的数组就是局部的。相反的，如果首个元素已被[声明为全局](#)，那么创建的是全局数组。[StringSplit\(See 27.21\)](#) 的首个元素是 *ArrayName0*。对于其他创建数组的命令比如 [WinGet List\(See 28.15\)](#)，首个元素是 *ArrayName*(即没有数字)。

常见疑点：在脚本启动时对变量的任何非动态引用都将创建那个变量。例如：在脚本启动的时候，在函数外使用 *MsgBox %Array1%* 将创建全局的 *Array1*。相反的，在脚本启动时，在函数内部使用 *MsgBox %Array1%* 将创建 *Array1* 作为函数局部变量之一(除非[假设全局](#)已生效)。

当在[表达式\(See 9.\)](#)中使用 *AND*, *OR* 和[三元运算符\(See 9.\)](#)时，他们为提高性能而优化(不管当前是否有函数调用)。通过拒绝计算一个表达式里那些不能影响它的最终结果的部分来实行优化操作。要阐明此观点，请看下例：

```
if (ColorName <> "" AND not FindColor(ColorName))
    MsgBox 没找到 %ColorName%.
```

在上例中，如果 *ColorName* 变量为空，`FindColor()` 函数永远不会被调用。这是因为 *AND* 左侧的结果将为 *false*，因此它的右侧不可能让最终结果输出为 *true*。

由于此特性，所以必须明白，如果在 *AND* 或 *OR* 右侧调用函数，函数可能永远会产生任何副作用(例如改变一个全局变量的内容)。

同时也要注意在嵌套的 *AND* 和 *OR* 中串联的求值优化。例如，在下面的表达式里，每当 *ColorName* 为空，就只需最左侧的比较就能执行了。这是因为左侧的表达式已经能确保最终的结果：

```
if (ColorName = "" OR FindColor(ColorName, Region1) OR FindColor(ColorName,
Region2))
    break ;搜索内容为空，或者已经有一个匹配。
```

从上面例子来看，任何耗时的函数一般都应该在 *AND* 或 *OR* 的右侧调用从而提高性能。这个技术还能用来阻止函数的某个参数在传递一个不恰当的值比如一个空字符串时函数被调用。

在 v1.0.46 及之后的版本，[三重条件运算符 \(?:\)](#)(See 9.) 也通过不计算丢失的分支来优化求值。

尽管一个函数不能包含其他函数的[定义](#)，但它可以包含子程序。与其他子程序一样，可使用 [Gosub](#)(See 17.8) 来启动它们，[Return](#)(See 17.19) 来返回结果(这时候 `Return` 属于 `Gosub` 而不是函数)。

已知限制：当前，在整个脚本中每个子程序的名称(标签)必须是独一无二的。如果存在重复的标签，程序将会通知你。

如果一个函数使用 [Gosub](#)(See 17.8) 跳转到一个公共子程序(在函数括号以外的子程序)，那么在外部的所有变量都是全局变量，而函数自己的[局部变量](#)在子程序返回之前将不可用。

虽然一般不鼓励使用 [Goto](#)(See 17.9) 命令，但是它能用来在同个函数内跳转到函数的其他位置。这能帮助我们简化很多返回点的复杂函数，所有那些返回点需要在返回之前做一些清理。

尽管 [Goto](#)(See 17.9) 那些函数外部的目标被忽略，但是对函数来说可以用 [Gosub](#)(See 17.8) 进入一个外部的/公用的子程序然后从那里使用 `Goto`。

一个函数可以包含从外部调用的子程序，比如[定时器](#)(See 17.21), [GUI](#)(See 19.3) 和[菜单项](#)(See 22.13)。通常将它们封装在不同的文件中供 [#Include](#)(See 17.1) 使用，这将防止它们干扰脚本的[自动执行部分](#)(See 8.)。不过，还有如下限制：

- 如果它们的函数是正常地调用，这样的子程序只能用[静态变量](#)和[全局变量](#)(不能是[局部变量](#))。这是因为一个子程序[线程](#)(See 30.19)打断一个函数调用线程(或者反过来也一样)将能够改变被打断的线程看到的局部变量的值。此外，每当函数返回它的调用者时，它所有的局部变量都被清空从而释放它们的内存。
- 这样的子程序应该只将[全局变量](#)(See 10.)(不是[静态变量](#)(See 10.))用作 [GUI 控件变量](#)(See 19.3)。
- 当一个子程序[线程](#)(See 30.19)进入一个函数时，任何由此线程引用的[动态变量](#)(See 30.5)将被作为[全局变量](#)对待(包括创建数组的命令)。

如果一个函数内部的执行流在到达函数的结束大括号之前遇上一个 [Return\(See 17.19\)](#)，那么函数将结束并返回一个空值(空字符串)给它的调用者。每当函数明确地省略 [Return\(See 17.19\)](#) 的参数时也将返回一个空值。

当一个函数使用 [Exit \(See 17.6\)](#)命令来结束[当前线程\(See 30.19\)](#)的时候,它的调用者根本不会收到返回值。例如：在语句 `Var := Add(2, 3)` 中如果 `Add()` 退出，那么 `Var` 将不做改变。如果函数遇上运行错误比如[运行\(See 24.5\)](#)了一个不存在的文件(当 [UseErrorLevel\(See 24.5\)](#) 无效时)，那么也将发生同样的事情。

一个函数可以改变 [ErrorLevel\(See 30.8\)](#) 的值为了返回一个更容易记忆的额外值。

要在调用函数时使用一个或多个空值(空字符串)，可以像这个例子一样使用一对空的双引号：

```
FindColor(ColorName, "")
```

因为调用函数不会启用新的[线程\(See 30.19\)](#)，所以函数做出的任何设置比如 [SendMode\(See 20.13\)](#) 和 [SetTitleMatchMode\(See 28.8\)](#) 的改变也都会影响到它的调用者。

一个函数的调用者可能传递一个不存在的变量或者[数组\(See 30.5\)](#)元素给它，这在函数期望这个相应的参数被 [ByRef](#) 时将变得很有用。例如：调用 `GetNextLine(BlankArray%i%)` 将自动地创建一个[局部](#)或全局变量 `BlankArray%i%` (根据调用者是否在函数内并且是否有[假设的全局模式](#)在起作用)。

当在一个函数里使用时，[ListVars\(See 22.12\)](#) 会显示一个函数的[局部变量](#)和它们的内容。这能帮助我们调试脚本。

你会发现如果给复杂的函数的特定变量加一个独特的前缀，将使函数更易于阅读和维护。例如，用 "p" 或 "p_" 开头来命名函数的每个参数能让你一眼就辨别出它们的特性，特别是当函数还有很多[局部变量](#)来争着吸引你眼球的时候。相似地，"r" 或 "r_" 可以用在 [ByRef 参数](#)的前面，"s" 或 "s_" 可以用在[静态变量](#)的前面。

[单个正确的大括号\(OTB\)](#) [类型\(See 17.2\)](#)也可以用来定义函数。例如：

```
Add(x, y) {
    return x + y
}
```

可以用 [#Include\(See 17.1\)](#) 指令(甚至在脚本的顶部)从外部文件加载函数。

说明：当脚本的执行流遇到函数定义时，它会跳过函数(用一种瞬间完成的方法)，并在它的结束大括号的下一行恢复执行。所以执行流不会从上面掉进一个函数，也不会因为在脚本的顶部有一个或多个函数而影响到[自动执行部分\(See 8.\)](#)。

一个脚本不一定非要使用 [#Include](#)(See 17.1) 来调用外部文件中的函数。要达到这个目的，一个与函数同名的文件必须存在于下列的库目录之一中：

`%A_MyDocuments%(See 9.)\AutoHotkey\Lib\`; 用户库。此目录是可选的；也可以完全不存在。

当前运行的 `AutoHotkey.exe` 的路径`\Lib\`; 标准库。同样是可选的。

例如，假设一个脚本调用一个不存在的函数 `MyFunc()`，程序将在用户库里搜索名为 `MyFunc.ahk` 的文件。如果找不到，它将在标准库里继续搜索。如果仍未找到并且函数的名字里有一个下划线（比如 `MyPrefix_MyFunc`），那么程序将在两个库里搜索名为 `MyPrefix.ahk` 的文件，如果存在的话会加载它。这使得 `MyPrefix.ahk` 可以包含函数 `MyPrefix_MyFunc` 以及其他相关的函数名以 `MyPrefix_` 开头的函数。

虽然一个库文件通常只包含一个和它的文件名同名的函数，但它也可以包含仅被同名函数调用的私有函数和子程序。然而，这些函数应该具有相当独特的名称，因为它们仍会在全局命名空间里；也就是说，可以从脚本的任意位置调用它们。

如果一个库函数使用 [#Include](#)(See 17.1)，那么 `#Include` 的工作目录就是库函数自身所在的目录。这能被用来创建一个重定向到一个包含此函数和其他与之相关的函数的巨大的库文件。

脚本编译器 (`ahk2exe`)(See 8.)同样支持库函数。不过，它需要编译器目录的上层目录里存在一个复制的 `AutoHotkey.exe`(通常已是如此)。如果 `AutoHotkey.exe` 不存在，编译器仍可以运行，不过无法自动地包含库函数。

包含一个库函数也会执行地同其他函数一样好，因为它们在脚本开始执行之前已经被预加载了。

在一个内置函数的参数列表末尾的任何可选的参数是可以被完全省略的。例如，`WinExist("Untitled - Notepad")` 是有效的，因为它的另外三个参数将被视为空。

如果脚本定义了一个和内置函数同名的函数，内置函数将被覆盖。例如：一个脚本将调用它自定义的 `WinExist()` 函数来代替标准的那个。

存在于 DLL 文件里的外部函数可以通过 [DIICall\(\)](#)(See 18.3) 来调用。

经常使用的函数

FileExist(FilePattern): 如果 `FilePattern` 不存在(如果未指定绝对路径，程序将假设 `FilePattern` 在 `A_WorkingDir`(See 9.))就返回一个空值(空字符串)。否则，它将返回首个匹配的文件或文件夹的属性字符串(See 16.10)("RASHNDOCT"的子集)。如果文件没有属性(罕见)，将返回 "X"。`FilePattern` 可以是文件或文件夹的准确的名称，或者包含通配符 (*) 或 (?)。因为空字符串被视为 "false"，所以函数的返回值总是可以作为一个准布尔值来使用。例如：如果文件存在，语句 `if FileExist("C:\My File.txt")` 将为 true，反之为 false。相似地，只有当文档存在并且是一个目录，语句 `if InStr(FileExist("C:\My Folder"), "D")` 才为 true。相关命令：[IfExist](#)(See 16.27) 和 [FileGetAttrib](#)(See 16.10)。

GetKeyState(KeyName(Sec 7.) [, "P" or "T"]): 与 `GetKeyState` 命令(See 20.7)不同，后者是按下的返回 D，弹起的返回 U；而这个函数是如果键是按下的就返回 true (1)，弹起的返回 false (0)。

如果参数 [KeyName](#)(See 7.) 是无效的，将返回空字符串。请看 [GetKeyState](#)(See 20.7) 来了解其他返回值和用法。

InStr(Haystack, Needle [, CaseSensitive = false, StartingPos = 1]): 返回 *Haystack* 字符串中首个匹配的 *Needle* 字符串的位置。与 [StringGetPos](#)(See 27.14) 不同，第一个字符的位置是 1；这是因为 0 同义于 "false"，会使其成为一个直观的"未找到"的指示。如果参数 *CaseSensitive* 被省略或者为 *false*，则搜索将不区分大小写(不区分的模式取决于 [StringCaseSense](#)(See 27.13) 命令)；否则就得精确地匹配大小写。如果省略 *StartingPos*，其默认为 1(*Haystack* 字符串的起点)。否则，指定 2 从 *Haystack* 的第二个字符开始搜索，3 从第三个开始，等等。如果 *StartingPos* 超出 *Haystack* 的长度，那么函数将返回 0。如果 *StartingPos* 为 0，那么搜索将逆序执行(从右到左)以便找到在最右边匹配的结果。不管 *StartingPos* 的值是多少，返回的位置将始终相对于 *Haystack* 的首个字符而言。例如：在 "123abc789" 中 "abc" 的位置永远是 4。相关链接：[RegExMatch\(\)](#)(See 18.12), [IfInString](#)(See 27.3) 和 [StringGetPos](#)(See 27.14)。（译注：例如 InStr("123abc789","abc",false,1)）

RegExMatch(Haystack, NeedleRegEx [, UnquotedOutputVar = "", StartingPos = 1]):
请看 [RegExMatch](#)(See 18.12)()

RegExReplace(Haystack, NeedleRegEx [, Replacement = "", OutputVarCount = "", Limit = -1, StartingPos = 1]): 请看 [RegExReplace](#)(See 18.13)()

SubStr(String, StartingPos [, Length]) [v1.0.46+]: 在 *String* 字符串中从 *StartingPos* 起始点开始向右复制不超过 *Length* 长度的字符的子字符串(如果参数 *Length* 省略，就默认为"所有字符")。对于 *StartingPos*，指定 1 则从首个字符开始，2 则从第二个字符开始，以此类推(如果 *StartingPos* 超出了 *String* 的长度，将返回一个空字符串)。如果 *StartingPos* 小于 1，将被视为从字符串末尾开始的位移。例如，0 提取最后一个字符，-1 提取最后两个字符(但是如果 *StartingPos* 超过了字符串的左侧末尾，提取又会从左侧首个字符开始)。*Length* 是要获取的字符的最大数目(每当字符串剩余部分太短的时候，获取的长度会比最大数目少一些)。指定一个负的 *Length* 从而在返回的字符串的末尾省略这么多个字符(如果省略了全部或更多字符，将返回一个空字符串)。相关链接：[RegExMatch\(\)](#)(See 18.12), [StringMid](#)(See 27.19), [StringLeft/Right](#)(See 27.15), [StringTrimLeft/Right](#)(See 27.22)。

StrLen(String): 返回 *String* 的长度。如果 *String* 是 [ClipboardAll](#)(See 30.6) 先前分配的变量，将返回它的总大小。相关命令：[StringLen](#)(See 27.17)。

WinActive([WinTitle, WinText, ExcludeTitle, ExcludeText]): 如果激活的窗口匹配指定的条件，则返回窗口的唯一 ID (HWND)(See 28.15)。如果不匹配，函数返回 0。因为所有非零的值都视为 "true"，所以每当匹配了 *WinTitle* 的窗口被激活时语句 *if WinActive("WinTitle")* 都是 true。最后，*WinTitle* 支持 ahk_id, ahk_class 以及其他特殊字符串。关于这些和窗口激活方面的其他信息，详见 [IfWinActive](#)(See 28.6)。

WinExist([WinTitle, WinText, ExcludeTitle, ExcludeText]): 返回以十六进制整数表示的首个匹配窗口的唯一 ID (HWND)(See 28.15)(如果没有就是 0)。因为所有非零的值都视为 "true"，所以每当匹配了 *WinTitle* 的窗口存在时语句 *if WinExist("WinTitle")* 都是 true。最后，*WinTitle* 支持 ahk_id, ahk_class 以及其他特殊字符串。关于这些和窗口查找方面的其他信息，详见 [IfWinExist](#)(See 28.7)。

杂项函数

Asc(String): 返回 *String* 里首个字符的 ASCII 码(一个 1 到 255 之间的数字)。如果 *String* 为空，则返回 0。

Chr(Number): 根据 *Number* 返回 ASCII 码里相应的单个字符。如果 *Number* 不是 1 和 255 及其之间的数，将返回一个空字符串。常用的 ASCII 代码包括 9 (tab), 10 (换行), 13 (回车), 32 (空格), 48-57 (数字 0-9), 65-90 (大写字母 A-Z) 和 97-122 (小写字母 a-z)。

DllCall(): 请看 [DllCall\(\)](#)(See 18.3)。

IsLabel(LabelName): 如果 *LabelName* 是脚本中子程序(See 17.8), 热键(See 4.)或热字符串(See 5.)的标签名称(在 *LabelName* 中不要包括尾部的冒号)，则返回一个非零值。例如：如果标签存在，语句 `if IsLabel(VarContainingLabelName)` 将为 true，反之就是 false。当你在 [Gosub](#)(See 17.8), [Hotkey](#)(See 20.1.10), [Menu](#)(See 22.13) 和 [Gui](#)(See 19.3) 这样的命令中指定了一个动态标签时，这个函数对避免运行错误将十分有用。

ListView 和 TreeView 函数: 详见 [ListView](#)(See 19.7) 和 [TreeView](#)(See 19.8) 页面。

NumGet(VarOrAddress [, Offset = 0, Type = "UInt"]) [v1.0.47+]: 返回储存在指定地址 + 偏移量中的二进制数。对于 *VarOrAddress*, 传递 MyVar 相当于传递 &MyVar。不过，省略 "&" 可以执行地更好并确保目标地址是有效的(See 18.17)（无效的地址将返回 ""）。相比之下，除了变量，传递给 *VarOrAddress* 的其他任何东西都被当作原始地址；因此，指定 MyVar+0 会强制用 MyVar 中的数字来代替 MyVar 本身的地址。对于 *Type*, 可以指定为 UInt, Int, Int64, Short, UShort, Char, UChar, Double 或者 Float(不过与 DllCall 不同，当这些作为原义字符串使用时必须被括在引号里)；详见 [DllCall Types](#)(See 18.3)。

NumPut(Number, VarOrAddress [, Offset = 0, Type = "UInt"]) [v1.0.47+]: 在指定地址 + 偏移量中以二进制的格式储存 *Number*，并在刚刚写入的项目的右边返回地址。对于 *VarOrAddress*, 传递 MyVar 相当于传递 &MyVar。不过，省略 "&" 可以执行地更好并确保目标地址是有效的(See 18.17)（无效的地址将返回 ""）。相比之下，除了变量，传递给 *VarOrAddress* 的其他任何东西都被当作原始地址；因此，指定 MyVar+0 会强制用 MyVar 中的数字来代替 MyVar 本身的地址。对于 *Type*, 可以指定为 UInt, Int, Int64, Short, UShort, Char, UChar, Double 或者 Float(不过与 DllCall 不同，当这些作为原义字符串使用时必须被括在引号里)；详见 [DllCall Types](#)(See 18.3)。如果一个整数太大而无法适应指定的 *Type*, 它最重要的字节就被忽略了；比如 `NumPut(257, var, 0, "Char")` 将存储数字 1。

OnMessage(MsgNumber [, "FunctionName"]): 监控消息/事件。详见 [OnMessage\(\)](#)(See 18.11)。

RegisterCallback(): 请看 [RegisterCallback\(\)](#)(See 18.14)。

VarSetCapacity(UnquotedVarName [, RequestedCapacity, FillByte]): 扩大一个变量的容积或者释放它的内存。详见 [VarSetCapacity\(\)](#)(See 18.17)。

常用数学函数

注意：如果传入的任何参数是非数值的，那么数学函数通常会返回一个空值(空字符串)。

Abs(Number): 返回 *Number* 的绝对值。返回值的类型和 *Number* 一致(整数或浮点数)。

Ceil(Number): 返回 *Number* 向上取整数(没有任何的 .00 后缀)的值。例如, Ceil(1.2) 是 2, Ceil(-1.2) 是 -1。

Exp(N): 返回 e (大约为 2.71828182845905) 的 *N* 次幂。*N* 可以是负数也可以包含小数点。要抬升除了 e 以外的数到某个幂, 请用 [** 运算符](#)(See 9.)。

Floor(Number): 返回 *Number* 向下取整数(没有任何的 .00 后缀)的值。比如 Floor(1.2) 是 1, Floor(-1.2) 是 -2。

Log(Number): 返回 *Number* 的对数(以 10 为底)。结果被格式化为[浮点数](#)(See 21.10)。如果 *Number* 是负数, 将返回一个空字符串。

Ln(Number): 返回 *Number* 的自然对数(以 e 为底)。结果被格式化为[浮点数](#)(See 21.10)。如果 *Number* 是负数, 将返回一个空字符串。

Mod(Dividend, Divisor): 求模。返回被除数 *Dividend* 除以除数 *Divisor* 时余下的值。结果的正负号和被除数一致。例如: mod(5, 3) 和 mod(5, -3) 都得出 2, 但是 mod(-5, 3) 和 mod(-5, -3) 得出 -2。如果输入的任何一个浮点数, 那么结果也会是浮点数。例如, mod(5.0, 3) 结果为 2.0, mod(5, 3.5) 结果为 1.5。如果除数为零, 则函数得到空值(空字符串)。

Round(Number [, N]): 如果 *N* 为 0 或者省略 *N*, *Number* 将四舍五入为整数。如果 *N* 是正数, *Number* 取 *N* 个小数位。如果 *N* 是负数, 在 *Number* 的小数点的左边取 *N* 位来四舍五入。例如: Round(345, -1) 结果为 350, Round(345, -2) 结果为 300。与 [Transform Round](#)(See 21.11) 不同, 每当 *N* 小于 1 或者省略时, 结果没有 .000 后缀。在 v1.0.44.01 及之后的版本中, 一个大于零的 *N* 值会确切地显示 *N* 个小数位, 而不会服从 [SetFormat](#)(See 21.10)。要避免这种情况的话, 对 Round() 的返回值再执行另一个数学操作; 例如: Round(3.333, 1)+0。

Sqrt(Number): 返回 *Number* 的平方根。结果被格式化为[浮点数](#)(See 21.10)。如果 *Number* 是负数, 函数将得出空值(空字符串)。

三角函数

Sin(Number) | Cos(Number) | Tan(Number): 返回 *Number* 的正弦|余弦|正切三角函数值。*Number* 必须用弧度表示(详见下面)。

ASin(Number): 返回以弧度表示的反正弦(其正弦是 *Number*)。如果 *Number* 小于 -1 或者大于 1, 则函数得到一个空值(空字符串)。

ACos(Number): 返回以弧度表示的反余弦(其余弦是 *Number*)。如果 *Number* 小于 -1 或者大于 1, 则函数得到一个空值(空字符串)。

ATan(Number): 返回以弧度表示的反正切(其正切是 *Number*)。

注意: 要将弧度转换成角度, 将其乘以 $180/\pi$ (大约为 57.29578)。要将角度转换成弧度, 将其乘以 $\pi/180$ (大约为 0.01745329252)。 π (大约为 3.141592653589793) 的值是 1 的反正切乘以 4。

其他函数

Titan 的命令函数：为每个有 OutputVar 的 AutoHotKey 命令提供一个可调用的函数。可以通过 [#Include\(See 17.1\)](#) 将这个库包含在任何一个脚本里。

翻译：hsudataalks 修正：天堂之门 menk33@163.com 2008 年 11 月 10 日

18.1 Asc

Asc(String)：返回 *String* 里首个字符的 ASCII 码(一个 1 到 255 之间的数字)。如果 *String* 为空，则返回 0。

18.2 Chr

Chr(Number)：根据 *Number* 返回 ASCII 码里相应的单个字符。如果 *Number* 不是 1 和 255 及其之间的数，将返回一个空字符串。常用的 ASCII 代码包括 9 (tab), 10 (换行), 13 (回车), 32 (空格), 48-57 (数字 0-9), 65-90 (大写字母 A-Z) 和 97-122 (小写字母 a-z)。

DllCall()：请看 [DllCall\(\)](#)。

18.3 DllCall

该命令可以调用一个 DLL 文件中的函数，例如标准的 Windows API 函数。

Result := DllCall("[DllImport]Function" [, Type1, Arg1, Type2, Arg2, "Cdecl ReturnType"])

参数

Result	DllCall 返回调用函数的返回值。如果该函数没有返回值，则结果是未定义的整数类型。如果函数在调用时发生错误 error(See 18.3) ，则返回值为空（即一个空的字符串类型）。
[DllImport]Function	<p>要调用的函数可表示为，DLL 或者 EXE 文件的名字加上反斜杠 \ 再加上函数名。例如：“MyDLL\MyFunction”（如果未指明文件的后缀，则默认为其“.dll”文件）。如果不是一个绝对路径，则假定 <i>DllFile</i>（DLL 文件）在系统目录或者在 A_WorkingDir(See 9.)（当前工作目录）。</p> <p>如果该函数在 User32.dll, Kernel32.dll, ComCtl32.dll, Gdi32.dll 这几个 DLL 文件中，则 <i>DllFile</i> 文件名可以被忽略。例如 "User32\IsWindowVisible" 等同于 "IsWindowVisible"。对于那些标准的 DLL 文件，其 API 函数的后缀“A”字母也可以被忽略。如"MessageBox" 等同于"MessageBoxA"。</p> <p>对于重复调用的情况，使用 Loading it beforehand(See 18.3)（预先装载 DLL 文件），可对执行效率有显著的改善。</p> <p>在 v1.0.46.08+ 版本中，这个参数也可以是指向需要调用函数的内存地址的整数。如 COM(See 18.3) 和 RegisterCallback()(See 18.14) 所提供的地址。</p>
Type1, Arg1	这样的每一对数据，表示需要传递给函数的一个参数。参数的个数没有限制。Type 是

	用来描述传递参数的类型，具体见下表 types table (See 18.3)（参数的类型的描述）说明， Arg 是给函数传递的参数的具体数值。
Cdecl ReturnType	<p>Cdecl: 该参数一般被忽略，因为大多数函数使用的是标准调用习惯而很少是符合“C”标准的调用习惯。如果在忽略这个参数后，产生了错误信息 ErrorLevel An(See 18.3)，n 表示你传递参数的个数，你可能需要填上 Cdecl 参数，声明为以 C 语言方式调用。</p> <p>如果需要在返回的类型前面增加参数 Cdecl 时，请用空格或 tab 将他们分开，如：“Cdecl Str”</p> <p>ReturnType（返回值的类型）: 如果该函数值类型是一个 32-bit 的有符号整型(Int)，BOOL，或是无返回值，ReturnType 可以被忽略。否则必须从下表中 types table(See 18.3) 指定一个参数类型。也支持传址传递 asterisk suffix(See 18.3)。</p>

参数及返回值的类型描述

Str	<p>表示该参数是一个如"Blue" 或 MyVar 的字符串类型。如果函数需要修改这个字符串，或这参数是个不被保护的变量，他其中的内容会被更新。例如，下面的代码将变量 MyVar 里的内容改为大写：DIIcall("CharUpper", "str", MyVar)。</p> <p>然而，如果函数需要储存一个超过当前变量容量的大型字符串，确保在调用函数前该变量的容量足够大。这个可以通过调用函数 VarSetCapacity(MyVar, 123)(See 18.17) 来实现，其中 123 是为变量 MyVar 保留的容量。</p> <p>一个 str 的参数不能是一个赋值的表达式（如 i+1），如果那样，函数将不能成功调用，并且，错误信息会设置为-2。</p> <p>str 类型一般用于描述参数为 LPSTR, LPCSTR, LPTSTR, char * 或类似的类型。</p> <p>同样支持星号后缀 "str *" asterisk variable(See 18.3)（传址引用类型），但很少使用。他被用于参数类似于"char **" 或 "LPSTR *"的类型。</p>
Int64	表示该参数是一个 64-bit 的整型，其范围是 -9223372036854775808 (-0x8000000000000000) 到 9223372036854775807 (0x7FFFFFFFFFFFFF)
Int	<p>表示该参数是一个 32-bit 的整数类型（参数大多数情况为整数类型），其范围是 -2147483648 (-0x80000000) 到 2147483647 (0x7FFFFFFF)。Int 有时也写作"Long"。</p> <p>Int 也用于描述 BOOL 类型的参数（BOOL 类型的参数值只能是 1 或 0）。</p> <p>无符号整型 unsigned(See 18.3) (UInt)也是使用频繁的类型，可以表示如 DWORD 和 COLORREF 类型。他也被用于所有的句柄类型，如 HWND, HBRUSH, 和 HBITMAP 类型。</p> <p>注意：如要传递的值是 NULL 或是指针，可传递整数 0。</p>
Short	表示参数为 16-bit 的整型，其范围是 -32768 (-0x8000) 到 32767 (0x7FFF)。一个无符号

	短整型 unsigned (See 18.3) Short (UShort) 可用于函数的参数为 WORD 类型。
Char	表示参数为 8-bit 的整型，其范围是 -128 (-0x80) 到 127 (0x7F)。一个无符号字符 unsigned (See 18.3) character (UChar) 可用于函数的参数为 BYTE 类型。
Float	表示参数为 32-bit 的浮点型，具有 6 位精确度。
Double	表示参数为 32-bit 的浮点型，具有 15 位精确度。
* or P (suffix)	<p>在上述类型后，附加一个星号（可以在之前有空格）表示函数的参数需要传递的是变量的地址而不是变量的数值（当然函数原型要接受该类型的参数）。该参数的值有可能会被函数修改，一个未被保护的变量作为一个参数传入，其内容也会随时被更新。例如，下面的例子向函数 MyFunction 传递了变量 MyVar 的地址，其变量 MyVar 可以被 MyFunction 函数随时更新以反映其变化：DllCall("MyDll\MyFunction", "int *", MyVar)</p> <p>一般来说，星号可用于描述任何参数类型或返回类型是以 "LP" 开头的（字符串类型 LPSTR 除外，他应使用 "str"(See 18.3) 类型）。例如常见的 LPDWORD，他就是指向 DWORD 类型变量的指针类型。DWORD 是一个 32-bit 的整型，所以用"UInt *" 或 "UIntP"来表示 LPDWORD 类型。</p> <p>注意："char *"与 "str"(See 18.3) 的类型是不一样的，因为 "char *" 传递的是一个 8-bit 数字的地址，但 "str"(See 18.3) 传递的是一个序列字符的地址。并且对于函数需要向变量储存数据的传址引用类型的参数，不需要调用 VarSetCapacity(See 18.17) 来设置其容量。</p>
U (prefix)	<p>在上述的整数类型前加上前缀 "U" 表示无符号整型（如 UInt64, UInt, UShort 和 UChar）。严格地说，这只对返回值类型和传址引用类型有必要，因为有符号或无符号对参数的传递没有影响（Int64 除外）。</p> <p>一个 32-bit 的无符号整型 (UInt) 可以替代如 DWORD, HWND, 或类似的类型。一个 HWND 值（窗口句柄）和窗口的 unique ID(See 28.15) 也是一样的。</p> <p>如果把一个负数类型当成一个无符号整数，则这个负整数会被转换到无符号的范围内。例如，把 -1 作为一个无符号类型 UInt，他的值会变成 0xFFFFFFFF。然而，这种情况对于 64-bit 整数 (UInt64) 不成立。</p> <p>对由一个函数产生的无符号 64-bit 整型不完全支持。对于大于或等于 0x8000000000000000 的数，若省略前缀 U 后，则会将函数得到的任何负数，转换成一个很大的正数。例如一个被设计其返回值为 UInt64 (64 位整型) 类型的函数，返回结果为 Int64 类型的 -1，则脚本实际返回的值将是 0xFFFFFFFFFFFFFF.</p>

注意：当描述一个不包含空格或星号的参数类型或是返回值类型时，其引号可以忽略，如，str 与 "str" 等同，CDecl 与 "CDecl" 等同。另外，字母 P 可以替代星号，以便允许忽略其周围的引号。如：UIntP。

ErrorLevel 错误信息

[ErrorLevel](#)(See 30.8) 会赋值为以下几种情况中的一个，不论调用成功或是失败。

0: 成功。

- 1 (负 1): 函数 `[DllFile]\Function` 的参数是一个浮点类型。需要一个字符串或是一个正数的参数。
- 2: `return type` (返回值类型) 或一个描述参数类型的 `arg types`(See 18.3) 数据无效。这个错误可能是因为向字符串 (`str`(See 18.3)) 类型变量传递了一个赋值表达式所引起的。
- 3: 指定的 DLL 文件不能够使用。若没有指定 DLL 文件的路径，则文件不存在于系统目录或是 `A_WorkingDir`(See 9.) (当前工作目录)。这个错误也可能是因为用户没有读取文件的权限所引起的。
- 4: 无法在 DLL 文件中找到指定的函数。

N (任意正数): 表示函数被调用，但发生了异常错误，错误序号为 **N** (例如，`0xC0000005` 表示“读取错误”)。这种情况下函数会返回一个空值 (空字符串)，但传址引用类型的变量会被更新。例如可能是因为引用了一个无效的 `NULL` 指针而导致的异常错误。由于 `Cdecl`(See 18.3) 函数不可能产生 "*An*" 错误 (下一段提到)，也可能是因为传入的参数过少而产生了异常。

An (字母 A 后面加一个整数 **n**): 表示函数被调用，但传入的参数过少。`"n"` 表示错误参数列表的字节数。如果 **n** 为正数，表示传入的参数类型太多 (或是参数类型太大)，也有可能是函数调用需要 `CDecl`(See 18.3) 声明。如果 **n** 是负数，表示传入的参数类型太少。这种情况可以通过改正参数类型来保证函数的正常运行。若函数返回值为空，也可能是因为产生了异常而导致的错误。

异常和 `A_LastError`

尽管具有内建的异常处理机制，但仍有可能因为使用 `DllCall` 而使脚本假死。这种情况会产生于，当函数不是直接产生了异常而是产生了一些不恰当的数据，如产生了一个野指针或是一个没有结束符的字符串。这可能不是因为 函数本身产生的错误，而是脚本传入了一个不恰当的参数，例如一个野指针或是一个容量不足的字符串 `"str`(See 18.3)"。当对参数类型或返回类型的型描述不恰当的时，也可能使脚本假死。例如把一个普通的整数类型错误的声明为一个传址引用类型 `asterisk variable`(See 18.3) 或是 `str`(See 18.3)。

内建的变量 `A_LastError` 包含了系统函数 `GetLastError()` 返回的结果。该函数会在函数调用后被立即执行 (对效率的影响不可知)。`A_LastError` 是一个介于 0 与 4294967295 之间的数 (通常以十进制表示而不是十六进制)。像 `ErrorLevel`(See 30.8) 一样，`A_LastError` 也是一个独立的线程。也就是说其他的线程 `threads`(See 30.19) 无法中断他。然而 `A_LastError` 可受到 `Run/RunWait`(See 24.5) 的影响。

性能

当需要重复调用 DLL 时，事先明确的装载可以显著提高运行效率 (标准的 `standard DLL`(See 18.3) 如 `User32` 文件可以不需要，因为他们总是挂起的)。以下是一个在每次调用 `DllCall` 时不需要重复进行 `LoadLibrary` 和 `FreeLibrary` 的例子：

```
hModule := DllCall("LoadLibrary", "str", "MyFunctions.dll"); 避免了每次在循环中调用
DllCall() 时都进行装载文件。
Loop, C:\My Documents\*.*,,1
result := DllCall("MyFunctions\BackupFile", "str", A_LoopFileFullPath)
DllCall("FreeLibrary", "UInt", hModule); 为了释放内存，DLL 在使用完后进行卸载。
```

在 1.0.46.08+ 版本中，通过预先查找函数的地址甚至可以获得更快的性能。例如：

; 在下面的例子中，如果 DLL 还没有加载，使用 LoadLibrary 替换 GetModuleHandle 。

```
MulDivProc := DllCall("GetProcAddress", uint, DllCall("GetModuleHandle", str,
"kernel32"), str, "MulDiv")
```

```
Loop 500
```

```
DllCall(MulDivProc, int, 3, int, 4, int, 3)
```

最后，向一个函数传入一个字符串变量，不会改变字符串的长度，如果将变量的地址 [by address\(See 9.\)](#) (如，`&MyVar`) 传递给函数而不是以字符串 "`str(See 18.3)`" (特别是一些长字符串) 的形式，可以提高运行效率。以下是一个将字符串改写为大写的例子：`DllCall("CharUpper", uint, &MyVar)`

结构体和数组

一个结构体是一些成员变量（空间）在内存中连续排列的集合。大多数成员变量是整型。

一些函数可以接受结构体的地址（或是一个内存块数组）的类型参数，可以调用他以二进制的形式向结构体写入数据。一般步骤如下：

- 1) 调用 [VarSetCapacity\(MyStruct, 123, 0\)\(See 18.17\)](#) 确保目标变量足够大以便储存结构体数据。将 123 替换为结构体的大小。后面的 0 是可选项；他会初始化所有的成员变量为二进制形式的 0，这样通常可以有效的避免在下一步频繁的调用 `NumPut()`。
- 2) 如果目标函数需要其结构体变量初始化，调用 [NumPut\(123, MyStruct, 4\)\(See 10.\)](#) 初始化其中的每个成员变量为非 0 数据。替换 123 为需要向成员变量存入的整数（或使用 `&Var` 形式存入一个变量的地址 [address\(See 9.\)](#)）。替换 4 为成员变量的偏移量（“偏移量”请详见第四步）。
- 3) 调用目标函数，以 UInt 类型的形式传递结构体 MyStruct 的地址 [address\(See 9.\)](#)。例如 `DllCall("MyDII\MyFunc", UInt, &MyStruct)`。函数调用后会检查、改变一些传入的结构体成员变量。
- 4) 利用 [MyInteger := NumGet\(MyStruct, 4\)\(See 10.\)](#) 命令从结构体中获取想要的成员变量。替换 4 为目标成员变量在结构体中的偏移量。第一个成员变量的偏移量总是 0。第二个成员变量的偏移量为第一个成员变量的偏移量 0 加上第一个成员变量的大小（一般为 4 字节）。后面的成员变量的偏移量等于前一个成员变量的偏移量加上前一个成员变量的大小。大多数成员变量 -- 如 `DWORD`, `Int` 和 [other types of 32-bit integers\(See 18.3\)](#)（其他 32 位整型）其大小都是 4 字节。

参见 [Structure Examples\(See 18.3\)](#) 结构体详细用法的例子。

已知的局限性

当一个 [variable's address\(See 9.\)](#) 函数的地址（如 `&MyVar`）传递给一个函数时，这个函数改变了这个变量内容的长度，如果在此时使用了这个变量，可能就会产生错误。想要解决这个问题，有以下两种方法：

- 1) 使用 `"str"(See 18.3)` 类型而不是 `uint/address` 类型；2) 在 v1.0.44.03+ 版本，在调用 `DllCall` 命令前，使用 [VarSetCapacity\(MyVar, -1\)\(See 18.17\)](#) 命令更新变量的内部容量。

任何存入二进制数据 0 的变量都会被函数隐藏掉其右边的 0；这种情况下数据不能被多数的命令或函数改变或使用。但可以被当作地址或解引用(`& and *`(See 9.))及 `DllCall` 自身操作。

当一个字符串传入函数后，该函数可能会返回一个地址不同但内容相同的字符串。例如在某些程序中调用 `CharLower(CharUpper(MyVar))` 可以将 `MyVar` 里的内容转换为小写，但当在脚本中调用 `DllCall()` 完成相同的功能后，`MyVar` 里的内容可能仍是大写的，这是因为 `CharLower` 函数是对一个与 `MyVar` 内容相同的临时字符串变量进行操作（而不是直接对变量 `MyVar` 操作）。

```
MyVar = ABC
result := DllCall("CharLower", str, DllCall("CharUpper", str, MyVar, str), str)
```

如果想让以上的代码直接对变量 `MyVar` 中的内容进行修改，可以将以上两个带有下划线的 "str" 改为 `UInt`。这种改变会将 `CharUpper` 函数的返回值转换为地址，然后再以整数类型传递给函数 `CharLower`。

VBScript, JScript 和 组件对象模型 (COM)

`VBScript` 和 `JScript` 可以通过 [Windows Scripting for AutoHotkey](#) (AHK 中的 `Window` 脚本) 嵌入到 AHK 脚本中，他们是通过访问 COM 组件来完成的。

同样，可直接调用 `DllCall` 命令来使用 COM 对象，参考范例：

www.autohotkey.com/wiki/index.php?title=

相关命令

[PostMessage](#)(See 28.1.12), [OnMessage\(\)](#)(See 18.11), [RegisterCallback\(\)](#)(See 18.14), [Run](#)(See 24.5), [VarSetCapacity](#)(See 18.17), [Functions](#)(See 10.), [SysGet](#)(See 22.21), [MSDN Library](#)

示例

```
; 例子：调用 Windows API 函数 "MessageBox" 并报告用户按下了哪个按钮。
WhichButton := DllCall("MessageBox", "int", "0", "str", "Press Yes or No", "str", "Title of
box", "int", 4)
MsgBox You pressed button #%WhichButton%.
```

```
; 例子：改变桌面为指定的 Bitmap 文件 (.bmp) 文件。
DllCall("SystemParametersInfo", UInt, 0x14, UInt, 0, Str, A_WinDir . "\winnt.bmp", UInt,
2)
```

```
; 例子：调用 API 函数 "IsWindowVisible" 来判断记事本窗口是否可见。
DetectHiddenWindows On
if not DllCall("IsWindowVisible", "UInt", WinExist("无标题 - 记事本"))
; WinExist()函数返回窗口的句柄。
 MsgBox 窗口不可见
```

; 例子：调用 API 的 wsprintf() 函数，在一个长度为 10 字节的字符串中填上 432，其他空位用 0 填充，也就是(0000000432)。

```
VarSetCapacity(ZeroPaddedNumber, 20) ;确保变量的容量足够大以便能够容纳新的字符串。
DllCall("wsprintf", "str", ZeroPaddedNumber, "str", "%010d", "int", 432, "Cdecl") ;需要声明 Cdecl 的调用方法。
MsgBox %ZeroPaddedNumber%
```

; 例子：调用 QueryPerformanceCounter() 函数的一个示范，他提供了比 A_TickCount's(See 9.) 的 10ms 更精确的计时器。

```
DllCall("QueryPerformanceCounter", "Int64 *", CounterBefore)
Sleep 1000
DllCall("QueryPerformanceCounter", "Int64 *", CounterAfter)
MsgBox % "Elapsed QPC time is " . CounterAfter - CounterBefore
```

; 例子：这是一个用快捷键来减慢鼠标的移动速度，以便能够精确的定位。

; 按下 F1 键降低移动速度，释放回复原来的速度。

```
F1:::
SPI_GETMOUSESPEED = 0x70
SPI_SETMOUSESPEED = 0x71
; 得到当前鼠标的速度以便稍后恢复：
DllCall("SystemParametersInfo", UInt, SPI_GETMOUSESPEED, UInt, 0, UIntP,
OrigMouseSpeed, UInt, 0)
; 现在降低鼠标的移动速度，第三个参数为速度设定（范围是 1-20，10 为默认值）：
DllCall("SystemParametersInfo", UInt, SPI_SETMOUSESPEED, UInt, 0, UInt, 3, UInt, 0)
KeyWait F1 ; 这句话可防止按键重复调用 DllCall。
return
F1 up::DllCall("SystemParametersInfo", UInt, 0x71, UInt, 0, UInt, OrigMouseSpeed, UInt,
0); 恢复原来的鼠标速度
```

; 例子：当传入一个窗口的 ID 和其控件的类名或控件上的文字

; 下面的函数就可以返回该控件的 HWND (句柄)。

; 在 v1.0.43.06+ 版本中：下面的命令能够更准确的完成该例子的功能。（该例子省略）

[ControlGet, OutputVar, Hwnd,, ClassNN, WinTitle\(See 28.1.4\)](#)

; 例子：监视活动窗口并显示其垂直滚动条的位置。

; 因为焦点控件随时更新，所以需要用到 v1.0.43.06+ 本版中的 [ControlGet Hwnd\(See 28.1.4\)](#) 命令。

#Persistent

```
SetTimer, WatchScrollBar, 100
```

```

return

WatchScrollBar:
ActiveWindow := WinExist("A")
if not ActiveWindow ; No active window.
return
ControlGetFocus, FocusedControl, ahk_id %ActiveWindow%
if not FocusedControl ; No focused control.
return
; 在 ToolTip 中显示垂直或水平滚动条的位置:
ControlGet, ChildHWND, Hwnd,, %FocusedControl%, ahk_id %ActiveWindow%
ToolTip % DllCall("GetScrollPos", "UInt", ChildHWND, "Int", 1)
; 最后一个参数若为 1 表示 SB_VERT 垂直滚动条,为 0 表示 SB_HORZ 水平滚动条。
return

```

; 例子: 这个脚本向文件中写入一些文字, 然后再把它们读入到内存中。(需要 v1.0.34+ 版本).
; 这个方法可以实现对一个文件同时并发读写的操作。

```

FileSelectFile, FileName, S16,, Create a new file:
if FileName =
return
GENERIC_WRITE = 0x40000000 ; Open the file for writing rather than reading.
CREATE_ALWAYS = 2 ; 创建新文件 (如果该文件存在则覆盖).
hFile := DllCall("CreateFile", str, FileName, UInt, GENERIC_WRITE, UInt, 0, UInt, 0, UInt,
CREATE_ALWAYS, UInt, 0, UInt, 0)
if not hFile
{
    MsgBox 不能对文件 "%FileName%" 进行写操作
    return
}
TestString = This is a test string.`r`n ; 向新文件中写入文字, 使用 `r`n (不要用 `n ) 来表示
新的一行
DllCall("WriteFile", UInt, hFile, str, TestString, UInt, StrLen(TestString), UIntP,
BytesActuallyWritten, UInt, 0)
DllCall("CloseHandle", UInt, hFile) ; Close the file.
; 现在文件已经成功的写入, 再把它的内容读到内存中来
GENERIC_READ = 0x80000000 ; 以读取方式而不是写入方式来打开文件
OPEN_EXISTING = 3 ; 这个参数表示要打开的文件必须存在
FILE_SHARE_READ = 0x1 ; 这一行以及下一行表示, 允许其他进程在我们打开文件的同时也能够打
开该文件
FILE_SHARE_WRITE = 0x2
hFile := DllCall("CreateFile", str, FileName, UInt, GENERIC_READ, UInt,
FILE_SHARE_READ|FILE_SHARE_WRITE, UInt, 0, UInt, OPEN_EXISTING, UInt, 0, UInt, 0)
if not hFile

```

```
{
    MsgBox 不能够读取文件 "%FileName%"
    return
}
; 清空变量以便能够检查是否正确读入，但要确保该变量的容量可以容纳将要读入的文字
BytesToRead := VarSetCapacity(TestString, StrLen(TestString))
DIICall("ReadFile", UInt, hFile, str, TestString, UInt, BytesToRead, UIntP,
BytesActuallyRead, UInt, 0)
DIICall("CloseHandle", UInt, hFile) ; Close the file.
MsgBox 以下是读取文本的内容: %TestString%
```

```
; 例子：当按下 Win+C 时隐藏鼠标指针，再次压下时显示。
; 这个脚本来自 www.autohotkey.com/forum/topic6107.html
OnExit, ShowCursor ; 确保当脚本退出时，鼠标指针可见。
return
ShowCursor:
SystemCursor("On")
ExitApp
#c::SystemCursor("Toggle") ; Win+C 快捷键切换显示和隐藏
SystemCursor(OnOff=1) ; 初始化 = "I"或"Init"; 隐藏 = 0 或 "Off"; 相反 = -1 或 "T" 或
"Toggle"; 显示 = 其他
{
static AndMask, XorMask, $, h_cursor
,c0,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,c13 ; 系统指针
, b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,b12,b13 ; 空白指针
, h1,h2,h3,h4,h5,h6,h7,h8,h9,h10,h11,h12,h13 ; 默认指针的句柄
if (OnOff = "Init" or OnOff = "I" or $ = "") ; 请求或第一次调用时初始化
{
$ = h ; active default cursors
VarSetCapacity( h_cursor,4444, 1 )
VarSetCapacity( AndMask, 32*4, 0xFF )
VarSetCapacity( XorMask, 32*4, 0 )
system_cursors =
32512,32513,32514,32515,32516,32642,32643,32644,32645,32646,32648,32649,326
50
StringSplit c, system_cursors, `,
Loop %c0%
{
h_cursor := DIICall( "LoadCursor", "uint",0, "uint",c%A_Index% )
h%A_Index% := DIICall( "CopyImage", "uint",h_cursor, "uint",2, "int",0, "int",0, "uint",0 )
b%A_Index% := DIICall("CreateCursor","uint",0, "int",0, "int",0
, "int",32, "int",32, "uint",&AndMask, "uint",&XorMask )
}
```

```

}

if (OnOff = 0 or OnOff = "Off" or $ = "h" and (OnOff < 0 or OnOff = "Toggle" or OnOff = "T"))
$ = b ; 使用空白指针
else
$ = h ; 使用储存的指针
Loop %c0%
{
h_cursor := DllCall( "CopyImage", "uint",%$%%A_Index%, "uint",2, "int",0, "int",0,
"uint",0 )
DllCall( "SetSystemCursor", "uint",h_cursor, "uint",c%A_Index% )
}
}

```

; 结构体例子：把结构体 RECT 的地址传递给 GetWindowRect() 函数。
; 完成把窗口的左，上，右，下（相对与屏幕的位置）储存到该结构体的成员中。

Run Notepad
WinWait 无标题 - 记事本 ; 也可以设置为 "last found window(See 30.2)" 使用 WinExist() 得到记事本句柄
VarSetCapacity(Rect, 16) ; 一个 RECT 结构体包含了 4 个 32-bit 的整型，(其容量为 $4 \times 4 =$
DllCall("GetWindowRect", UInt, WinExist(), UInt, &Rect) ; WinExist() 函数返回一个句柄
MsgBox % "Left " . **NumGet**(See 10.)(Rect, 0, true) . " Top " . NumGet(Rect, 4, true)
. " Right " . NumGet(Rect, 8, true) . " Bottom " . NumGet(Rect, 12, true)

; 结构体例子：向函数 FillRect() 传递一个 RECT 结构体，在屏幕上临时画出红色矩形
VarSetCapacity(Rect, 16, 0) ; 设置容量为 4 个 4 字节的整型即 16，并初始化为 0。
NumPut(See 10.)(A_ScreenWidth//2, Rect, 8) ; 第三个整数在结构体中为 "rect.right"
NumPut(A_ScreenHeight//2, Rect, 12) ; 第四个整数在结构体中为 s "rect.bottom".
hDC := **DllCall**("GetDC", UInt, 0) ; 传入 0 获得桌面的设备上下文
hBrush := **DllCall**("CreateSolidBrush", UInt, 0x0000FF) ; 创建一个红色画刷 (0x0000FF 是
RGB 格式的红色)
DllCall("FillRect", UInt, hDC, Str, Rect, UInt, hBrush) ; 用上面创建的画刷填充指定的矩形
DllCall("ReleaseDC", UInt, 0, UInt, hDC) ; 释放 DC
DllCall("DeleteObject", UInt, hBrush) ; 释放画刷

; 结构体例子：改变系统时间为指定的日期，请注意如果时间改为未来的日期，可能使一些软件过早的
运行计划的任务。
SetSystemTime("20051008142211") ; 传入一个 **timestamp**(See 16.26) 参数 (本地时间
Local 而非全球时间 **UTC**)
SetSystemTime(YYYYMMDDHHMISS)
; 设置系统时间为指定日期时间，调用必须确保传入的参数是一个有效的时间格式数据 (本地时间而非
全球时间)

```

; 返回值为非零表示调用成功, 否则为失败
{
; 把时间参数从 local 转换为 UTC 以便 SetSystemTime() 函数能够调用
UTC_Delta -= %A_NowUTC%, Seconds ; 四舍五入使秒更加精确
UTC_Delta := Round(-UTC_Delta/60) ; 四舍五入使分更加精确
YYYYMMDDHHMISS += %UTC_Delta%, Minutes ; 把时间转换为 UTC 格式
VarSetCapacity(SystemTime, 16, 0) ; 这个结构体是由 8 个 UInts 组成, 所以容量为 8×2=
StringLeft, Int, YYYYMMDDHHMISS, 4 ; YYYY (年)
NumPut(See 10.)(Int, SystemTime, 0, 2)
StringMid, Int, YYYYMMDDHHMISS, 5, 2 ; MM (月份, 1-12)
NumPut(Int, SystemTime, 2, 2)
StringMid, Int, YYYYMMDDHHMISS, 7, 2 ; DD (日)
NumPut(Int, SystemTime, 6, 2)
StringMid, Int, YYYYMMDDHHMISS, 9, 2 ; HH (小时 0-23)
NumPut(Int, SystemTime, 8, 2)
StringMid, Int, YYYYMMDDHHMISS, 11, 2 ; MI (分)
NumPut(Int, SystemTime, 10, 2)
StringMid, Int, YYYYMMDDHHMISS, 13, 2 ; SS (秒)
NumPut(Int, SystemTime, 12, 2)
return DllCall("SetSystemTime", UInt, &SystemTime)
}

```

更多结构体的例子:

- 1) WinLIRC client script(See 30.36) 的范例, 介绍了如何使用 DllCall() 来完成 TCP/IP 服务器端和客户端的数据交互和网络链接。
- 2) 在 www.autohotkey.com/forum/topic17230.html 这里介绍了如何利用结构体来使用系统提供的标准对话框, 如字体, 颜色, 图标的选取对话框。

BLooM.2 Fantasy OnLine 08.07.30 翻译

18.4 FileExist

FileExist(FilePattern): 如果 *FilePattern* 不存在(如果未指定绝对路径, 程序将假设 *FilePattern* 在 *A_WorkingDir*)就返回一个空值(空字符串)。否则, 它将返回首个匹配的文件或文件夹的属性字串 ("RASHNDOCT"的子集)。如果文件没有属性(罕见), 将返回 "X"。*FilePattern* 可以是文件或文件夹的准确的名称, 或者包含通配符 (*) 或 (?). 因为空字符串被视为 "false", 所以函数的返回值总是可以作为一个准布尔值来使用。例如: 如果文件存在, 语句 *if FileExist("C:\My File.txt")* 将为 *true*, 反之为 *false*。相似地, 只有当文档存在并且是一个目录, 语句 *if InStr(FileExist("C:\My Folder"), "D")* 才为 *true*。相关命令: *IfExist* 和 *FileGetAttrib*。

18.5 GetKeyState

GetKeyState(KeyName [, "P" or "T"]): 与 [GetKeyState](#) 命令不同，后者是按下的返回 D，弹起的返回 U；而这个函数是如果键是按下的就返回 true (1)，弹起的返回 false (0)。如果参数 *KeyName* 是无效的，将返回空字符串。请看 [GetKeyState](#) 来了解其他返回值和用法。

18.6 InStr

InStr(Haystack, Needle [, CaseSensitive = false, StartingPos = 1]): 返回 *Haystack* 字符串中首个匹配的 *Needle* 字符串的位置。与 [StringGetPos](#) 不同，第一个字符的位置是 1；这是因为 0 同义于 "false"，会使其成为一个直观的“未找到”的指示。如果参数 *CaseSensitive* 被省略或者是 false，则搜索将不区分大小写(不区分的模式取决于 [StringCaseSense](#) 命令)；否则就得精确地匹配大小写。如果省略 *StartingPos*，其默认为 1(*Haystack* 字符串的起点)。否则，指定 2 从 *Haystack* 的第二个字符开始搜索，3 从第三个开始，等等。如果 *StartingPos* 超出 *Haystack* 的长度，那么函数将返回 0。如果 *StartingPos* 为 0，那么搜索将逆序执行(从右到左)以便找到在最右边匹配的结果。不管 *StartingPos* 的值是多少，返回的位置将始终相对于 *Haystack* 的首个字符而言。例如：在"123abc789"中"abc"的位置永远是 4。相关链接：[RegExMatch\(\)](#), [IfInString](#) 和 [StringGetPos](#)。（译注：例如 `InStr("123abc789","abc",false,1)`）

RegExMatch(Haystack, NeedleRegEx [, UnquotedOutputVar = "", StartingPos = 1]):
请看 [RegExMatch\(\)](#)。

RegExReplace(Haystack, NeedleRegEx [, Replacement = "", OutputVarCount = "", Limit = -1, StartingPos = 1]): 请看 [RegExReplace\(\)](#)。

18.7 IsFunc

- 介绍以及简单的例子
- 参数
- 可选参数
- 局部变量
- 优化布尔求值
- 在函数中使用子程序
- `Return`, `Exit` 和一般说明
- 用 `#Include` 在多个脚本中共享函数
- 函数库：标准库和用户库
- 内置函数

函数和子程序([Gosub](#))相似，只不过它能从它的调用者那里接受参数(输入)。此外，函数能随意地向它的调用者返回一个值。参考下面这个简单的函数，它接受了两个数字并返回它们的和：

```
Add(x, y)
{
```

```
return x + y ; "Return" 需要一个表达式。
}
```

上面的被称为**函数定义**, 因为它创建了一个名为 "Add"(不区分大小写)的函数, 并且确定无论谁要调用它都必须确切地提供两个参数(x 和 y)。要调用函数并将它的结果赋值给变量, 用 **:=运算符**。例如:

```
Var := Add(2, 3) ;数字 5 将存储在变量 var 中。
```

而且, 可以调用函数而存储它的返回值:

```
Add(2, 3)
```

但是如果这样, 任何通过函数返回的值都将被抛弃; 因此除非函数除了它的返回值外还会产生其他一些效果, 不然此调用将无济于事。

一旦函数在**表达式**中被调用, 任何在它参数列表中的变量名都不能被附上百分号。相反, 原义字符串却要加上双引号。例如:

```
if InStr(MyVar, "fox")
    MsgBox MyVar 变量包含单词 fox.
```

在 v1.0.47.06 及之后版本里, 一个函数(甚至是**内置函数**)可以通过百分号被动态地调用。例如, **%Var%(x, "fox")** 将调用一个名称存储在变量 Var 里的函数。相似地, **Func%A_Index%()** 将调用名称存储在指定**数组元素**中的函数。所调用的函数必须在脚本中明确地定义, 也可通过 **#Include** 或者非动态调用一个**函数库**。如果被调用的函数不存在, 或者给函数参数传递的值或类型不对, 那么包含此调用的表达式将产生一个空字符串。

最后, 函数可以被任何命令的参数调用(除了像 **StringLen** 命令中的那些 **OutputVar** 和 **InputVar** 参数)。不过, 那些不支持**表达式**的参数必须使用 "%"作为前缀, 例如下面这个例子:

```
MsgBox % "答案是: " . Add(3, 2)
```

"%"前缀也允许用在本来就支持表达式的参数里, 但它将被简单地忽略。

当一个函数被定义的时候, 它的参数都列在它的名称后面的括号中(它的名称和左括号之间不能有空格)。如果一个函数不接收任何参数, 就让括号空着; 例如: **GetCurrentTimestamp()**。

ByRef 参数: 从函数的观点来看, 参数本质上和**局部变量**一样, 除非它们像在下面这个例子中那样被定义为 **ByRef**:

```
Swap(ByRef Left, ByRef Right)
{
    temp := Left
    Left := Right
    Right := temp
}
```

在上面的例子中，**ByRef** 的使用导致各个参数变成了从调用者传递进来的变量的一个别名，换句话说，参数和调用者的变量在内存中都引用了相同的内容。这就允许了 **Swap** 函数通过移动 *Left* 的内容给 *Right* 来改变调用者的变量，反过来也一样。

相比之下，如果在上面的例子中没有使用 **ByRef** 的话，*Left* 和 *Right* 将是调用者的变量的复制，因此 **Swap** 函数也将没有效果。

由于 **return** 只能送回一个值给函数的调用者，所以 **ByRef** 可以用来送回更多的结果。这是由函数向调用者传递进来的变量(通常为空)储存一个值来实现的。

当向函数传递大量字符串的时候，**ByRef** 可以提高性能并且通过回避复制字符串的需求来节约内存。同样，使用 **ByRef** 送回一个长字符串给调用者往往比例如 **Return** 巨型字符串执行得更好。

已知限制：

- 不能向函数的 **ByRef** 参数传递剪贴板，内置变量或者环境变量，甚至当 **#NoEnv** 在脚本里不存在的时候。传递一个内置变量给一个 **ByRef** 参数将导致显示一个错误对话框。
- 尽管一个函数能递归地调用自身，但如果它传递它自己的一个局部变量或者非 **ByRef** 参数给它自身的 **ByRef**，新一层的 **ByRef** 参数将引用它自身那个局部变量的名称而不是之前的层的。不过，这个矛盾在一个函数传递一个全局变量，静态变量或者 **ByRef** 参数给它自身的时候不会发生。
- 如果一个参数在函数调用里被解析为一个变量（比如 **Var** 或 **++Var** 或 **Var*=**），它左边或右边的其他参数能在它被传递给函数前改变那个变量。比如，当 **Var** 最初为 **0** 时，即使当函数的首个参数不是 **ByRef** 时，**func(Var, Var++)** 仍会意外地传递 **1** 和 **0**。由于这种行为是违反常规的，所以可能在将来放出的版本中改变。

在定义一个函数时，它的一个或多个参数可以被标记为可选。这可以通过给它们添加一个等号和一个默认值来实现。下面的函数的参数 **Z** 已标记为可选：

```
Add(X, Y, Z = 0)
{
    return X + Y + Z
}
```

当调用者传递满三个参数给上面的函数时，**Z** 的默认值将被忽略。但是当调用者仅传递了两个参数时，**Z** 自动地获取默认值 **0**。

不能将可选参数孤立在参数列表的中间。换句话说，位于首个可选参数右边的所有的参数也必须标记为可选。

在 v1.0.46.13 及之后版本中，**ByRef** 参数也可支持默认值；例如：**Func(ByRef p1 = "")**。每当调用者省略了这样一个参数，函数会创建一个局部变量并赋上默认值；换句话说，函数会表现得像没有 **ByRef** 这个关键词一样。

一个参数的默认值必须是下列形式之一：**true**, **false**, 一个原义的整数，一个原义的浮点数，或一个引用的/原义的字符串例如 "fox" 或 "" (但在 1.0.46.13+ 之前的版本只允许 "")。

局部变量

所有在函数内部引用或创建的变量默认都是**局部的**(除了 [Clipboard](#), [ErrorLevel](#) 和 [A_TimeIdle](#) 这些内置变量)。每个局部变量的内容都只能在函数内可见。所以，一个局部变量可以和一个全局变量有着相同的名字却有着不同的内容。最后，所有的局部变量每次在函数被调用时都以空值开始。

全局变量

要在一个函数里引用一个存在的全局变量(或创建一个新的)，需要在使用它之前先声明此变量为 `global`。例如：

```
LogToFile(TextToLog)
{
    global LogFileName ;这个全局变量先前已经在此函数外赋过值了。
    FileAppend, %TextToLog%`n, %LogFileName%
}
```

如果一个函数需要引用或创建大量的全局变量，可以通过将它的首行设为单词"**global**"或者声明一个局部变量来假设它所有的变量都是全局的(它的参数除外)。例如：

```
SetDefaults()
{
    global ;如果在此函数的首行有比如"local MyVar"这样的词，那么这个单词可以被省略。
    Var := 33 ;将 33 赋值给一个全局变量，如果需要，首次可创建变量。
    local x, y:=0, z ;局部变量必须用这种形式来声明，不然它们会被假定为全局变量。
    ; ...等等。
}
```

这种假设的全局模式也可以被函数用来创建一个全局数组，例如一个赋值给 `Array%A_Index%` 的循环。

静态变量

一个变量可以被声明为 `static` 来使它的值在多次调用期间被记住。例如：

```
LogToFile(TextToLog)
{
    static LineCount = 0
    LineCount += 1 ;保持自身的累加(它的值在多次调用期间能被记住)。
    global LogFileName
    FileAppend, %LineCount%: %TextToLog%`n, %LogFileName%
}
```

静态变量一般都是隐式的局部变量。在 **1.0.46** 之前的版本，所有的静态变量都以空值开始；所以要检查一个静态变量首次被使用的唯一办法就是检查它是否为空值。在 **v1.0.46** 及之后的版本，一个静态变量可以初始化为除了 "" 外的东西，通过其后跟 `:=` 或 `=` 以及下列之一： `true`, `false`, 一个原义的整数，一个原义的浮点数，或一个引用的/原义的字符串比如 "fox"。例如：`static X:=0, Y:="fox"`。每个静态变量都只初始化一次(在脚本执行之前)。

关于局部和全局的更多信息：

在下面例子中，通过逗号将多种变量分隔开从而在同一行声明：

```
global LogFileName, MaxRetries := 5
static TotalAttempts = 0, PrevResult
```

在 v1.0.46 及之后版本，一个局部或者全局变量可以在同一行被初始化，通过在它的声明后加上 `:=` 或 `=` 以及任何[表达式](#)(在声明中运算符 `=` 和 `:=` 作用相同)。在一个特定行有多个声明时，由于[性能原因](#)(不像普通的[逗号分隔语句](#))每个有初始化设定的声明都被作为单独的行执行。与[静态变量初始化设定](#)不同，局部变量和全局变量的初始化设定在每次函数被调用时都会执行，但是仅当控制流实际能到达它们这时。换句话说，在一行中写下 `local x = 0` 和在两个单独的行写下 `local x` 以及 `x = 0` 的作用是一样的。

因为单词 `local`, `global` 和 `static` 在脚本启动时会被立即处理，所以不能用 [IF 语句](#)来有条件地声明变量。换句话说，在一个 [IF](#) 或 [ELSE 区块](#)里的声明，声明与函数末尾大括号之间的所有行都会无条件地生效。同时注意目前还不可能声明一个动态变量比如 `global Array%i%`。

在一个函数里，任何动态变量引用比如 `Array%i%` 常常会解析为一个局部变量，除非那个名称的变量不存在，在这时才会使用全局变量如果它存在的话。如果两者都不存在，那么需要变量先被创建才能使用，它会被创建为一个局部变量，除非[假设全局模式](#)已生效。因此，仅当函数被定义为[假设全局](#)函数时，它才能够手动地创建一个全局[数组](#)(比如使用 `Array%i% := A_Index` 这样的方法)。

对于创建[数组](#)的命令(例如 [StringSplit](#))，如果[假设全局模式](#)未生效或者数组的首个元素已声明为局部变量(将函数的一个参数传递给它也可以 -- 即使那个参数是 [ByRef](#) 的 -- 因为参数和局部变量很相似)，那么得出的数组就是局部的。相反的，如果首个元素已被声明为全局，那么创建的是全局数组。[StringSplit](#) 的首个元素是 `ArrayName0`。对于其他创建数组的命令比如 [WinGet List](#)，首个元素是 `ArrayName`(即没有数字)。

常见疑点：在脚本启动时对变量的任何[非动态引用](#)都将创建那个变量。例如：在脚本启动的时候，在函数外使用 `MsgBox %Array1%` 将创建全局的 `Array1`。相反的，在脚本启动时，在函数内部使用 `MsgBox %Array1%` 将创建 `Array1` 作为函数局部变量之一(除非[假设全局](#)已生效)。

当在[表达式](#)中使用 `AND`, `OR` 和[三元运算符](#)时，他们为提高性能而优化(不管当前是否有函数调用)。通过拒绝计算一个表达式里那些不能影响它的最终结果的部分来实行优化操作。要阐明此观点，请看下例：

```
if (ColorName <> "" AND not FindColor(ColorName))
    MsgBox 没找到 %ColorName%.
```

在上例中，如果 `ColorName` 变量为空，`FindColor()` 函数永远不会被调用。这是因为 `AND` 左侧的结果将为 `false`，因此它的右侧不可能让最终结果输出为 `true`。

由于此特性，所以必须明白，如果在 `AND` 或 `OR` 右侧调用函数，函数可能永远不会产生任何副作用(例如改变一个全局变量的内容)。

同时也要注意在嵌套的 `AND` 和 `OR` 中串联的求值优化。例如，在下面的表达式里，每当 `ColorName` 为空，就只需最左侧的比较就能执行了。这是因为左侧的表达式已经能确保最终的结果：

```
if (ColorName = "" OR FindColor(ColorName, Region1) OR FindColor(ColorName,
Region2))
break ;搜索内容为空，或者已经有一个匹配。
```

从上面例子来看，任何耗时的函数一般都应该在 *AND* 或 *OR* 的右侧调用从而提高性能。这个技术还能用来阻止函数的某个参数在传递一个不恰当的值比如一个空字符串时函数被调用。

在 v1.0.46 及之后的版本，[三重条件运算符 \(?:\)](#) 也通过不计算丢失的分支来优化求值。

尽管一个函数不能包含其他函数的[定义](#)，但它可以包含子程序。与其他子程序一样，可使用 [Gosub](#) 来启动它们，[Return](#) 来返回结果(这时候 [Return](#) 属于 [Gosub](#) 而不是函数)。

已知限制：当前，在整个脚本中每个子程序的名称(标签)必须是独一无二的。如果存在重复的标签，程序将会通知你。

如果一个函数使用 [Gosub](#) 跳转到一个公共子程序（在函数括号以外的子程序），那么在外部的所有变量都是全局变量，而函数自己的[局部变量](#)在子程序返回之前将不可用。

虽然一般不鼓励使用 [Goto](#) 命令，但是它能用来在同个函数内跳转到函数的其他位置。这能帮助我们简化有很多返回点的复杂函数，所有那些返回点需要在返回之前做一些清理。

尽管 [Goto](#) 那些函数外部的目标被忽略，但是对函数来说可以用 [Gosub](#) 进入一个外部的/公用的子程序然后从那里使用 [Goto](#)。

一个函数可以包含从外部调用的子程序，比如[定时器](#), [GUI](#) 和[菜单项](#)。通常将它们封装在不同的文件中供 [#Include](#) 使用，这将防止它们干扰脚本的[自动执行部分](#)。不过，还有如下限制：

- 如果它们的函数是正常地调用，这样的子程序只能用[静态变量](#)和[全局变量](#)(不能是[局部变量](#))。这是因为一个子程序[线程](#)打断一个函数调用线程(或者反过来也一样)将能够改变被打断的线程看到的局部变量的值。此外，每当函数返回它的调用者时，它所有的局部变量都被清空从而释放它们的内存。
- 这样的子程序应该只将[全局变量](#)(不是[静态变量](#))用作 [GUI 控件变量](#)。
- 当一个子程序[线程](#)进入一个函数时，任何由此线程引用的[动态变量](#)将被作为[全局变量](#)对待(包括创建数组的命令)。

如果一个函数内部的执行流在到达函数的结束大括号之前遇上一个 [Return](#)，那么函数将结束并返回一个空值(空字符串)给它的调用者。每当函数明确地省略 [Return](#) 的参数时也将返回一个空值。

当一个函数使用 [Exit](#) 命令来结束[当前线程](#)的时候，它的调用者根本不会收到返回值。例如：在语句 `Var := Add(2, 3)` 中如果 `Add()` 退出，那么 `Var` 将不做改变。如果函数遇上运行错误比如[运行](#)了一个不存在的文件(当 [UseErrorLevel](#) 无效时)，那么也将发生同样的事情。

一个函数可以改变 [ErrorLevel](#) 的值为了返回一个更容易记忆的额外值。

要在调用函数时使用一个或多个空值(空字符串)，可以像这个例子一样使用一对空的双引号：

```
FindColor(ColorName, "")
```

因为调用函数不会启用新的线程，所以函数做出的任何设置比如 `SendMode` 和 `TitleMatchMode` 的改变也都会影响到它的调用者。

一个函数的调用者可能传递一个不存在的变量或者数组元素给它，这在函数期望这个相应的参数被 `ByRef` 时将变得很有用。例如：调用 `GetNextLine(BlankArray%i%)` 将自动地创建一个局部或全局变量 `BlankArray%i%` (根据调用者是否在函数内并且是否有假设的全局模式在起作用)。

当在一个函数里使用时，`ListVars` 会显示一个函数的局部变量和它们的内容。这能帮助我们调试脚本。

你会发现如果给复杂的函数的特定变量加一个独特的前缀，将使函数更易于阅读和维护。例如，用 "p" 或 "p_" 开头来命名函数的每个参数能让你一眼就辨别出它们的特性，特别是当函数还有很多局部变量来争着吸引你眼球的时候。相似地，"r" 或 "r_" 可以用在 `ByRef` 参数的前面，"s" 或 "s_" 可以用在静态变量的前面。

单个正确的大括号(OTB) 类型也可以用来定义函数。例如：

```
Add(x, y) {
    return x + y
}
```

可以用 `#Include` 指令(甚至在脚本的顶部)从外部文件加载函数。

说明：当脚本的执行流遇到函数定义时，它会跳过函数(用一种瞬间完成的方法)，并在它的结束大括号的下一行恢复执行。所以执行流不会从上面掉进一个函数，也不会因为在脚本的顶部有一个或多个函数而影响到自动执行部分。

一个脚本不一定非要使用 `#Include` 来调用外部文件中的函数。要达到这个目的，一个与函数同名的文件必须存在于下列的库目录之一中：

`%A_MyDocuments%\AutoHotkey\Lib\`; 用户库。此目录是可选的；也可以完全不存在。

当前运行的 `AutoHotkey.exe` 的路径`\Lib\`; 标准库。同样是可选的。

例如，假设一个脚本调用一个不存在的函数 `MyFunc()`，程序将在用户库里搜索名为 `MyFunc.ahk` 的文件。如果找不到，它将在标准库里继续搜索。如果仍未找到并且函数的名字里有一个下划线（比如 `MyPrefix_MyFunc`），那么程序将在两个库里搜索名为 `MyPrefix.ahk` 的文件，如果存在的话会加载它。这使得 `MyPrefix.ahk` 可以包含函数 `MyPrefix_MyFunc` 以及其他相关的函数名以 `MyPrefix_` 开头的函数。

虽然一个库文件通常只包含一个和它的文件名同名的函数，但它也可以包含仅被同名函数调用的私有函数和子程序。然而，这些函数应该具有相当独特的名称，因为它们仍会在全局命名空间里；也就是说，可以从脚本的任意位置调用它们。

如果一个库函数使用 **#Include**, 那么 **#Include** 的工作目录就是库函数自身所在的目录。这能被用来创建一个重定向到一个包含此函数和其他与之相关的函数的巨大的库文件。

脚本编译器 ([ahk2exe](#)) 同样支持库函数。不过, 它需要编译器目录的上层目录里存在一个复制的 `AutoHotkey.exe`(通常已是如此)。如果 `AutoHotkey.exe` 不存在, 编译器仍可以运行, 不过无法自动地包含库函数。

包含一个库函数也会执行地同其他函数一样好, 因为它们在脚本开始执行之前已经被预加载了。

在一个内置函数的参数列表末尾的任何可选的参数是可以被完全省略的。例如, `WinExist("Untitled - Notepad")` 是有效的, 因为它的另外三个参数将被视为空。

如果脚本定义了一个和内置函数同名的函数, 内置函数将被覆盖。例如: 一个脚本将调用它自定义的 `WinExist()` 函数来代替标准的那个。

存在于 `DLL` 文件里的外部函数可以通过 `DllCall()` 来调用。

经常使用的函数

18.8 IsLabel

IsLabel(LabelName): 如果 `LabelName` 是脚本中子程序, 热键或热字符串的标签名称(在 `LabelName` 中不要包括尾部的冒号), 则返回一个非零值。例如: 如果标签存在, 语句 `if IsLabel(VarContainingLabelName)` 将为 `true`, 反之就是 `false`。当你在 `Gosub`, `Hotkey`, `Menu` 和 `Gui` 这样的命令中指定了一个动态标签时, 这个函数对避免运行错误将十分有用。

ListView 和 **TreeView** 函数: 详见 [ListView](#) 和 [TreeView](#) 页面。

18.9 NumGet

NumGet(VarOrAddress [, Offset = 0, Type = "UInt"]) [v1.0.47+]: 返回储存在指定地址 + 偏移量中的二进制数。对于 `VarOrAddress`, 传递 `MyVar` 相当于传递 `&MyVar`。不过, 省略 `"&"` 可以执行地更好并确保目标地址是有效的(无效的地址将返回 `""`)。相比之下, 除了变量, 传递给 `VarOrAddress` 的其他任何东西都被当作原始地址; 因此, 指定 `MyVar+0` 会强制用 `MyVar` 中的数字来代替 `MyVar` 本身的地址。对于 `Type`, 可以指定为 `UInt`, `Int`, `Int64`, `Short`, `UShort`, `Char`, `UChar`, `Double` 或者 `Float`(不过与 `DllCall` 不同, 当这些作为原义字符串使用时必须被括在引号里); 详见 [DllCall Types](#)。

18.10 NumPut

NumPut(Number, VarOrAddress [, Offset = 0, Type = "UInt"]) [v1.0.47+]: 在指定地址 + 偏移量中以二进制的格式储存 `Number`, 并在刚刚写入的项目的右边返回地址。对于 `VarOrAddress`, 传递 `MyVar` 相当于传递 `&MyVar`。不过, 省略 `"&"` 可以执行地更好并确保目标地址是有效的(无效的地址将返回 `""`)。相比之下, 除了变量, 传递给 `VarOrAddress` 的其他任何东西都被当作原始地址; 因此,

指定 `MyVar+0` 会强制用 `MyVar` 中的数字来代替 `MyVar` 本身的地址。对于 `Type`, 可以指定为 `UInt`, `Int`, `Int64`, `Short`, `UShort`, `Char`, `UChar`, `Double` 或者 `Float`(不过与 `DllCall` 不同, 当这些作为原义字符串使用时必须被括在引号里); 详见 [DllCall Types](#)。如果一个整数太大而无法适应指定的 `Type`, 它最重要的字节就被忽略了; 比如 `NumPut(257, var, 0, "Char")` 将存储数字 1。

OnMessage(MsgNumber [, "FunctionName"]): 监控消息/事件。详见 [OnMessage\(\)](#)。

RegisterCallback(): 请看 [RegisterCallback\(\)](#)。

VarSetCapacity(UnquotedVarName [, RequestedCapacity, FillByte]): 扩大一个变量的容积或者释放它的内存。详见 [VarSetCapacity\(\)](#)。

18.11 OnMessage

Specifies a [function](#)(See 10.) to call automatically when the script receives the specified message.

`OnMessage(MsgNumber [, "FunctionName", MaxThreads])`

参数

MsgNumber	The number of the message to monitor or query, which should be between 0 and 4294967295 (0xFFFFFFFF). If you do not wish to monitor a system message (See 30.17) (that is, one below 0x400), it is best to choose a number greater than 4096 (0x1000) to the extent you have a choice. This reduces the chance of interfering with messages used internally by current and future versions of AutoHotkey.
FunctionName	A function (See 10.)'s name, which must be enclosed in quotes if it is a literal string. This function will be called automatically when the script receives <code>MsgNumber</code> . Omit this parameter and the next one to retrieve the name of the function currently monitoring <code>MsgNumber</code> (blank if none). Specify an empty string ("") or an empty variable to turn off the monitoring of <code>MsgNumber</code> .
MaxThreads [v1.0.47+]	This integer is normally omitted, in which case the monitor function is limited to one thread (See 30.19) at a time. This is usually best because otherwise, the script would process messages out of chronological order whenever the monitor function interrupts itself. Therefore, as an alternative to <code>MaxThreads</code> , consider using <code>Critical</code> as described below .

Return Values

If `FunctionName` and `MaxThreads` are omitted, it returns the name of the function currently monitoring `MsgNumber` (blank if none). However, no changes are made.

If `FunctionName` is explicitly blank (e.g. ""), it returns the name of the function currently monitoring `MsgNumber` (blank if none), then disables the monitoring of `MsgNumber`.

If *FunctionName* is non-blank: If *MsgNumber* is already being monitored, it returns the name of that function then puts the new function into effect. Otherwise, it assigns *FunctionName* to monitor *MsgNumber* then returns the same *FunctionName*. In either case, a blank value is returned upon failure. Failure occurs when *FunctionName*: 1) does not exist (perhaps due to missing quotes around *FunctionName*); 2) accepts more than four parameters; or 3) has any [ByRef](#)(See 10.) or [optional](#)(See 10.) parameters. It will also fail if the script attempts to monitor a new message when there are already 500 messages being monitored.

The Function's Parameters

A [function](#)(See 10.) assigned to monitor one or more messages can accept up to four parameters:

```
MyMessageMonitor(wParam, lParam, msg, hwnd)
{
    ... body of function...
}
```

Although the names you give the parameters do not matter, the following information is sequentially assigned to them:

Parameter #1: The message's WPARAM value, which is an integer between 0 and 4294967295.
 Parameter #2: The message's LPARAM value, which is an integer between 0 and 4294967295.
 Parameter #3: The message number, which is useful in cases where a function monitors more than one message.
 Parameter #4: The HWND (unique ID) of the window or control to which the message was sent.
 The HWND can be used with [ahk_id](#)(See 30.2).

You can omit one or more parameters from the end of the list if the corresponding information is not needed. For example, a function defined as *MyMsgMonitor(wParam, lParam)* would receive only the first two parameters, and one defined as *MyMsgMonitor()* would receive none of them.

If an incoming WPARAM or LPARAM is intended to be a signed integer, any negative numbers can be revealed by following this example:

```
if wParam > 0x7FFFFFFF
    wParam := -(~wParam) - 1
```

Additional Information Available to the Function

In addition to the parameters received above, the function may also consult the values in the following built-in variables:

- [A_Gui](#)(See 9.): Blank unless the message was sent to a GUI window or control, in which case A_Gui is the [Gui Window number](#)(See 19.3) (this window is also set as the function's [default GUI window](#)(See 19.3)).
- [A_GuiControl](#)(See 9.): Blank unless the message was sent to a GUI control, in which case it contains the control's variable name or other value as described at [A_GuiControl](#)(See 9.). Some controls never receive certain types of messages. For example, when the user clicks a [text control](#)(See 19.4), the operating system sends WM_LBUTTONDOWN to the parent window rather than the control; consequently, A_GuiControl is blank.
- [A_GuiX](#)(See 9.) and [A_GuiY](#)(See 9.): Both contain -2147483648 if the incoming message was sent via [SendMessage](#)(See 28.1.12). If it was sent via [PostMessage](#)(See 28.1.12), they contain the mouse cursor's coordinates (relative to the screen) at the time the message was posted.
- [A_EventInfo](#)(See 9.): Contains 0 if the message was sent via SendMessage. If sent via PostMessage, it contains the [tick-count time](#)(See 9.) the message was posted.

A monitor function's [last found window](#)(See 30.2) starts off as the parent window to which the message was sent (even if it was sent to a control). If the window is hidden but not a GUI window (such as the script's main window), turn on [DetectHiddenWindows](#)(See 28.5) before using it. For example:

```
DetectHiddenWindows On

MsgParentWindow := WinExist() ; This stores the unique ID of the window to which
                           the message was sent.
```

What the Function Should *Return*

If a monitor function uses [Return](#)(See 17.19) without any parameters, or it specifies a blank value such as "" (or it never uses Return at all), the incoming message goes on to be processed normally when the function finishes. The same thing happens if the function [Exits](#)(See 17.6) or causes a runtime error such as [running](#)(See 24.5) a nonexistent file. By contrast, returning an integer between -2147483648 and 4294967295 causes that number to be sent immediately as a reply; that is, the program does not process the message any further. For example, a function monitoring WM_LBUTTONDOWN (0x201) may return an integer to prevent the target window from being notified that a mouse click has occurred. In many cases (such as a message arriving via [PostMessage](#)(See 28.1.12)), it does not matter which integer is returned; but if in doubt, 0 is usually safest.

General Remarks

Unlike a normal function-call, the arrival of a monitored message calls the function as a new [thread](#)(See 30.19). Because of this, the function starts off fresh with the default values for settings such as [SendMode](#)(See 20.13) and [DetectHiddenWindows](#)(See 28.5). These defaults can be changed in the [auto-execute section](#)(See 8.).

Messages sent to a control (rather than being posted) are not monitored because the system routes them directly to the control behind the scenes. This is seldom an issue for system-generated messages because most of them are posted.

Any script that calls `OnMessage` anywhere is automatically [persistent](#)(See 29.21). It is also single-instance unless [#SingleInstance](#) (See 22.2)has been used to override that.

If a message arrives while its function is still running due to a previous arrival of the same message, the function will not be called again (except if `MaxThreads` is greater than 1); instead, the message will be treated as unmonitored. If this is undesirable, a message greater than or equal to 0x312 can be buffered until its function completes by specifying [Critical](#)(See 22.7) as the first line of the function. Alternatively, [Thread Interrupt](#)(See 22.22) can achieve the same thing as long as it lasts long enough for the function to finish. By contrast, a message less than 0x312 cannot be buffered by Critical or Thread Interrupt (however, in v1.0.46+, Critical may help because it checks messages [less often](#)(See 22.7), which gives the function more time to finish). The only way to guarantee that no such messages are missed is to ensure the function finishes in under 6 milliseconds (though this limit can be raised via [Critical 30](#)(See 22.7)). One way to do this is to have it queue up a future thread by [posting](#)(See 28.1.12) to its own script a monitored message number higher than 0x312. That message's function should use [Critical](#)(See 22.7) as its first line to ensure that its messages are buffered.

If a monitored message that is numerically less than 0x312 arrives while the script is absolutely uninterruptible -- such as while a [menu](#)(See 22.13) is displayed, a [KeyDelay](#)(See 20.14)/[MouseDelay](#)(See 23.8) is in progress, or the clipboard is being [opened](#)(See 29.2) -- the message's function will not be called and the message will be treated as unmonitored. By contrast, a monitored message of 0x312 or higher will be buffered during these uninterruptible periods; that is, its function will be called when the script becomes interruptible.

If a monitored message numerically less than 0x312 arrives while the script is uninterruptible merely due to the settings of [Thread Interrupt](#)(See 22.22) or [Critical](#)(See 22.7), the current thread will be interrupted so that the function can be called. By contrast, a monitored message of 0x312 or higher will be buffered until the thread finishes or becomes interruptible.

The [priority](#)(See 30.19) of `OnMessage` threads is always 0. Consequently, no messages are monitored or buffered when the current thread's priority is higher than 0.

Caution should be used when monitoring system messages (those below 0x400). For example, if a monitor function does not finish quickly, the response to the message might take longer than the system expects, which might cause side-effects. Unwanted behavior may also occur if a monitor function returns an integer to suppress further processing of a message, but the system expected different processing or a different response.

When the script is displaying a system dialog such as [MsgBox](#)(See 19.11), any message posted to a control is not monitored. For example, if the script is displaying a MsgBox and the user clicks a button in a GUI window, the WM_LBUTTONDOWN message is sent directly to the button without calling the monitor function.

Although an external program may post messages directly to a script's thread via `PostThreadMessage()` or other API call, this is not recommended because the messages would be lost if the script is displaying a system window such as a [MsgBox](#)(See 19.11). Instead, it is usually best to post or send the messages to the script's main window or one of its GUI windows.

相关命令

[RegisterCallback\(\)](#)(See 18.14), [OnExit](#)(See 17.17), [OnClipboardChange](#)(See 30.6),
[Post/SendMessage](#)(See 28.1.12), [Functions](#)(See 10.), [List of Windows Messages](#)(See 30.17),
[Threads](#)(See 30.19), [Critical](#)(See 22.7), [DllCall\(\)](#)(See 18.3)

示例

```
; Example: The following is a working script that monitors mouse clicks in a GUI window.

; Related topic: GuiContextMenu(See 19.3)

Gui, Add, Text,, Click anywhere in this window.

Gui, Add, Edit, w200 vMyEdit

Gui, Show

OnMessage(0x201, "WM_LBUTTONDOWN")

return


WM_LBUTTONDOWN(wParam, lParam)
{
    X := lParam & 0xFFFF
    Y := lParam >> 16
    if A_GuiControl
        Control := `n(in control " . A_GuiControl . ")"
    ToolTip You left-clicked in Gui window #`A_Gui% at client
    coordinates %X%x%Y%.%Control%
}

GuiClose:
```

ExitApp

; Example: The following script detects system shutdown/logoff and allows you to abort it (this is

; reported NOT to work on Windows Vista).

; Related topic: [OnExit\(See 17.17\)](#)

; The following DllCall is optional: it tells the OS to shut down this script *first* (prior to all other applications).

; This call has no effect on Windows 9x.

```
DllCall("kernel32.dll\SetProcessShutdownParameters", UInt, 0x4FF, UInt, 0)
```

```
OnMessage(0x11, "WM_QUERYENDSESSION")
```

```
return
```

```
WM_QUERYENDSESSION(wParam, lParam)
```

```
{
```

```
    ENDSESSION_LOGOFF = 0x80000000
```

```
    if (lParam & ENDSESSION_LOGOFF) ; User is logging off.
```

```
        EventType = Logoff
```

```
    else ; System is either shutting down or restarting.
```

```
        EventType = Shutdown
```

```
    MsgBox, 4,, %EventType% in progress. Allow it?
```

```
IfMsgBox Yes
```

```
    return true ; Tell the OS to allow the shutdown/logoff to continue.
```

```
else
```

```
    return false ; Tell the OS to abort the shutdown/logoff.
```

```
}
```

; Example: Have a script receive a custom message and up to two numbers from some other script or program

; (to send strings rather than numbers, see the example after this one).

```
OnMessage(0x5555, "MsgMonitor")
OnMessage(0x5556, "MsgMonitor")

MsgMonitor(wParam, lParam, msg)
{
    ; Since returning quickly is often important, it is better to use a ToolTip than
    ; something like MsgBox that would prevent the function from finishing:
   ToolTip Message %msg% arrived: `nWPARAM: %wParam%`nLPARAM: %lParam%
}
```

; The following could be used inside some other script to run the function inside the above script:

```
SetTitleMatchMode 2
DetectHiddenWindows On
if WinExist("Name of Receiving Script.ahk ahk_class AutoHotkey")
    PostMessage, 0x5555, 11, 22 ; The message is sent to the "last found window(See
                                ; 30.2)" due to WinExist() above.
DetectHiddenWindows Off ; Must not be turned off until after PostMessage.
```

; Example: Send a string of any length from one script to another. This is a working example.

; To use it, save and run both of the following scripts then press Win+Space to show an InputBox that will prompt you to type in a string.

; Save the following script as "**Receiver.ahk**" then launch it:

```
#SingleInstance
OnMessage(0x4a, "Receive_WM_COPYDATA") ; 0x4a is WM_COPYDATA
return
```

```

Receive_WM_COPYDATA(wParam, lParam)
{
    StringAddress := NumGet(lParam + 8) ; lParam+8 is the address of
CopyDataStruct's lpData member.

    StringLength := DllCall("lstrlen", UInt, StringAddress)

    if StringLength <= 0

        ToolTip %A_ScriptName%`nA blank string was received or there was an error.

    else

    {

        VarSetCapacity(CopyOfData, StringLength)

        DllCall("lstrcpy", "str", CopyOfData, "uint", StringAddress) ; Copy the string out
of the structure.

        ; Show it with ToolTip vs. MsgBox so we can return in a timely fashion:

        ToolTip %A_ScriptName%`nReceived the following string:`n%CopyOfData%

    }

    return true ; Returning 1 (true) is the traditional way to acknowledge this message.
}

; Save the following script as "Sender.ahk" then launch it. After that, press the
Win+Space hotkey.

TargetScriptTitle = Receiver.ahk ahk_class AutoHotkey

#space:: ; Win+Space hotkey. Press it to show an InputBox for entry of a message
string.

InputBox, StringToSend, Send text via WM_COPYDATA, Enter some text to Send:

if ErrorLevel ; User pressed the Cancel button.

    return

result := Send_WM_COPYDATA(StringToSend, TargetScriptTitle)

if result = FAIL

```

```

    MsgBox SendMessage failed. Does the following WinTitle
exist?: `n%TargetWindowTitle%`n

else if result = 0

    MsgBox Message sent but the target window responded with 0, which may mean it
ignored it.

return

Send_WM_COPYDATA(ByRef StringToSend, ByRef TargetWindowTitle) ; ByRef saves a
little memory in this case.

; This function sends the specified string to the specified window and returns the reply.

; The reply is 1 if the target window processed the message, or 0 if it ignored it.

{

    VarSetCapacity(CopyDataStruct, 12, 0) ; Set up the structure's memory area.

    ; First set the structure's cbData member to the size of the string, including its zero
terminator:

    NumPut(StrLen(StringToSend) + 1, CopyDataStruct, 4) ; OS requires that this be
done.

    NumPut(&StringToSend, CopyDataStruct, 8) ; Set lpData to point to the string itself.

    Prev_DetectHiddenWindows := A_DetectHiddenWindows

    Prev_TitleMatchMode := A_TitleMatchMode

    DetectHiddenWindows On

    SetTitleMatchMode 2

    SendMessage, 0x4a, 0, &CopyDataStruct,, %TargetWindowTitle% ; 0x4a is
WM_COPYDATA. Must use Send not Post.

    DetectHiddenWindows %Prev_DetectHiddenWindows% ; Restore original setting for
the caller.

    SetTitleMatchMode %Prev_TitleMatchMode% ; Same.

    return ErrorLevel ; Return SendMessage's reply back to our caller.

}

; Example: See the WinLIRC client script(See 30.36) for a demonstration of how to use
OnMessage() to receive

```

; notification when data has arrived on a network connection.

18.12 RegExMatch

确定一个字符串中所包含的匹配模式 (即: 正则表达式)。

```
FoundPos := RegExMatch(Haystack, NeedleRegEx [, UnquotedOutputVar = "", StartingPosition = 1])
```

参数

FoundPos	<code>RegExMatch()</code> 返回从源字符串 <i>Haystack</i> 最左边开始找到的第一个匹配 <i>NeedleRegEx</i> 模式的位置。首字符的位置是 1。要是在字符串中没有找到匹配模式的时候就会返回 0。如果出现了错误 (比如: 正则表达式 <i>NeedleRegEx</i> 的语法错误), 它将会返回一个空的字符串和 <i>ErrorLevel</i> 。 <i>ErrorLevel</i> 将是 下面 值中的一个, 包括 0。
Haystack	源字符串。
NeedleRegEx	这种匹配模式是和 Perl 兼容的正则表达式 (PCRE)。如果有必要的话, 可以在正则表达式前加上 选项 , 并以 ")" 结束。例如: 这个匹配模式 " <i>i</i> abc.*123" 将匹配不区分大小写的 "abc" , 中间可以是任何字符, 并以 123 结尾的模式进行搜索。要是没有 选项 , 这个 ")" 是可选的, 例如: ")abc" 就等同于 "abc" 。
UnquotedOutputVar	<p>形式 1 (默认): <i>OutputVar</i> 是 unquoted 的一个变量名, 它将存储 <i>Haystack</i> 中符合匹配模式的那一部分字符串。如果没有找到符合这个模式的子串 (那么, <code>RegExMatch()</code> 将返回 0), 而输出变量和所有的数组元素将被置为空。</p> <p>如果在 <i>NeedleRegEx</i> 中包含 捕捉子模式(See 30.14) , 那么它们将存储在以 <i>OutputVar</i> 为基变量的 数组(See 30.5) 里。例如: 输出的变量名是 <i>Match</i> , 那它第一个能匹配子模式的子串将存储在 <i>Match1</i> , 第二个存储在 <i>Match2</i> , 等等。有一个例外是 命名子模式 : 它用名字代替了数字的形式来存储。例如: 这种模式 <code>(?P<Year>)d{4}</code> , 会把输出的子串存储在 <i>MatchYear</i> 里面。如果没有任何字符串 (或者此函数返回的值是 0), 那么这一输出变量置为空。</p> <p>在 函数(See 10.) 中创建一个全局变量的数组来替代局部变量, 就要在使用前 声明(See 10.) 数组名 (比如刚才的 <i>Match</i>) 为全局变量。</p> <p>形式 2 (位置 - 长度): 若以 P 开头的正则表达式, 比如, "<i>P</i>abc.*123" , 则完全匹配正则表达式的字符串的 长度 将存储在 <i>OutputVar</i> 里 (没有匹配到, 长度就是 0)。如果现在使用了 捕捉子模式(See 30.14) , 就会有两个数组 <i>OutputVarPos</i> 和 <i>OutputVarLen</i> 分别存储匹配到的字符串的位置和长度。例如: 若基变量名为 <i>Match</i> , 则符合模式的第一个子串的位置存储在 <i>MatchPos1</i> , 长度存储在 <i>MatchLen1</i> (没到匹配到任何子串则全部返回 0 , 函数值也会为 0)。仍有一个例外, 就是 命名子模式: 它将存储名用名字替代了数字 (比如, <i>MatchPosYear</i> 和 <i>MatchLenYear</i>)。</p>

	如果省略 <i>StartingPosition</i> 的值，它会默认为 1 (从源字符串 <i>Haystack</i> 的第一个字符开始)。另外，设置为 2 则以第二个字符开始，值为 3 就从第三个开始，依次类推。如果 <i>StartingPosition</i> 的值超出了 <i>Haystack</i> 的长度范围，则函数返回为 0，变量值为空。
StartingPosition	<p>如果 <i>StartingPosition</i> 的值小于 1，函数会认为是从 <i>Haystack</i> 的末尾处开始。例如：0 就是倒数一个字符的位置，-1 就是倒数第二个字符，依次类推。如果 <i>StartingPosition</i> 超过了 <i>Haystack</i> 最左边的位置，它就会搜索整个 <i>Haystack</i>。</p> <p>无论 <i>StartinPosition</i> 的值是什么，函数返回的值都是根据 <i>Haystack</i> 的第一个字符所确定的。例如："123abc789" 中 "abc" 的位置总是 4。</p>

ErrorLevel

[ErrorLevel](#)(See 30.8) 的值是下面中的一个：

- 0, 表示没发现错误。
- 一个有如下格式的字符串：*Complie error N at offset M: description*. 在这个字符串里，N 是 PCRE (Perl 兼容正则表达式) 的错误值，M 是正则表达式中出现错误的位置，*description* 是描述这个错误的文本信息。
- 一个负数，是指在 执行 正则表达式过程中出现的错误。尽管这种错误非常罕见，如这些情况就容易出现这种错误：“太多的空匹配” (-22)，“递归太深” (-21)，“太到匹配极限” (-8)。如果出现了这些错误，就试着重新写更严格的正则表达式，如用 ?, + 或者用有限制的方式，如 {0,3} 来取代 *。

选项（区分大小写）

在正则表达式之前加上 0 个或者多个选项并以右括号结束。例如：模式 "im)abc" 将搜索多行中不区分大小写的 abc，若没有选项可以省略圆括号。尽管这将原来的语法打断，但是它不需要新的前分割符 (比如右斜杠)，因此没有必要转换模式内的分割符。另外，由于使用了选项，函数性能有所提升。

i	不区分大小写。
m	<p>多行是指由多个单行 (有换行符) 所组成的一个集合 (它包括换行符)。但在下列情况会有所改变：</p> <ol style="list-style-type: none"> 1) 弯折符号 (^) 在所有行内匹配 -- 总是从源字符串开头的地方开始匹配 (但不是匹配源字符串 <i>Haystack</i> 中换行 靠后 的地方)。 2) 美元符号 (\$) 在源字符串中任意换行符之前匹配 (也就是说，它总是匹配每行靠近结尾的地方)。 <p>例如：源字符串是 "xyz`r`nabc" 中的 abc 要用模式 "m)^abc\$" 来精确匹配。</p> <p>如果选项是 "D"，它就将会忽略当前的 "m"。</p>
s	该选项会使点(.) 能匹配一行内所有的字符 (通常不会换行符后的)。如果换行符是默认的 CRLF (`r`n)，就要用两个点去匹配它 (而不是一个点)。无论这个选项如何，[^a] 总是匹配换行符。

x	忽略模式中空白字符类。如字符 `n 和 `t 这类的字符将被在正则表达式中的忽略，这些字符都是出现在正则表达式中（与此相反，使用 \n 和 \t 之类的字符就不会被正则表达式忽略）。x 选项也是一个复杂的模式。然而，这种模式必须遵循这个条件：只能 应用在数据字符中；空白字符也许不会出现特殊的序列，如 ?(。
A	强制将匹配模式固定；也就是说，它只能从源字符串的第一个字符开始匹配。大多数情况下，和 "^\n" 的功能一样。
D	强制用 (\$) 的方式来从靠近源字符串末尾的地方开始匹配，即使源字符串以的换行符结尾的。没有这个选项时，\$ 就只能匹配换行符（如果有的话）前的字符了。注意：这个选项会让 "m" 选项不起作用。
J	允许重复 命名子模式 。这是一个很有用的模式，它在一组相同命名子模式下也能够匹配。注意：如果有多个实例命名去匹配字符串的话，就只有最左边的那个命名才会被存储（变量名不区分大小写）。
U	非贪婪的。只有在绝对必要的的情况下才会使用量词 *+?{}，并将剩下的部分提供给下一个模式。当 "U" 选项无效时，可以通过一个问号作为量词进行非贪婪匹配。相反地，当 "U" 选项有效，问号就会进行贪婪搜索。
X	PCRE 扩展。它使 PCRE 不完全符合 Perl 的正则表达式。目前，仅仅只有一个这样的特点，那就是在模式中的任何的反斜杠后加一个字符就没有特殊的意义并导致匹配失败和设置 ErrorLevel 相应的值。这个选项帮助保留未使用的反斜杠序列供以后使用。没有这个选项，反斜杠后的字符就会看成一般的字符而没有特殊的意义（例如：\g 和 g 都被认为是字母 g）。不论此选项怎样，非字母反斜杠序列都没有特别的意义，它将总是认为是普通的字符（例如：\V 和 / 都被认为是 /）。
P	位置模式。它会使函数 RegExMatch() 产生匹配到的子串的位置和长度。更详细的了解，请看上面的 输出变量 。
S	研究模式可以提高函数的性能。当匹配模式比较复杂或者被重复执行很多次的情况下，这个选项很有用。它会让匹配模式存储在高速缓存中，以便下次使用，通过这种方法来提高了正则表达式的性能。
`n	替换默认的换行符 (`r`n) 为符合 UNIX 系统的标准的 (`n)。所选择的换行符会影响到 anchors (^ and \$) 和 dot/period pattern。
`r	替换默认的换行符 (`r`n) 为符合 Windwos 标准的 (`r)。
`a	在 1.0.46.06+ 的版本中，`a 看作是换行符，也就是指这些符号，`r, `n, `r`n, `v/VT/vertical tab/chr(0xB), `f/FF/formfeed/chr(0xC), and NEL/next-line/chr(0x85)。在 1.0.47.05+ 的版本中，换行符被限制为 CR, LF 和 CRLF 为 (*ANYCRLF)，在选项后的正则表达式之前大写；例如：im)(*ANYCRLF)^abc\$

注意：可用空格和 Tab 随意的隔开每个选项。

性能

如果从一个较长的字符串中搜索一个简单的子串，请使用 [InStr\(\)](#)(See 10.) 函数，因为它比 [RegExMatch\(\)](#) 更有效率。

为了提高性能，它将最近使用的 100 个正则表达式缓存在内存里（在编译的时候）。

当一个正则表达式重复使用的情况下，使用 [S 选项](#) 来提高性能。（比如在 loop 循环中）

注意

一个命名子模式都有一个名字，如上文模式 `(?P<Year>\d{4})` 中的 `Year`。这些名字可能包括多达 32 个字母、数字和下划线。虽然这些命名子模式都是通过数字在对正则表达式本身的操作（如：[向后引用\(See 18.13\)](#)），结果是仅仅用名字存储 [输出数组](#)，而不是用数字。例如 "Year" 是第一个名字，把匹配到的子串存储在 `OutputVarYear` 里，但是 `OutputVar1` 的值是没有改变（它将保留以前的值，如果有的话）。如果是 [非命名子模式\(See 30.14\)](#) 的 "Year"，它将匹配到的子串存储在 `OutputVar2` 里，而不是 `OutputVar1` 里。

大多数的字符串（像 `abc123`）就能用一般的字符来匹配。而匹配像 `\.*?+[{|()^$` 这样被保护的字符就要在其前面加是一个反斜杠。比如：`\.` 代表一个点，`\\\` 代表一个反斜杠。转义符可避免使用 `\Q...\E`。比如 `\QLiteral Text\E`。

在正则表达式中，一些特殊的字符，如制表符和换行符就用重音符号 (`) 或反斜杠 (\) 来表示。例如：`t is the same as \t。

如学习正则表达式的基础（或者重新记忆一下正则表达式的语法），请看 [RegEx Quick Reference\(See 30.14\)](#)。

AutoHotKey 所使用的正则表达式是来自于 www.pcre.org 的兼容 Perl 语言的正则表达式。

相关命令

[RegExReplace\(\)\(See 18.13\)](#), [RegEx Quick Reference\(See 30.14\)](#), [InStr\(\)\(See 10.\)](#),
[IfInString\(See 27.3\)](#), [StringGetPos\(See 27.14\)](#), [SubStr\(\)\(See 10.\)](#), [SetTitleMatchMode RegEx\(See 28.8\)](#), [Global matching and Grep \(forum link\)](#)

Common sources of text data: [FileRead\(See 16.19\)](#), [UrlDownloadToFile\(See 22.24\)](#),
[Clipboard\(See 30.6\)](#), [GUI Edit controls\(See 19.4\)](#)

示例

```
FoundPos := RegExMatch("xxxabc123xyz", "abc.*xyz") ; 返回值 4，它就是匹配到的位置。
FoundPos := RegExMatch("abc123123", "123$") ; 返回值 7，因为 $ 要求从靠近最后的字符处开始匹配。
FoundPos := RegExMatch("abc123", "i)^ABC") ; 返回值 1，因为是要求从第一个字符处开始匹配，而且不区分大小写。
FoundPos := RegExMatch("abcXYZ123", "abc(.*)123", SubPat) ; 返回 1 并且 SubPat1 的值是 "XYZ"。
FoundPos := RegExMatch("abc123abc456", "abc\d+", "", 2) ; 返回值 7，由于它是从第二个字符开始匹配的。
;
; 更多正则表达式的例子，请看 RegEx Quick Reference\(See 30.14\).
```

18.13 RegExReplace

确定一个字符串中所包含的匹配模式（即：正则表达式）。

```
FoundPos := RegExMatch(Haystack, NeedleRegEx [, UnquotedOutputVar = "", StartingPosition = 1])
```

参数

FoundPos	<code>RegExMatch()</code> 返回从源字符串 <i>Haystack</i> 最左边开始找到的第一个匹配 <i>NeedleRegEx</i> 模式的位置。首字符的位置是 1。要是在字符串中没有找到匹配模式的时候就会返回 0。如果出现了错误（比如：正则表达式 <i>NeedleRegEx</i> 的语法错误），它将会返回一个空的字符串和 <i>ErrorLevel</i> 。 <i>ErrorLevel</i> 将是 下面 值中的一个，包括 0。
Haystack	源字符串。
NeedleRegEx	这种匹配模式是和 Perl 兼容的正则表达式 (PCRE)。如果有必要的话，可以在正则表达式前加上 选项 ，并以 ")" 结束。例如：这个匹配模式 " <i>i</i> abc.*123" 将匹配不区分大小写的 "abc"，中间可以是任何字符，并以 123 结尾的模式进行搜索。要是没有 选项 ，这个 ")" 是可选的，例如："abc" 就等同于 "abc"。
UnquotedOutputVar	<p>形式 1 (默认): <i>OutputVar</i> 是 unquoted 的一个变量名，它将存储 <i>Haystack</i> 中符合匹配模式的那一部分字符串。如果没有找到符合这个模式的子串（那么，<code>RegExMatch()</code> 将返回 0），而输出变量和所有的数组元素将被置为空。</p> <p>如果在 <i>NeedleRegEx</i> 中包含 捕捉子模式(See 30.14)，那么它们将存储在以 <i>OutputVar</i> 为基变量的 数组(See 30.5) 里。例如：输出的变量名是 <i>Match</i>，那它第一个能匹配子模式的子串将存储在 <i>Match1</i>，第二个存储在 <i>Match2</i>，等等。有一个例外是 命名子模式：它用名字代替了数字的形式来存储。例如：这种模式 <code>(?P<Year>\d{4})</code>，会把输出的子串存储在 <i>MatchYear</i> 里面。如果没有任何字符串（或者此函数返回的值是 0），那么这一输出变量置为空。</p> <p>在 函数(See 10.) 中创建一个全局变量的数组来替代局部变量，就要在使用前 声明(See 10.) 数组名（比如刚才的 <i>Match</i>）为全局变量。</p> <p>形式 2 (位置 - 长度): 若以 P 开头的正则表达式，比如，"<i>P</i>abc.*123"，则完全匹配正则表达式的字符串的 长度 将存储在 <i>OutputVar</i> 里（没有匹配到，长度就是 0）。如果现在使用了 捕捉子模式(See 30.14)，就会有两个数组 <i>OutputOverPos</i> 和 <i>OutputVarLen</i> 分别存储匹配到的字符串的位置和长度。例如：若基变量名为 <i>Match</i>，则符合模式的第一个子串的位置存储在 <i>MatchPos1</i>，长度存储在 <i>MatchLen1</i>（没到匹配到任何子串则全部返回 0，函数值也会为 0）。仍有一个例外，就是 命名子模式：它将存储名用名字替代了数字（比如，<i>MatchPosYear</i> 和 <i>MatchLenYear</i>）。</p>
StartingPosition	如果省略 <i>StartingPosition</i> 的值，它会默认为 1（从源字符串 <i>Haystack</i> 的第一个字符开始）。另外，设置为 2 则以第二个字符开始，值为 3 就从第三个开始，依

	<p>次类推。如果 <i>StartingPosition</i> 的值超出了 <i>Haystack</i> 的长度范围，则函数返回为 0，变量值为空。</p> <p>如果 <i>StartingPosition</i> 的值小于 1，函数会认为是从 <i>Haystack</i> 的末尾处开始。例如：0 就是倒数一个字符的位置，-1 就是倒数第二个字符，依次类推。如果 <i>StartingPosition</i> 超过了 <i>Haystack</i> 最左边的位置，它就会搜索整个 <i>Haystack</i>。</p> <p>无论 <i>StartinPosition</i> 的值是什么，函数返回的值都是根据 <i>Haystack</i> 的第一个字符所确定的。例如："123abc789" 中 "abc" 的位置总是 4。</p>
--	--

ErrorLevel

[ErrorLevel](#)(See 30.8) 的值是下面中的一个：

- 0，表示没发现错误。
- 一个有如下格式的字符串：*Complie error N at offset M: description.* 在这个字符串里，N 是 PCRE (Perl 兼容正则表达式) 的错误值，M 是正则表达式中出现错误的位置，description 是描述这个错误的文本信息。
- 一个负数，是指在 执行 正则表达式过程中出现的错误。尽管这种错误非常罕见，如这些情况就容易出现这种错误：“太多的空匹配”(-22), “递归太深”(-21), “太到匹配极限”(-8)。如果出现了这些错误，就试着重新写更严格的正则表达式，如用 ? , + 或者用有限制的方式，如 {0,3} 来取代 * 。

选项 (区分大小写)

在正则表达式之前加上 0 个或者多个选项并以右括号结束。例如：模式 "im)abc" 将搜索多行中不区分大小写的 abc，若没有选项可以省略圆括号。尽管这将原来的语法打断，但是它不需要新的前分割符（比如右斜杠），因此没有必要转换模式内的分割符。另外，由于使用了选项，函数性能有所提升。

i	不区分大小写。
m	<p>多行是指由多个单行（有换行符）所组成的一个集合（它包括换行符）。但在下列情况会有所改变：</p> <p>1) 弯折符号 (^) 在所有行内匹配 -- 总是从源字符串开头的地方开始匹配（但不是匹配源字符串 <i>Haystack</i> 中换行 靠后 的地方）。</p> <p>2) 美元符号 (\$) 在源字符串中任意换行符之前匹配（也就是说，它总是匹配每行靠近结尾的地方）。</p> <p>例如：源字符串是 "xyz`r`nabc" 中的 abc 要用模式 "m)^abc\$" 来精确匹配。</p> <p>如果选项是 "D"，它就将会忽略当前的 "m"。</p>
s	该选项会使点(.) 能匹配一行内所有的字符（通常不会换行符后的）。如果换行符是默认的 CRLF (`r`n)，就要用两个点去匹配它（而不是一个点）。无论这个选项如何，[^a] 总是匹配换行符。
x	忽略模式中空白字符类。如字符 `n 和 `t 这类的字符将被在正则表达式中的忽略，这些字符都是出现在正则表达式中（与此相反，使用 \n 和 \t 之类的字符就不会被正则表达式忽略）。x 选项也是一个复杂的模式。然而，这种模式必须遵循这个条件：只能 应用在数据字符中；空白字符也许不会出现特

	殊的序列，如 <code>(?()</code> 。
A	强制将匹配模式固定；也就是说，它只能从源字符串的第一个字符开始匹配。大多数情况下，和 " <code>^</code> " 的功能一样。
D	强制用 <code>(\$)</code> 的方式来从靠近源字符串末尾的地方开始匹配，即使源字符串以的换行符结尾的。没有这个选项时， <code>\$</code> 就只能匹配换行符（如果有的话）前的字符了。注意：这个选项会让 " <code>m</code> " 选项不起作用。
J	允许重复 命名子模式 。这是一个很有用的模式，它在一组相同命名子模式下也能够匹配。注意：如果有多个实例命名去匹配字符串的话，就只有最左边的那个命名才会被存储（变量名不区分大小写）。
U	非贪婪的。只有在绝对必要的的情况下才会使用量词 <code>*+?{}</code> ，并将剩下的部分提供给下一个模式。当 " <code>U</code> " 选项无效时，可以通过一个问号作为量词进行非贪婪匹配。相反地，当 " <code>U</code> " 选项有效，问号就会进行贪婪搜索。
X	PCRE 扩展。它使 PCRE 不完全符合 Perl 的正则表达式。目前，仅仅只有一个这样的特点，那就是在模式中的任何的反斜杠后加一个字符就没有特殊的意义并导致匹配失败和设置 <code>ErrorLevel</code> 相应的值。这个选项帮助保留未使用的反斜杠序列供以后使用。没有这个选项，反斜杠后的字符就会看成一般的字符而没有特殊的意义（例如： <code>\g</code> 和 <code>g</code> 都被认为是字母 <code>g</code> ）。不论此选项怎样，非字母反斜杠序列都没有特别的意义，它将总是认为是普通的字符（例如： <code>\V</code> 和 <code>/</code> 都被认为是 <code>/</code> ）。
P	位置模式。它会使函数 <code>RegExMatch()</code> 产生匹配到的子串的位置和长度。更详细的了解，请看上面的 输出变量 。
S	研究模式可以提高函数的性能。当匹配模式比较复杂或者被重复执行很多次的情况下，这个选项很有用。它会让匹配模式存储在高速缓存中，以便下次使用，通过这种方法来提高了正则表达式的性能。
<code>'n</code>	替换默认的换行符（` <code>r`n`</code> ）为符合 UNIX 系统的标准的（` <code>n`））。所选择的换行符会影响到 anchors (^ and \$) 和 dot/period pattern。</code>
<code>'r</code>	替换默认的换行符（` <code>r`n`</code> ）为符合 Windwos 标准的（` <code>r`）。</code>
<code>'a</code>	在 1.0.46.06+ 的版本中，` <code>a</code> 看作是换行符，也就是指这些符号，` <code>r</code> , ` <code>n</code> , ` <code>r`n`, `v/VT/vertical tab/chr(0xB), `f/FF/formfeed/chr(0xC), and NEL/next-line/chr(0x85)。在 1.0.47.05+ 的版本中，换行符被限制为 CR, LF 和 CRLF 为 (*ANYCRLF)，在选项后的正则表达式之前大写；例如：im)(*ANYCRLF)^abc\$</code>

注意：可用空格和 Tab 随意的隔开每个选项。

性能

如果从一个较长的字符串中搜索一个简单的子串，请使用 [InStr\(\)](#)(See 10.) 函数，因为它比 `RegExMatch()` 更有效率。

为了提高性能，它将最近使用的 100 个正则表达式缓存在内存里（在编译的时候）。

当一个正则表达式重复使用的情况下，使用 [S 选项](#) 来提高性能。（比如在 `loop` 循环中）

注意

一个命名子模式都有一个名字，如上文模式 `(?P<Year>\d{4})` 中的 `Year`。这些名字可能包括多达 32 个字母、数字和下划线。虽然这些命名子模式都是通过数字在对正则表达式本身的操作（如：[向后引用\(See 18.13\)](#)），结果是仅仅用名字存储 [输出数组](#)，而不是用数字。例如 "Year" 是第一个名字，把匹配到的子串存储在 `OutputVarYear` 里，但是 `OutputVar1` 的值是没有改变（它将保留以前的值，如果有的话）。如果是 [非命名子模式\(See 30.14\)](#) 的 "Year"，它将匹配到的子串存储在 `OutputVar2` 里，而不是 `OutputVar1` 里。

大多数的字符串（像 `abc123`）就能用一般的字符来匹配。而匹配像 `\.*?+[{}|()^$` 这样被保护的字符就要在其前面加是一个反斜杠。比如：`\.` 代表一个点，`\\"` 代表一个反斜杠。转义符可避免使用 `\Q...\E`。比如 `\QLiteral Text\E`。

在正则表达式中，一些特殊的字符，如制表符和换行符就用重音符号 (`) 或反斜杠 (\) 来表示。例如：`t is the same as \t。

如学习正则表达式的基础（或者重新记忆一下正则表达式的语法），请看 [RegEx Quick Reference\(See 30.14\)](#)。

AutoHotKey 所使用的正则表达式是来自于 www.pcre.org 的兼容 Perl 语言的正则表达式。

相关命令

[RegExReplace\(\)](#)(See 18.13), [RegEx Quick Reference](#)(See 30.14), [InStr\(\)](#)(See 10.),
[IfInString](#)(See 27.3), [StringGetPos](#)(See 27.14), [SubStr\(\)](#)(See 10.), [SetTitleMatchMode](#)
[RegEx](#)(See 28.8), [Global matching and Grep \(forum link\)](#)

Common sources of text data: [FileRead](#)(See 16.19), [UrlDownloadToFile](#)(See 22.24),
[Clipboard](#)(See 30.6), [GUI Edit controls](#)(See 19.4)

示例

```
FoundPos := RegExMatch("xxxabc123xyz", "abc.*xyz") ; 返回值 4，它就是匹配到的位置。
FoundPos := RegExMatch("abc123123", "123$") ; 返回值 7，因为 $ 要求从靠近最后的字符处开始匹配。
FoundPos := RegExMatch("abc123", "i)^ABC") ; 返回值 1，因为是要求从第一个字符处开始匹配，而且不区分大小写。
FoundPos := RegExMatch("abcXYZ123", "abc(.*)123", SubPat) ; 返回 1 并且 SubPat1 的值是 "XYZ"。
FoundPos := RegExMatch("abc123abc456", "abc\d+", "", 2) ; 返回值 7，由于它是从第二个字符开始匹配的。

; 更多正则表达式的例子，请看 RegEx Quick Reference\(See 30.14\).
```

18.14 RegisterCallback

Creates a machine-code address that when called, redirects the call to a [function](#)(See 10.) in the script.

```
Address := RegisterCallback("FunctionName" [, Options = "", ParamCount = FormalCount, EventInfo = Address])
```

参数

Address	Upon success, RegisterCallback() returns a numeric address that may be called by DIIICall() (See 18.3) or anything else capable of calling a machine-code function. Upon failure, it returns an empty string. Failure occurs when <i>FunctionName</i> : 1) does not exist; 2) accepts too many or too few parameters according to <i>ParamCount</i> ; or 3) accepts any ByRef parameters (See 10.).
FunctionName	A function (See 10.)'s name, which must be enclosed in quotes if it is a literal string. This function is called automatically whenever <i>Address</i> is called. The function also receives the parameters that were passed to <i>Address</i> .
Options	<p>Specify zero or more of the following words. Separate each option from the next with a space (e.g. "C Fast").</p> <p>Fast or F: Avoids starting a new thread(See 30.19) each time <i>FunctionName</i> is called. Although this performs better, it must be avoided whenever the thread from which <i>Address</i> is called varies (e.g. when the callback is triggered by an incoming message). This is because <i>FunctionName</i> will be able to change global settings such as ErrorLevel(See 30.8), A_LastError(See 9.), and the last-found window(See 30.2) for whichever thread happens to be running at the time it is called. For more information, see Remarks.</p> <p>CDecl or C : Makes <i>Address</i> conform to the "C" calling convention. This is typically omitted because the standard calling convention is much more common for callbacks.</p>
ParamCount	The number of parameters that <i>Address</i> 's caller will pass to it. If entirely omitted, it defaults to the number of mandatory parameters in the definition (See 10.) of <i>FunctionName</i> . In either case, ensure that the caller passes exactly this number of parameters.
EventInfo	An integer between 0 and 4294967295 that <i>FunctionName</i> will see in A_EventInfo (See 9.) whenever it is called via this <i>Address</i> . This is useful when <i>FunctionName</i> is called by more than one <i>Address</i> . If omitted, it defaults to <i>Address</i> . Note: Unlike other global settings, the current thread (See 30.19)'s A_EventInfo is not disturbed by the fast mode .

The Callback Function's Parameters

A [function](#)(See 10.) assigned to a callback address may accept up to 31 parameters. [Optional parameters](#)(See 10.) are permitted, which is useful when the function is called by more than one caller.

All incoming parameters are integers between 0 and 4294967295. If an incoming parameter is intended to be a signed integer, any negative numbers can be revealed by following this example:

```
if wParam > 0xFFFFFFFF
    wParam := -(~wParam) - 1
```

If an incoming parameter is intended by its caller to be a string, that string may be retrieved by copying it into a variable. For example:

```
VarSetCapacity(MyString, DllCall("lstrlen", UInt, MyParameter)) ; Unnecessary if
MyString is already big enough.

DllCall("lstrcpy", Str, MyString, UInt, MyParameter) ; Copy the string into the script's
MyString variable.

VarSetCapacity(MyString, -1) ; Update the variable's internally-stored length to
reflect its new contents.
```

If an incoming parameter is the address of a structure, the individual members may be extracted by following the steps at [DllCall structures](#)(See 18.3).

What the Function Should *Return*

If the function uses [Return](#)(See 17.19) without any parameters, or it specifies a blank value such as "" (or it never uses Return at all), 0 is returned to *Address*'s caller. Otherwise, the function should return an integer between -2147483648 and 4294967295, which is then returned to *Address*'s caller.

Fast vs. Slow

The default/slow mode causes the function to start off fresh with the default values for settings such as [SendMode](#)(See 20.13) and [DetectHiddenWindows](#)(See 28.5). These defaults can be changed in the [auto-execute section](#)(See 8.).

By contrast, the **fast mode** inherits global settings from whichever [thread](#)(See 30.19) happens to be running at the time the function is called. Furthermore, any changes the function makes to global settings (including [ErrorLevel](#)(See 30.8) and the [last-found window](#)(See 30.2)) will go into effect for the [current thread](#)(See 30.19). Consequently, the fast mode should be used only when it is known exactly which thread(s) the function will be called from.

To avoid being interrupted by itself (or any other thread), a callback may use [Critical](#)(See 22.7) as its first line. However, this is not completely effective when the function is called indirectly via the arrival of a message less than 0x312 (increasing Critical's [interval](#)(See 22.7) may help).

Furthermore, [Critical](#)(See 22.7) does not prevent the function from doing something that might indirectly result in a call to itself, such as calling [SendMessage](#)(See 28.1.12) or [DII Call](#)(See 18.3).

Memory

Each use of RegisterCallback() allocates a small amount of memory (32 bytes plus system overhead). Since the OS frees this memory automatically when the script exits, any script that allocates a small, *fixed* number of callbacks does not have to explicitly free the memory. By contrast, a script that calls RegisterCallback() an indefinite/unlimited number of times should explicitly call the following on any unused callbacks: *DII Call("GlobalFree", UInt, Address)*

相关命令

[DII Call](#)(See 18.3), [OnMessage](#)()(See 18.11), [OnExit](#)(See 17.17), [OnClipboardChange](#)(See 30.6), [Sort's callback](#)(See 27.12), [Critical](#)(See 22.7), [Post/SendMessage](#)(See 28.1.12), [Functions](#)(See 10.), [List of Windows Messages](#)(See 30.17), [Threads](#)(See 30.19)

示例

```
; Example: The following is a working script that displays a summary of all top-level windows.

; For performance and memory conservation, call RegisterCallback() only once for a given callback:

if not EnumAddress ; Fast-mode is okay because it will be called only from this thread:
    EnumAddress := RegisterCallback("EnumWindowsProc", "Fast")

DetectHiddenWindows On ; Due to fast-mode, this setting will go into effect for the callback too.

; Pass control to EnumWindows(), which calls the callback repeatedly:

DII Call("EnumWindows", UInt, EnumAddress, UInt, 0)

MsgBox %Output% ; Display the information accumulated by the callback.

EnumWindowsProc(hwnd, lParam)
{

```

```

global Output

WinGetTitle, title, ahk_id %hwnd%
WinGetClass, class, ahk_id %hwnd%
if title
    Output .= "HWND: " . hwnd . "`tTitle: " . title . "`tClass: " . class . "`n"
return true ; Tell EnumWindows() to continue until all windows have been
enumerated.
}

```

; Example: The following is a working script that demonstrates how to subclass a GUI window by

; redirecting its WindowProc to a new WindowProc in the script. In this case, the background
; color of a text control is changed to a custom color.

TextBackgroundColor := 0xFFBBBB ; A custom color in BGR format.

TextBackgroundBrush := DllCall("CreateSolidBrush", UInt, TextBackgroundColor)

Gui, Add, Text, HwndMyTextHwnd, Here is some text that is given`na custom background color.

Gui +LastFound

GuiHwnd := WinExist()

WindowProcNew := **RegisterCallback**("WindowProc", "" ; Specifies "" to avoid fast-mode for subclassing.

, 4, MyTextHwnd) ; Must specify exact ParamCount when EventInfo parameter is present.

WindowProcOld := DllCall("SetWindowLong", UInt, GuiHwnd, Int, -4 ; -4 is GWL_WNDPROC

, Int, WindowProcNew, UInt) ; Return value must be set to UInt vs. Int.

Gui Show

```

return

WindowProc(hwnd, uMsg, wParam, lParam)
{
    Critical

    global TextBackgroundColor, TextBackgroundBrush, WindowProcOld

    if (uMsg = 0x138 && lParam = A_EventInfo) ; 0x138 is WM_CTLCOLORSTATIC.

    {
        DllCall("SetBkColor", UInt, wParam, UInt, TextBackgroundColor)

        return TextBackgroundBrush ; Return the HBRUSH to notify the OS that we
altered the HDC.

    }

    ; Otherwise (since above didn't return), pass all unhandled events to the original
WindowProc.

    return DllCall("CallWindowProcA", UInt, WindowProcOld, UInt, hwnd, UInt, uMsg, UInt,
wParam, UInt, lParam)
}
}

GuiClose:

ExitApp

```

18.15 StrLen

StrLen(String): 返回 *String* 的长度。如果 *String* 是 [ClipboardAll](#) 先前分配的变量，将返回它的总大小。相关命令: [StringLen](#)。

18.16 SubStr

SubStr(String, StartingPos [, Length]) [v1.0.46+]: 在 *String* 字符串中从 *StartingPos* 起始点开始向右复制不超过 *Length* 长度的字符的子字符串(如果参数 *Length* 省略，就默认为“所有字符”)。对于 *StartingPos*，指定 1 则从首个字符开始，2 则从第二个字符开始，以此类推(如果 *StartingPos* 超出了 *String* 的长度，将返回一个空字符串)。如果 *StartingPos* 小于 1，将被视为从字符串末尾开始的位置。例如，0 提取最后一个字符，-1 提取最后两个字符(但是如果 *StartingPos* 超过了字符串的左侧末尾，提取又会从左侧首个字符开始)。*Length* 是要获取的字符的最大数目(每当字符串剩余部分太短的时候，获取的长度会比最大数目少一些)。指定一个负的 *Length* 从而在返回的字符串的末尾省略这么多个字符(如果

省略了全部或更多字符，将返回一个空字符串）。相关链接: [RegExMatch\(\)](#), [StringMid](#), [StringLeft/Right](#), [StringTrimLeft/Right](#).

18.17 VarSetCapacity

Enlarges a variable's holding capacity or frees its memory. Normally, this is necessary only for unusual circumstances such as [DIICall](#)(See 18.3).

`GrantedCapacity := VarSetCapacity(UnquotedVarName [, RequestedCapacity, FillByte])`

参数

GrantedCapacity	The length of string that Var can now hold, which will be greater or equal to <i>RequestedCapacity</i> . If <i>VarName</i> is not a valid variable name (such as a literal string or number), 0 is returned. If the system has insufficient memory to make the change (very rare), an error dialog will be displayed and the current thread (See 30.19) will exit.
UnquotedVarName	The name of the variable (<i>not in quotes</i>). For example: <code>VarSetCapacity(MyVar, 1000)</code> . This can also be a dynamic variable such as <code>Array%i%</code> or a function's ByRef parameter (See 10.).
RequestedCapacity	<p>If omitted, the variable's current capacity will be returned and its contents will not be altered. Otherwise, anything currently in the variable is lost (the variable becomes blank).</p> <p>Specify for <i>RequestedCapacity</i> the length of string that the variable should be able to hold after the adjustment. This length does not include the internal zero terminator. For example, specifying 1 would allow the variable to hold up to one character in addition to its internal terminator. Note: the variable will auto-expand if the script assigns it a larger value later.</p> <p>Since this function is often called simply to ensure the variable has a certain minimum capacity, for performance reasons, it shrinks the variable only when <i>RequestedCapacity</i> is 0. In other words, if the variable's capacity is already greater than <i>RequestedCapacity</i>, it will not be reduced (but the variable will still made blank for consistency).</p> <p>Therefore, to explicitly shrink a variable, first free its memory with <code>VarSetCapacity(Var, 0)</code> and then use <code>VarSetCapacity(Var, NewCapacity)</code> -- or simply let it auto-expand from zero as needed.</p> <p>For performance reasons, freeing a variable whose previous capacity was between 1 and 63 might have no effect because its memory is of a permanent type. In this case, the current capacity will be returned rather than 0.</p>

	<p>For performance reasons, the memory of a variable whose capacity is under 4096 is not freed by storing an empty string in it (e.g. <code>Var := ""</code>). However, <code>VarSetCapacity(Var, 0)</code> does free it.</p> <p>In v1.0.44.03+, specify -1 for <code>RequestedCapacity</code> to update the variable's internally-stored length to the length of its current contents. This is useful in cases where the variable has been altered indirectly, such as by passing its address(See 9.) via DII Call()(See 18.3). In this mode, <code>VarSetCapacity()</code> returns the length rather than the capacity.</p>
FillByte	<p>This parameter is normally omitted, in which case the memory of the target variable is not initialized (instead, the variable is simply made blank as described above). Otherwise, specify a number between 0 and 255. Each byte in the target variable's memory area (its current capacity) is set to that number. Zero is by far the most common value, which is useful in cases where the variable will hold raw binary data such as a DII Call structure(See 18.3).</p>

注意

In addition to its uses described at [DII Call](#)(See 18.3), this function can also be used to enhance performance when building a string by means of gradual concatenation. This is because multiple automatic resizings can be avoided when you have some idea of what the string's final length will be. In such a case, `RequestedCapacity` need not be accurate: if the capacity is too small, performance is still improved and the variable will begin auto-expanding when the capacity has been exhausted. If the capacity is too large, some of the memory is wasted, but only temporarily because all the memory can be freed after the operation by means of `VarSetCapacity(Var, 0)` or `Var := ""`.

[#MaxMem](#)(See 29.15) restricts only the automatic expansion that a variable does on its own. It does not affect `VarSetCapacity`(See 18.17).

相关命令

[DII Call](#)(See 18.3), [#MaxMem](#)(See 29.15)

示例

```
VarSetCapacity(MyVar, 10240000) ; ~10 MB
Loop
{
    ...
    MyVar = %MyVar%StringToConcatenate%
```

```

...
}
```

18.18 WinActive

WinActive([*WinTitle*, *WinText*, *ExcludeTitle*, *ExcludeText*]): 如果激活的窗口匹配指定的条件，则返回窗口的唯一 ID (HWND)。如果不匹配，函数返回 0。因为所有非零的值都视为 "true"，所以每当匹配了 *WinTitle* 的窗口被激活时语句 `if WinActive("WinTitle")` 都是 true。最后，*WinTitle* 支持 `ahk_id`, `ahk_class` 以及其他特殊字符串。关于这些和窗口激活方面的其他信息，详见 [IfWinActive](#)。

18.19 WinExist

WinExist([*WinTitle*, *WinText*, *ExcludeTitle*, *ExcludeText*]): 返回以十六进制整数表示的首个匹配窗口的唯一 ID (HWND)(如果没有就是 0)。因为所有非零的值都视为 "true"，所以每当匹配了 *WinTitle* 的窗口存在时语句 `if WinExist("WinTitle")` 都是 true。最后，*WinTitle* 支持 `ahk_id`, `ahk_class` 以及其他特殊字符串。关于这些和窗口查找方面的其他信息，详见 [IfWinExist](#)。

18.20 Math

注意：如果传入的任何参数是非数值的，那么数学函数通常会返回一个空值(空字符串)。

18.20.1 Abs

Abs(*Number*): 返回 *Number* 的绝对值。返回值的类型和 *Number* 一致(整数或浮点数)。

18.20.2 Ceil

Ceil(*Number*): 返回 *Number* 向上取整数(没有任何的 .00 后缀)的值。例如，Ceil(1.2) 是 2，Ceil(-1.2) 是 -1。

18.20.3 Exp

Exp(*N*): 返回 e (大约为 2.71828182845905) 的 *N* 次幂。*N* 可以是负数也可以包含小数点。要抬升除了 e 以外的数到某个幂，请用 [** 运算符](#)。

18.20.4 Floor

Floor(*Number*): 返回 *Number* 向下取整数(没有任何的 .00 后缀)的值。比如 Floor(1.2) 是 1，Floor(-1.2) 是 -2。

18.20.5 Log

Log(Number): 返回 *Number* 的对数(以 10 为底)。结果被格式化为浮点数。如果 *Number* 是负数, 将返回一个空字符串。

18. 20. 6 Ln

Ln(Number): 返回 *Number* 的自然对数(以 e 为底)。结果被格式化为浮点数。如果 *Number* 是负数, 将返回一个空字符串。

18. 20. 7 Mod

;例 3: 检测热键的单击、双击和三次按击。;这允许热键可以按你按键的次数来执行不同的操作: #c::if winc_presses > 0 ;

```
SetTimer 已经启动, 所以我们记录按键。{ winc_presses += 1
return} ;否则, 这是新一系列按键的首次按键。将计数设为 1 并启动定时器: winc_presses = 1SetTimer, KeyWinC, 400 ;在 400 毫秒内等待更多的按键。returnKeyWinC:SetTimer, KeyWinC, offif winc_presses = 1 ;该键已按过一次。{ Run, m:\ ;打开一个文件夹。 } else if winc_presses = 2 ;该键已按过两次。{ Run,
m:\multimedia ;打开一个不同的文件夹。 } else if winc_presses > 2 { MsgBox, 检测到三次或更多次点击。 };不论上面哪个动作被触发, 将计数复位以备下一系列的按键: winc_presses = 0return18. 20. 8
```

Round

Round(Number [, N]): 如果 *N* 为 0 或者省略 *N*, *Number* 将四舍五入为整数。如果 *N* 是正数, *Number* 取 *N* 个小数位。如果 *N* 是负数, 在 *Number* 的小数点的左边取 *N* 位来四舍五入。例如: Round(345, -1) 结果为 350, Round(345, -2) 结果为 300。与 [Transform Round](#) 不同, 每当 *N* 小于 1 或者省略时, 结果没有 .000 后缀。在 v1.0.44.01 及之后的版本中, 一个大于零的 *N* 值会确切地显示 *N* 个小数位, 而不会服从 [SetFormat](#)。要避免这种情况的话, 对 Round() 的返回值再执行另一个数学操作; 例如: Round(3.333, 1)+0。

18. 20. 9 Sqrt

Sqrt(Number): 返回 *Number* 的平方根。结果被格式化为浮点数。如果 *Number* 是负数，函数将得出空值(空字符串)。

18. 20. 10 Sin/Cos/Tan

Sin(Number) | Cos(Number) | Tan(Number): 返回 *Number* 的正弦|余弦|正切三角函数值。*Number* 必须用弧度表示(详见下面)。

18. 20. 11 ASin/ACos/ATan

ASin(Number): 返回以弧度表示的反正弦(其正弦是 *Number*)。如果 *Number* 小于 -1 或者大于 1，则函数得到一个空值(空字符串)。

ACos(Number): 返回以弧度表示的反余弦(其余弦是 *Number*)。如果 *Number* 小于 -1 或者大于 1，则函数得到一个空值(空字符串)。

ATan(Number): 返回以弧度表示的反正切(其正切是 *Number*)。

注意: 要将弧度转换成角度, 将其乘以 $180/\pi$ (大约为 57.29578)。要将角度转换成弧度, 将其乘以 $\pi/180$ (大约为 0.01745329252)。 π (大约为 3.141592653589793)的值是 1 的反正切乘以 4。

其他函数

Titan 的命令函数: 为每个有 OutputVar 的 AutoHotKey 命令提供一个可调用的函数。可以通过 #Include 将这个库包含在任何一个脚本里。

翻译: hsudatalks 修正: 天堂之门 menk33@163.com 2008 年 11 月 10 日

19. GUI, MsgBox, InputBox & 其它对话框

19.1 FileSelectFile

显示一个允许用户来打开或保存文件的标准对话框。

`FileSelectFile,OutputVar[,Options,RootDir\filename,Prompt,Filter]`

参数

OutputVar	用来保存用户选择的文件名的变量名称。如果用户取消了对话框(即不想选择一个文件), 该变量为空。
Options	如果省略此项, 默认值为 0, 意味着以下选项都不生效。 M: 多选。指定字母 M 表示允许用户通过按住 Shift、Control 键或者其它方式来选择多个文件。M 后面可以跟一个下面提到的数字(例如, M 和 M1 都是有效的)。要

	<p>提取单个文件，请看本页底部的例子。</p> <p>S: 保存按钮。指定字母 S 使对话框上总是包含保存按钮而不是打开按钮。S 后面可以跟一个下面提到的数字(或几个数的和。例如，S 和 S24 都是有效的)。</p> <p>即使没有 M 和 S，下列数字也可以被使用。要同时使多个选项生效，只要把数字加起来。例如，要使用 8 和 16，指定数字 24。</p> <ul style="list-style-type: none"> 1: 文件必须存在 2: 路径必须存在 8: 提示创建新文件 16: 提示覆盖文件 <p>32 [v1.0.43.09+]: 选择的快捷方式(.lnk 文件)是它本身而不是它所指向的目标。此选项也防止了选择一个文件夹快捷方式会跳转到那个文件夹。</p> <p>如果用“提示覆盖文件”选项并且不用“提示创建新文件”选项的话，对话框将包含保存按钮而不是打开按钮。这个问题是 Windows 的毛病。</p>
RootDir\Filename	<p>如果存在，此参数可包含下列的一项或全部：</p> <p>RootDir: 根(起始)目录，如果没有使用完全路径的话，假设为 %A_WorkingDir%(See 9.) 里的子文件夹。如果省略此参数或留空，起始目录会是默认值，而且随操作系统版本而有所不同(在 WinNT/2k/XP 及之后的版本中，它通常是用户上一次使用 <code>FileSelectFile</code> 时选择的目录)。在 v1.0.43.10 及之后的版本中，支持使用 CLSID(Windows 类标识符)(See 30.7)比如 <code>::{20d04fe0-3aea-1069-a2d8-08002b30309d}</code>(即我的电脑)，这时在 <code>CLSID</code> 后面的任何子目录路径都必须以反斜线结尾(否则，最后一个反斜杠之后的字符串将被认为是默认的文件名，请看下面)。</p> <p>Filename: 一开始显示在对话框里的编辑框中的默认文件名。只有单独的文件名(不带路径)才会被显示。要确保对话框正确地显示，请不要提交非法的字符(比如 /< :)。</p> <p>示例：</p> <pre>C:\My Pictures\Default Image Name.gif ; RootDir 和 Filename 都提供了。 C:\My Pictures ;只提供了 RootDir。 My Pictures ;只提供了 RootDir，它相对于当前的工作目录。 My File ;只提供了 Filename(但如果存在"My File"文件夹，则会被认为是 RootDir)。</pre>
Prompt	显示在窗口标题栏指示用户做什么的文本。如果省略或留空，它将默认为"Select File - %A_SCRIPTNAME%" (即当前脚本的名称)。
Filter	指示哪些类型的文件会被对话框显示。 例如：Documents (*.txt)

例如: `Audio (*.wav; *.mp2; *.mp3)`

如果省略, 过滤器默认是 `All Files (*.*)`。另外在对话框的“文件类型”下拉列表中的 `Text Documents (*.txt)` 选项也是可用的。

否则, 过滤器将使用此参数指示的字符串, 不过在对话框的“文件类型”下拉列表中仍然提供 `All Files (*.*)` 的选项。要在过滤器中包含多个文件扩展名, 像上面的例子中那样用分号将它们隔开。

ErrorLevel

如果用户没有选择文件(比如按了取消按钮)就关掉了对话框, [ErrorLevel\(See 30.8\)](#) 设为 1。如果系统拒绝显示对话框时(罕见)也将设为 1。之外设为 0。

注意

如果用户什么也没选(例如按了取消), `OutputVar` 为空。

如果没有使用多选, `OutputVar` 的值将设为用户选择的单个文件的完整路径和名称。

如果使用了 `M` 选项(多选), `OutputVar` 的值将设为一个列表, 其中除了最后一个外, 每个后面都跟了一个换行符(`n)。列表中的第一行是所有被选择的文件的路径(只有在此路径是根目录比如 C:\ 时才以反斜线结尾)。其它的条目是被选择的文件名(不带路径)。例如:

`C:\My Documents\New Folder` [这是下面所有文件所在的路径]
`test1.txt` [这些是单独的文件名: 没有路径]

`test2.txt`

.....等。

(本页底部的例子示范了如何一个个提取文件。)

在使用多选时, 被选择的文件名的总长度被限制在 64 KB。不过通常这足以容纳几千个文件了, 如果超出此限制的话, `OutputVar` 将为空。

GUI 窗口可以用 [Gui +OwnDialogs\(See 19.3\)](#) 来显示一个 modal(模态)文件选择对话框。模态对话框可以阻止用户在关掉对话框之前与 GUI 窗口的交互。

已知限制: 在文件选择对话框显示期间运行的[定时器\(See 17.21\)](#)会延迟用户在对话框中点击的效果直到定时器结束。要绕弯解决此限制, 应避免使用那些子程序会运行很长时间才结束的定时器, 或者在使用对话框时禁用所有的定时器:

[Thread\(See 22.22\)](#), NoTimers

`FileSelectFile`, `OutputVar`

`Thread, NoTimers, false`

作废的选项: 在 v1.0.25.06 及之后的版本中, 多选选项“4”已作废。可是为了与以前的脚本兼容, 这个选项仍然可用。具体是, 如果用户只选择了一个文件, `OutputVar` 会包括它的完整路径和文件名再加上一个换行符(`n)。如果用户选择了多个文件, 除了最后一行还是以换行符(`n)结尾外, 其它的格式都和上面描述的 `M` 选项一样。

相关命令

[FileSelectFolder](#)(See 16.24), [MsgBox](#)(See 19.11), [InputBox](#)(See 19.10), [ToolTip](#)(See 19.15), [GUI](#)(See 19.3), [CLSID List](#)(See 30.7), [parsing loop](#)(See 17.14), [SplitPath](#)(See 16.34)

操作系统也提供了标准的对话框来提示用户选择字体、颜色或图标。这些对话框可以通过 [DIICall\(\)](#)(See 18.3) 命令来显示，请看 www.autohotkey.com/forum/topic17230.html。

示例

```
FileSelectFile, SelectedFile, 3, , 打开一个文件 (*.txt; *.doc)
```

```
if SelectedFile =
    MsgBox, 用户什么也没有选择。
else
    MsgBox, 用户选了下列文件:`n%SelectedFile%
```

[CLSID](#)(See 30.7) 示例：

```
FileSelectFile, OutputVar,, ::{645ff040-5081-101b-9f08-00aa002f954e} ;回收站。
```

;多选示例：

FileSelectFile, files, M3 ; M3 = 选择多个存在的文件。

```
if files =
{
    MsgBox, 用户按了取消。
    return
}
Loop, parse, files, `n
{
    if a_index = 1
        MsgBox, 选择的文件都包含在 %A_LoopField%。
```

else

{

```

MsgBox, 4,, 下一个文件是 %A_LoopField%。继续?

IfMsgBox, No, break

}

}

return

```

翻译：单菜子 修正：天堂之门 menk33@163.com 2008年11月8日

19.2 FileSelectFolder

显示允许用户选择文件夹的标准对话框。

`FileSelectFolder, OutputVar [, StartingFolder, Options, Prompt]`

参数

OutputVar	<p>用来保存用户所选择的文件夹的变量名。如果用户取消了对话框，则该变量将被置为空（比如不想选择文件夹）。若用户选择了根目录（如 C:\），<i>OutputVar</i> 会以反斜杠结尾。如果不想要这个反斜杠，可用下面的方法去掉它：</p> <pre> FileSelectFolder, Folder Folder := RegExReplace(Folder, "\\\$") ; 若以反斜杠结尾，则去掉它。 </pre>
StartingFolder	<p>若为空或省略，对话框选择的起始目录为用户的我的文档文件夹（或者可能是我的电脑）。CLSID 文件夹(See 30.7)，比如 ::{20d04fe0-3aea-1069-a2d8-08002b30309d}（也就是我的电脑）可以作为特殊文件被指定为起始目录。</p> <p>另外，该参数最常见的用法是一个星号通配符后紧跟着作为起始位置的驱动器或文件夹的绝对路径。比如 *C:\ 将把 C 盘作为起始选择位置。与此类似，*C:\My Folder 将把该文件夹作为起始选择位置。</p> <p>星号表示允许用户从起始目录向上导航（接近根目录）。如果没有星号，用户只能选择 <i>StartingFolder</i> 中的文件夹（或者 <i>StartingFolder</i> 本身）。省略星号的一个好处是用户不必点击目录前的加号，<i>StartingFolder</i> 一开始即以树状目录形式显示出来。</p> <p>若存在星号，向上导航可被限制到除桌面外的某个文件夹。在星号前加上最上级文件夹的绝对路径和一个空格或制表符，即可达到此目的。在下面的例子中，不允许用户导航到高于 C:\My Folder 的目录（但起始目录为 C:\My Folder\Projects）：</p> <p><i>C:\My Folder *C:\My Folder\Projects</i></p>
Options	<p>以下数字之一：</p> <p>0: 以下所有选项无效（Windows 2000 除外，它可能始终显示“新建文件夹”按钮）。</p>

	<p>1 (默认)：提供允许用户新建文件夹的按钮。然而，该按钮在 Windows 95/98/NT 中不会出现。</p> <p>上述数字加 2 提供允许用户输入文件夹名的编辑框。比如该参数为数值 3，则同时提供一个编辑框和一个“新建文件夹”按钮。</p> <p>上述数字加 4 可以忽略 <code>BIF_NEWDIALOGSTYLE</code> 属性。加 4 确保了 <code>FileSelectFolder</code> 可以在诸如 WinPE 或者 BartPE 之类的预安装环境中也能正常工作。然而，这可能会妨碍“新建文件夹”按钮的出现，至少在 Windows XP 中是这样。 ["4" 需要 1.0.48+ 版本]</p> <p>如果用户在编辑框中输入非法文件夹名，<code>OutputVar</code> 将会被置为在导航树中选择的文件夹名，而不是用户输入的名称，至少在 Windows XP 中是这样。</p> <p>该参数可以是个表达式(See 9.)。</p>
Prompt	显示在窗口中，告诉用户该做什么的文字。如果省略或空白，则默认为“选择文件夹 - %A_SCRIPTNAME%”（也就是当前脚本的名称）。

ErrorLevel

如果用户关闭对话框而未选择文件，则 [ErrorLevel\(See 30.8\)](#) 被置为 **1**（比如按了取消按钮）。如果系统拒绝显示对话框（罕见），该变量也会被置为 **1**。否则其值为 **0**。

注意

在 GUI 窗口中，可以用 [Gui +OwnDialogs\(See 19.3\)](#) 来显示一个选择文件夹的模式对话框。在模式对话框关闭之前，禁止用户与 GUI 窗口进行交互操作。

已知的限制：在显示 `FileSelectFolder` 对话框时启动的[定时器\(See 17.21\)](#)，将会使用户在对话框中的点击延迟生效，直到该定时器程序执行完毕。为避免这一问题，应避免使用子程序耗时很长的定时器，或者在显示对话框时禁止所有的定时器：

```
Thread(See 22.22), NoTimers
```

```
FileSelectFolder, OutputVar,, 3
```

```
Thread, NoTimers, false
```

相关命令

[FileSelectFile\(See 16.23\)](#), [MsgBox\(See 19.11\)](#), [InputBox\(See 19.10\)](#), [ToolTip\(See 19.15\)](#), [GUI\(See 19.3\)](#), [CLSID List\(See 30.7\)](#), [FileCopyDir\(See 16.6\)](#), [FileMoveDir\(See 16.17\)](#), [SplitPath\(See 16.34\)](#)

操作系统还提供提示用户选择字体、颜色或图标的标准对话框。这些对话框可以通过 [DIIICall\(\)\(See 18.3\)](#) 来显示，示例见 www.autohotkey.com/forum/topic17230.html。

示例

```
FileSelectFolder, OutputVar, , 3

if OutputVar =
    MsgBox, 您未选择文件夹。
else
    MsgBox, 您选择了文件夹 "%OutputVar%"。
```

[CLSID\(See 30.7\)](#) 示例：

```
FileSelectFolder, OutputVar, ::{20d04fe0-3aea-1069-a2d8-08002b30309d} ; 我的电脑。
```

翻译：甲壳虫<jdchenjian@gmail.com>

19.3 Gui

该命令用来创建和管理一个窗体及窗体之上的控件。例如创建一个可以让用户输入数据的窗体或其他自定义界面。

Gui,子命令 [,参数 2, 参数 3, 参数 4]

- [Add](#) 添加：创建如文本、按钮、复选框等控件。
- [Show](#) 显示：显示一个窗体。同时可以最小化最大化或移动窗体。
- [Submit](#) 提交：保存用户输入的数据，同时可以选择隐藏窗体。
- [Cancel\(or Hide\)](#)取消或隐藏：隐藏窗体。
- [Destroy](#) 销毁：销毁窗体。
- [Font](#) 字体：设置随后创建控件中字体的大小、样式、以及颜色。
- [Color](#) 颜色：设置窗体、控件的背景色。
- [Margin](#) 边距：在没有明确指定控件的大小位置时，设置控件之间的默认间距及空隙。
- [Menu\(See 19.3\)](#) 菜单：添加、删除菜单。
- [Flash\(See 19.3\)](#) 闪烁：对窗体及其在任务栏的按钮进行闪烁强调。
- [Default\(See 19.3\)](#) 默认窗体：改变当前线程操作的窗体编号。
- 窗体样式和选项
- 控件大小位置的布局设计([See 19.3](#))
- 最小化、最大化、恢复窗口
- 保存和响应用户的输入：参考 [variables](#) 和 [g-labels](#)
- 控件的样式和选项

- 窗体事件: 关闭窗体 | 取消窗体 | 调整大小 | 窗体菜单 | 拖拽文件
- 创建多个界面
- 界面事件、线程、子过程
- 其他操作: 键盘导航 | 窗体外观 | 概述
- 示例: 创建窗体界面以及控件的示例

在窗体上增加一个控件 (首先必须创建一个界面窗体)

ControlType 参数可以是以下的一种:

- [Text](#)(See 19.4) 文字, [Edit](#)(See 19.4) 编辑框, [UpDown](#)(See 19.4) 增减按钮, [Picture](#)(See 19.4) 图片框
- [Button](#)(See 19.4) 按钮, [Checkbox](#)(See 19.4) 复选框, [Radio](#)(See 19.4) 单选框
- [DropDownList](#)(See 19.4), [ComboBox](#)(See 19.4) 组合框
- [ListBox](#)(See 19.4) 列表框, [ListView](#)(See 19.7) 列表视图, [TreeView](#)(See 19.8) 树形控件
- [Hotkey](#)(See 19.4) 热键输入框, [DateTime](#)(See 19.4) 日期控件, [MonthCal](#)(See 19.4) 日历
- [Slider](#)(See 19.4) 滑动条, [Progress](#)(See 19.4) 进度条
- [GroupBox](#)(See 19.4) 分组框, [Tab](#)(See 19.4) 属性页, [StatusBar](#)(See 19.4) 状态栏, [Internet Explorer Control](#)(See 19.4) IE 控件

示例:

```
Gui, Add, Text,, 请输入你的名字:  
Gui, Add, Edit, vName  
Gui, Show
```

如果没有在 *Options* 里明确指定, 窗口一般会以可见, 非最小化, [激活状态](#)(See 28.12)运行。如果参数 *Title* 未设置, 则窗体的标题和先前的窗体一致 (如果没有先前的窗体, 则以脚本的名字作为标题)。

若省略参数 *X*, *Y*, *W*, *H* 后面的数字, 则窗体保持先前窗体的大小和位置。若无先前窗体位置且没有设置 *X*, *Y*, 窗体则自动在屏幕居中。若无先前窗体的尺寸, 则窗体会根据其包含的控件的大小和尺寸, 自动调整窗体大小。

Options 参数可以省略, 或包含以下字符

Wn: 指定 *n* 为窗体客户区 (除去标题栏, [菜单](#)(See 19.3), 和窗体的边框的区域) 的宽度 (以像素为单位)。

Hn: 指定 *n* 为窗体客户区的高度, 以像素为单位。

Xn: 指定 *n* 为窗体在屏幕中的 *x* 坐标, 0 表示为屏幕 最左边第一列可见像素的位置。

Yn: 指定 *n* 为窗体在屏幕中的 *y* 坐标, 0 表示为屏幕 最上边第一行可见像素的位置。

Center: 使窗体在屏幕水平和垂直方向居中显示。

xCenter: 使窗体在屏幕的水平位置居中显示，例如: *Gui, Show, xCenter y0*

yCenter: 使窗体在屏幕的垂直方向居中显示。

AutoSize: 自动根据当前窗体上的控件来调整窗体的大小。这对于调整一个新加入控件的窗体，或控件移动，隐藏后的窗体的大小，是非常有用的。例如: *Gui, Show, AutoSize Center*

Options 中可以同时 包含以下参数:

Minimize: 最小化当前窗体并激活下一窗口

Maximize: 最大化并激活窗体

Restore: 还原，取消最小或最大化。窗体可能最小化后仍处于被取消激活状态。

NoActivate: 取消最大化最小化，并且不激活窗口。

NA: 显示窗体但不激活。如果窗体是最小化状态，可能会提升其 Z-Order 的数序（也就是在按下 alt-tab 时显示的顺序）。如果窗口原来为隐藏状态，他可能会显示在当前仍处于激活状态的窗体之上。

Hide: 隐藏窗体并激活下一窗口。除了允许移动，调整一个隐藏隐藏的窗体，设置窗体的标题外，这与 [Gui Cancel\(See 19.3\)](#) 函数的效果相同。例如: *Gui, Show, Hide x55 y66 w300 h200, New Title*

保存每个控件的内容到与之 [关联的变量\(See 19.3\)](#)，并且隐藏窗体（可以指定 **NoHide** 不隐藏）。对于那些可以产生多个变量的控件，如 [多选列表框\(See 19.4\)](#)，可以利用窗口的数据分隔符([See 19.3](#))来得到多个输出变量。如果之前调用了 [Gui Destroy\(See 19.3\)](#)，而使得窗口不存在，则该命令没有效果。

隐藏当前窗体，并且不保存控件的 [关联变量\(See 19.3\)](#)，如果由于之前调用了 [Gui Destroy\(See 19.3\)](#) 而使窗体不存在，则该命令没有效果。

移除一个存在的窗体和上面的所有控件，并且释放其所占用的内存及系统资源。如果脚本之后又需要显示窗体，则窗体的所有样式，如颜色和字体都会以默认样式显示（就如窗体从未产生过）。如果在脚本结束时没有调用 [Gui Destroy](#)，则所有的窗体会被自动销毁。

为该条语句之后创建的控件设置字体的样式，大小，样式，颜色。例如:

`gui, font, s10, Verdana ; 设置为 Verdana 字体，大小 10 号`

若省略最后的两个参数，则将字体、颜色、大小，设置为系统默认的样式。

FontName 可以是任意字体的名字，如 [字体列表\(See 30.9\)](#)中的一个。如果 **FontName** 参数被忽略，

或是系统中不存在，则设置为之前所设置的字体样式（如果没有，则设置为系统默认的样式）。一些系统缺少所设置的首选字体，这种方式也可以让 窗体在不同的系统上具有相似的字体。例如，以下顺序的命令，**Verdana** 字体比 **Arial** 具有优先权，**Arial** 比 **MS** 字体具有优先权：

```
gui, font,, MS sans serif
gui, font,, Arial
gui, font,, Verdana ; 首选字体
```

如果参数 **Options** 为空，则使用先前的字体属性。另外，你也可以指定下列关键字来设置字体的样式：

C: 后面可以跟颜色名称（参加[颜色名称表\(See 19.12\)](#)）或使用 RGB 值，或这关键字 **Default** 来使用系统默认的颜色（大多数系统默认色为黑色）。例如：cRed, cFFFFAA, cDefault 注意：[Buttons\(See 19.4\)](#) 不受自定义颜色的影响。如果单独创建一些带有颜色类型的控件时，它不会受到当前 C 关键字的影响！如：
Gui, Add, Text, cRed, My Text

S: 设置大小 如：s12 表示字号为 12

W: 设置粗细, W 后面可以跟一个 1 到 1000 之间的数字（400 为标准，700 为粗体）例如：w600

同时在 **Option** 中也可以跟以下这些关键字：**bold**（表示粗 体），**italic**（表示斜体），**strike**（表示删除线），**underline**（表示下划线），**norm**（表示正常粗细，并关闭斜体选项），这些关键字不会改变字体颜色和大 小。如果出现关键字 **norm** 则关闭所有的其他字体样式，如，指定 **Option** 为 **norm italic** 则设置字体为默认样式而忽略 **italic** 斜体属性。

想要设置多种样式的话，只用在每个样式之间加入空格即可，如：cBlue s12 bold

如果脚本创建了[多窗口界面\(See 19.3\)](#)的话，每个窗口都会在他创建控件时，使用自己的“当前字体”。

另外，操作系统为用户提供了标准的系统对话框，可以让用户选择字体，颜色，图标等。这些对话框都可以通过调用 [DIICall\(\)\(See 18.3\)](#) 来创建，更多示例请参看 www.autohotkey.com/forum/topic17230.html

该命令是用来设置窗口或其控件的背景颜色。**WindowColor** 用来设置窗体的背景 颜色。**ControlColor** 用 来设置当前窗口上的控件，或之后创建的控件的颜色（除了少数几个不支持自定义颜色的控件）。

ControlColor 可 以定义 [ListViews\(See 19.7\)](#) 和 [TreeViews\(See 19.8\)](#) 的初始颜色，但之后如果使用 **ControlColor** 来改变他们的颜色则无效，在这 种 情况下，可以使用 **GuiControl +BackgroundFF9977** 来精确 定义并改变他们的颜色！

如果参数 **WindowColor** 和 **ControlColor** 为空，则保留当前颜色；否则可以指定为 16 种 [HTML 颜色名 称\(See 19.12\)](#)中的一个，或是一个 6 位的 RGB 值（前缀 0x 可以忽略），或者指定为关键词 **Default** 设 置为当前系统默认颜色。

例如，选项 **WindowColor** 和 **ControlColor** 的值可以是：Silver, FFFFAA, 0xFFFFAA, Default
如果不使用该命令，则窗口的背景颜色为系统的按钮的颜色，控件的背景颜色为系统默认颜色(一般为白色)。

[菜单\(See 19.3\)](#)颜色和子菜单也可以改变颜色。例： [Menu\(See 22.13\)](#), MyMenuBar, Color, White

如果想使窗体的背景透明（只能在 Windows 2000/XP 或之后的系统中），可以使用 [WinSet TransColor\(See 28.29\)](#) 命令。然而，如果你在使用该命令之前没有调用 "Gui, Color" 为窗体指定一个颜色，则窗口上的按钮也会变得透明。你可以事先通过设置窗体的颜色来解决这个问题。例：

```
Gui, Color, EEA99
Gui +LastFound ; 指定 GUI 窗体为 last found window(See 30.2) .
WinSet, TransColor, EEA99
```

你会发现以上代码可使窗体透明，但仍保留窗体 的边框及标题栏，想要去除他们，可以在调用窗体透明命令之前使用以下命令：

Gui -Caption ; 我们修改了第一个窗体，如果要设置第二个窗体则可以写为 **Gui,2:-Caption**

如想了解该命令的更多介绍，请参见本页最后的 **display (OSD)** 示例

该命令用来设置当窗体上的控件没有指定相应的 **X** 和 **Y** 坐标(See 19.3)时，控件自动排列时距离窗体边框的距离。**X** 为控件距离窗体左右边框的距离，单位为像素。**Y** 是距离窗体上下边框的距离。该命令同样会影响控件自动排列时彼此之间的水平和垂直间距。最后，当显示窗体 [Gui Show\(See 19.3\)](#) 时边距是会被计算到窗体的大 小中去。（当窗体的大小没有明确指定时）

如果没有使用该命令，则当第一个控件在窗体上添加时，则窗体会根据控件的字体 [font\(See 19.3\)](#) 来计算边距。（使用字体高度的 0.75 倍作为上下边距，字体高度的 1.25 倍作为左右边距）

虽然在添加控件的过程中有可能改变边距，但这只影响之后创建的控件，而不是已经存在的。如果 **X** 和 **Y** 都未指定，则保持边距一致。

该命令可以为窗口指定更多的选项，在 **GUI** 后面可以跟一个或多个选项。考虑到效率的原因，最好将所有的选项都写在一行里面，并且在窗体创建之前调用（也就是，在使用任何子命令如，[Gui Add\(See 19.3\)](#) 前调用）

这个命令的效果是可以堆积的，也就是说，他只改变明确设置的选项，并保留其他没有改变的。

在选项前指定一个加号表明增加改选项，指定一个减号表明去除该效果，例如：

```
Gui +Resize -MaximizeBox ; 改变默认窗体 default 的设置
Gui 2:+Resize -MaximizeBox ; 改变第二个窗体的设置
```

Option 可以为以下选项：

AlwaysOnTop: 使窗体置顶，这个和 "[WinSet\(See 28.29\)](#) AlwaysOnTop" 效果相 同。

Border: 在窗体的周围增加一个细边框。该选项不经常用。

Caption (默认具有改选项): 为窗体增加标题栏，并且填充窗体的边距。如果想使用 [WinSet TransColor\(See 28.29\)](#) 命令使窗体透明，请在调用 **TransColor** 命令后再使用该命令去除标题栏。

Delimiter: 当 调用 [Gui Add](#)(See 19.3) , [GuiControl](#)(See 19.5) , [Gui Submit](#)(See 19.3) , [GuiControlGet](#)(See 19.6) 命令后, 当与控件关联的变量中可能包含有多个数据时, 包含在变量里的数据会以 (|) 符号隔开。该命令可以用来改变分隔数据的分隔符 (|) 为其他的符号。在关键词 **Delimiter** 后加一个字符, 来指定该字符为窗体数据的分隔符, 如: *Gui +Delimiter`n* 表示用换行符 (`n) 来分隔变量中的数据。这对于一个[连续部分](#)(See 8.)的数据是非常适合的! 同样, *Gui +Delimiter|* 可以重新使用 (|) 来作为数据分隔符。如果想使用“空格”或“制表符”来作为分隔符, 可以写为: *Gui +DelimiterSpace* 或 *Gui +DelimiterTab* 。一旦数据分隔符改变, 他将影响到在当前窗体上正在运行以及之后运行的[线程](#)(See 30.19)。

Disabled: 使窗口不可用, 阻止用户和控件的交互。该命令常用于包含有其他子窗体的窗体 (参见 [Owner](#)(See 19.3))

Label [v1.0.44.09+]: 为窗体的[特定标签](#)(See 19.3)指定一个自定义的名称。如 *Gui 2:+Label/MyGui* 表示使用 *MyGuiClose* 和 *MyGuiSize* 标签(如果存在)来替代默认的 *2GuiClose* 和 *2GuiSize* 标签。也就是说, 在这些[特殊标签](#)(See 19.3) 中, 默认的字符串“2Gui”被“MyGui”替换。这样可以让多窗体的 GUI 使用一个共同的标签(可以根据脚本内置变量 [A_Gui](#)(See 9.) 来判断是那个窗体调用了该标签)。

LastFound: 设置窗体为 [last found window](#)(See 30.2) (虽然这对于界 面线程(See 19.3)有些多余, 因为他自动设置为 [Last Found](#)), 该命令允许如 [WinSet](#)(See 28.29) 命令对该窗口进行操作, 即使该窗口为隐藏状态(可以不用 [DetectHiddenWindows](#)(See 28.5) 命令来设置)。这对于在窗口还没有显示之前, 改变他的属性是非常有用的。例如:

```
Gui +LastFound
WinSet, TransColor, %CustomColor% 150
Gui Show
```

LastFoundExist [v1.0.43.09+]: *LastFoundExist* 不同于其他选项, 是唯一一个不能和其他选项同时出现的选项。*+LastFoundExist* 和 *+LastFound* 的效果一样, 除了如果窗口不存在则不创建窗口。该选项主要是判断指定窗口是否已经存在! 例如:

```
Gui 2:+LastFoundExist
IfWinExist
  MsgBox 窗体 #2 已经存在
```

MaximizeBox: 使标题栏上的最大化按钮可用, [Resize](#) 也同时包含该效果。

MinimizeBox (默认): 使标题栏上的最小化按钮可用。

MinSize and MaxSize [v1.0.44.13+]: 当用户拖动窗体的边框, 限定窗口的最小尺寸和最大尺寸。在关键字 *MinSize* 或 *MaxSize* 后面没有任何数字, 则使用当前窗体的尺寸作为最小或最大限制(如果当前还没有窗口的尺寸, 则以窗体第一次显示时的大小为限制尺寸!)。也可以在后面加上“宽带 X 高度”例如: *Gui +Resize +MinSize640x480* 以像素为单位指定客户区(去除标题栏, 边框以及窗口菜单)的大小。

如果省略宽度或高度表示不改变高度和宽度, 如 *+MinSize640x* 或 *+MinSizex480*, 此外, 可以使用窗口的当前尺寸, 调用多次 [Min/MaxSize](#) 来指定窗口的大小。例如, *+MinSize +MinSize640x* 表示使用窗口当前的高度, 并限定窗口最小宽度为 640。

如果 和 选项没有使用, 则使用操作系统默认的大小(同样 *Gui -MinSize -MaxSize* 也可以使窗体返回默认大小)。

注意: 窗体必须具有 [+Resize](#)(See 19.3) 选项, 以便用户可以调整窗体的大小。

OwnDialogs: 命令 `Gui +OwnDialogs` 可以指定窗体上的每个线程（例如 `ButtonOK` 子过程）为子窗体，可以使窗体上激活的如 [MsgBox](#)(See 19.11), [InputBox](#)(See 19.10), [FileSelectFile](#)(See 16.23), 和 [FileSelectFolder](#)(See 16.24) 对话框为当前窗体的子窗体。这些对话框都为模式对话框，也就是说用户必须先关闭这些对话框才能和他们的父窗体进行交换。相比之下，[ToolTip](#)(See 19.15), [Progress](#)(See 19.12)，和 [SplashImage](#)(See 19.12) 窗体即使设置了为子窗体，他们仍为非模式，他们仅仅只是显示在其父窗体之上。不管是模式或是非模式，当起对话框销毁 [destroyed](#)(See 19.3) 时，其子窗体会自动销毁。 (See 16.24)

该命令不需要写在代码行的末尾，因为他不会影响到其他的[线程](#)(See 30.19)。但如果一个线程需要显示两个对话框，或是想要取消 `owned` 设置，可以通过命令 `Gui -OwnDialogs` 来关闭子窗体设置。

如果在 `Gui` 后面没有指定一个数字，如使用命令 `Gui +OwnDialogs` 而不是 `Gui 2:+OwnDialogs`，则对[默认窗体](#)(See 19.3)进行设置！

Owner: 使用 `+owner` 选项可以设置该窗体为其他窗体的子窗体（一旦窗体创建，选项 `-owner` 则没有效果）。一个子窗体不会在任务栏中显示按钮，并且子窗体永远显示在其父窗体之上。当父窗体销毁时，子窗体自动销毁。使用选项 `+Owner` 时，确保其父窗体已经创建（也就是说要在命令 [Gui Add](#)(See 19.3) 之后使用该选项）。以下有两种使用 `+owner` 选项的示例：

```
gui, 2:+owner1 ; 设置窗体#2 为窗体#1 的子窗体。
gui, 2:+owner ; 设置窗体#2 为脚本主窗体 script's main window(See 22.13) 的子窗体,
并且不显示任务栏按钮。
```

如果在子窗体显示时需要防止用户和父窗体进行交互，可以通过 `Gui +Disabled` 命令来使父窗体无效。当子窗体取消或销毁前，调用 `Gui -Disabled` 命令，则当子窗体销毁后，父窗体自动恢复为有效。

Resize: 指定该选项则允许用户改变窗体的大小，并且使标题栏上的最大化按钮可用。如果想避免，可以指定 `+Resize -MaximizeBox` 来取消最大化按钮。

SysMenu (默认): 指定 `-SysMenu` 表示删除系统菜单和点击窗体左上角的图片所弹出的菜单。同时也删除标题栏上的最小化最大化按钮。

Theme: 指定 `-Theme`，则之后所有创建的控件将以 WindowsXP 经典主题的样式显示。如果想让创建的控件遵循当前的系统主题，则指定 `+Theme` 即可。注意：该选项对 WindowsXP 之前的系统无效。当前 XP 的主题为经典样式时，该选项无效。对于个别的控件，也可以在控件的 `options` 中指定 `+Theme` 或 `-Theme` 来改变其样式。

ToolWindow: 使标题栏变成细长状，同时不显示任务栏按钮。

(未命名样式): 在表示样式的 10 进制或 16 进制的数字 [style number](#)(See 30.18) 前，增加加号或减号，表示添加或删除此样式。

(未命名扩展样式): 在表示样式的 10 进制或 16 进制数字前加字母 E 表示扩展样式，可以通过加号和减号添加和删除该扩展样式。例如+`E0x40000` 表示增加 `WS_EX_APPWINDOW` 样式，该样式可以提供任务栏按钮。虽然这里没有给出其他扩展样式的代码，但可以通过在 www.microsoft.com 搜索 `WS_EX_APPWINDOW` 得到相关的文档说明。

为窗体增加一个菜单栏。使用 [Menu\(See 22.13\)](#) 命令创建一个普通的菜单，再调用此命令。例：

```
Menu, FileMenu, Add, &Open Ctrl+O, MenuFileOpen ; 关于 Ctrl+O 见下面说明
Menu, FileMenu, Add, E&xit, MenuHandler
Menu, HelpMenu, Add, &About, MenuHandler
Menu, MyMenuBar, Add, &File, :FileMenu ; 添加以上创建的两个菜单
Menu, MyMenuBar, Add, &Help, :HelpMenu
Gui, Menu, MyMenuBar
```

注意第一行中，&Open 后面的 Ctrl+O（中用制表符分开）。这是一个热键的标识，用户可以通过该快捷键选择相应的菜单。如果启用该功能，请使用以下上下文热键：

```
#IfWinActive(See 20.1.4) GUI Window's Title ahk_class AutoHotkeyGUI
^o:: ; 热键 Ctrl+O
MenuFileOpen:
Gui +OwnDialogs ; 在用户回到主界面前必须对 FileSelectFile 对话框作出响应。
FileSelectFile(See 16.23), SelectedFile
MsgBox 你选择的文件: %SelectedFile%.
return

; 以下部分代码仅在 Windows 95/98/Me 系统上需要
#IfWinActive
$^o::Send ^o
```

如果想要删除窗体的菜单，使用 [Gui Menu](#) 命令（后一个参数留空）

一旦菜单被设置为窗体的菜单栏，就不能应用于弹出菜单或是其他子菜单。这是因为菜单栏其内部是一种不同的格式（但这只是限定于菜单栏本身，而对其子菜单没有影响）。如果你非要实现，你可以先创建一个与菜单栏完全相同的另一个菜单，这样就可以作为其他的用途。

如果对一个正在用于窗体的菜单，使用类似 [Delete](#) 和 [DeleteAll](#) 的[菜单子命令\(See 22.13\)](#)，会无效并且会弹出错误的对话框（除非使用 [UseErrorLevel\(See 22.13\)](#) 来避免）。如果想改变窗体菜单可使用以下步骤：1) 先通过 [Gui Menu](#) (省略菜单名) 来使菜单和窗体分离；2) 改变菜单；3) 通过 [Gui, Menu, MyMenuBar](#) 重现将菜单附加到窗体；

[Gui Hide](#) 等同 [Gui Cancel\(See 19.3\)](#)。其他三个命令在不隐藏窗体的条件下对窗体进行最小化、最大化、恢复操作。如果窗体不存在--可能由于调用 [Gui Destroy\(See 19.3\)](#) 而使窗体销毁，则改命令无效。

使窗体在任务栏上的按钮闪烁。通过改变窗体的标题栏和任务栏上按钮的颜色来完成该效果。如果选项后加关键字 OFF 则标题栏和任务栏按钮恢复其原来的颜色（但实际上取决于操作系统的版本）。下面的例子中，窗体将闪烁三次，因为每一对闪烁都会使其回复原来的外观：

```
Loop 6
{
```

Gui Flash

```
Sleep 500 ; 该变量比较敏感，改变他它可能会引起异常
```

```
}
```

改变[当前线程](#)(See 30.19)默认的窗 体标识，该标识不能指定为 [GuiControl](#)(See 19.5), [GuiControlGet](#)(See 19.6),和窗体命令本 身。以下语句将默认窗体标识被改变为 2 : *Gui 2:Default* 参考 [thread's default window](#)(See 19.3) 了解关于默认窗体的更多信息。

虽然在下一段详细描述了简单的界面布局选项，但你会发现使用 Rajat 的 SmartGUI 来创建则更为简单。由于它完全的可视化可以让你所见即所得。 SmartGUI 是免费的，可以在 www.autohotkey.com/docs/SmartGUI/ 下载。

如果控件的大小或坐标没有指定，则控件的位置会以上一个控件的位置和大小自动的进行排列，控件的大小则由控件的类型和内容决定。

同时也支持下列选项：

R: 指定文本有多少行（后面可以加小数，如 R2.5）。使用 **R** 通常比 使用 **H**（高度）更加合适。如果同时指定了 **R** 和 **H** 选项，则 **R** 具有更高的优先级。对于组合框 [GroupBox](#)，这个设置将设置可以容纳多少个控件。对于 [DropDownLists](#)(See 19.4), [ComboBoxes](#), and [ListBoxes](#),这个设置在控件的列表内显示多少个列表项目（但是在 XP 及之后的系统中，推荐省略 [DropDownList](#) 和 [ComboBox](#) 控件的 **R** 和 **H** 选 项，这样弹出列表框会自动设置为系统桌面的高度）。**R** 对于其他的类型的控件，表示在控件 内，文字可以显示多少行。

W: 宽度，单位像素。如果省略该选项，宽度则根据控件的内容和类型进行计算。根据控件的类型默认的宽度定义如下：

属性页控件: 30 倍的当前字体的大小，加上 3 倍的边距 [X-margin](#)(See 19.3)

垂直进度条: 2 倍的当前字体大小。

水平进度条，水平滑动条，下拉列表, [ComboBoxes](#), [ListBoxes](#), [GroupBoxes](#), [Edits](#), and [Hotkeys](#): 15 倍的当前字体的大小（ [GroupBoxes](#) 为 18 以便能够显示边距）。

H: 高度，单位像素。**H** 和 **R** 选项如果省略, [DropDownLists](#), [ComboBoxes](#), [ListBoxes](#),和多行 [Edit](#) 控件默认为 3 行; [GroupBoxes](#) 默认为 2 行; 垂直滑动条和进度条默认为 5 行; 水平滑动条默认为 30 像素（除了指定了他的厚度）；水平进度条默认高度为字体的 2 倍大小；热键控件默认为 1 行；[Tab](#) 控件默认为 10 行。对于其他类型控件，其高度由他们所包含的内容所决定。注意，对于 [DropDownLists](#) 和 [ComboBoxes](#) 控件， **H** 表示为控件下拉列表部分的高度（即使高度设为很小，下拉列表也会至少显示一个列表项目）。同样，对于所有控件，设定选项 **R** 通常比设置 **H** 更有效，这样可以有效防止控件显示字体不完 整的问题。

wp+n, hp+n, wp-n, hp-n (其中 **n** 可为任意数字) 该选项可用于设置控件的高和宽等于之前的控件，加号和减号可以进行调整。例如 **wp** 表示设置控件的宽度和之前的一样，**wp-50** 则表示设置为之前宽度减去 50 个像素。

X: 控件的 X 坐标，例如指定 "x0 y0" 表示设置控件位置是客户区的左上角，(客户区不包括标题栏和菜单栏)。如果 **X** 忽略而 **Y** 未忽略，则控件的 X 坐标等于上一个控件的左边的位置。可以理解为该控件与上一个控件在一行。

Y: 控件的 Y 坐标，如果省略 **Y** 指定 **X**，则控件将置于上一个控件下方，也可理解为与之前的在一列中。

如果 **X, Y** 其中一个被省略，则窗口会根据之前的控件或字体自动布局。相比之下，如果你指定了每个控件的详细坐标，你需要手动调整所有控件的坐标，来使他们对齐。

如果 **X, Y** 都被省略，则使用标准的填充间距将控件置于上一个控件的下方。

在 **X** 和 **Y**，选项后跟一个加号表示增加控件的相对位置，表示控件相对于上一个控件的右端和底端的距离。例如，指定 **Y+10** 表示控件的 Y 坐标距离上个控件的右端为 10 个像素，而不是使用标准的填充距离。类似，指定 **X+10** 表示在控件的 X 坐标距离上一个控件的距离为 10 个像素。对于负数如 **X-10** 这样的写法是用来表示设置控件的绝对位置。所以指定负数的写法是在负号之前再加一个正号。例如： **X+-10**

xp+n, yp+n, xp-n, yp-n (其中 **n** 为任意数字) 该选项用于设置控件的位置为以上一个控件的右上端为 0 点的坐标内。该选项多用于 [GroupBox\(See 19.4\)](#) 控件以方便封装控件。

xm 和 **ym** 表示把控件放置在窗体边框的最左边和最上边（可以在后面加上正号或符号再加数字）。指定 **ym** 而忽略 **X** 坐标，控件则被放置在最上面并且在所有之前创建控件的右边，可是看作为是重新开始的新的一列。反之亦然。

xs 和 **ys**: 他们与 **xm** 和 **ym** 类似，区别在于相对坐标是相对于上一个具有关键字 [Section\(See 19.3\)](#) (**Section** 表示之后创建的控件为一个新的区域) 的控件来说的（第一个控件被视为一个新的区域，即使他没有指定关键字 **Section**）。指定 **ys** 而省略任何 **X** 坐标，控件将被放置于上一个控件的 **Y** 坐标，切在之前最近创建的具有具有 [Section\(See 19.3\)](#) 选项控件的右边，这可以理解为控件被放置于上一个控件区域的右边一列。例如：

```
gui, add, edit, w600 ; 在窗体的顶部增加一个宽度适当的 Edit 控件
gui, add, text, section, First Name: ; 保存当前控件的位置并且新定义一个区域
gui, add, text,, Last Name:
gui, add, edit, ys ; 在该区域创建新的一列
gui, add, edit
gui, show
```

反过来 (指定 **xs** 而省略 **Y** 坐标) 是一样的。

xs 和 **ys** 后跟正号或负号再加上一个数字，该选项为可选项。他是与关键词 [Section\(See 19.3\)](#) 一同使用的，表示使用之前定义的区域并且为只有创建的控件定义一个新区域。

V: 要使一个控件和一个变量相关联。在字母 **V** 后面加上变量名，该变量将是全局变量 (或是引用类型(See 10.)变量，或一个静态变量(See 10.))。例如，指定 **vMyEdit**; 则表示当调用提交命令 [Gui Submit\(See](#)

19.3) 后, 将控件的内容保存到变量 *MyEdit* 中。可能控件是不可输入型变量, 如 **Text** 控件或 **GroupBox** 控件, 但与之关联的变量仍可以用于 [GuiControl](#), ([See 19.5](#))[GuiControlGet](#)([See 19.6](#)), 和 [A_GuiControl](#)([See 9.](#)) 命令。则这些命令中可以直接用控件的关联变量名替代控件的唯一标识。注意: [Gui Submit](#)([See 19.3](#)) 不会改变非输入型控件的关联变量, 如 **Text** 和 **GroupBox** 控件, 也不会记录其片段内容 (如 [ListView](#)([See 19.7](#)) 和 [TreeView](#)([See 19.8](#)) 控件)。

G: 字母 G 后面加上标签名, 表示当用户点击或改变一个变量, 脚本会自动跳转到关联的子过程。[gCancel](#) 可以表示 [Gui Cancel](#)([See 19.3](#)) 事件运行的子过程 (如果脚本中存在一个 "Cancel" 标签, 则会调用该子过程), 子过程可能使用以下内置变量: [A_Gui](#)([See 9.](#)), [A_GuiControl](#)([See 9.](#)), [A_GuiEvent](#)([See 9.](#)), 和 [A_EventInfo](#)([See 9.](#))

注意: 如果省略前面的符号, 则默认为加号; 例如, **Wra** 就等于 **+Wrap** 而 **-Wrap** 则表示去除自动换行属性。

AltSubmit: 在提交后改变控件关联变量的值 *w* 为选项的顺序而不是选项名称。对于窗体上的 **DropDownList**, **ComboBox**, 和 **ListBox** 控件, 在使用 [Gui Submit](#)([See 19.3](#)) 命令后可将用户选择的选项储存为该项目的位置而不是该选项的名称。如果没有选择项目, 对于 **ComboBox** 将储存编辑框中的文字; 对于 **DropDownList** 或 **ListBox** 则[输出变量](#)([See 19.3](#))为 空。注意: **AltSubmit** 同时也影响 [GuiControlGet](#)([See 19.6](#)) 命令的输出值。

C: 文字的颜色 (对于 [buttons](#)([See 19.4](#)) 控件无效)。在字母 C 后面加上一个颜色的名称 (可参考[颜色名称表](#)([See 19.12](#))) 或是个一个 RGB 值 (0x 前缀可省略)。例如: **cRed**, **cFF2211**, **c0xFF2211**, **cDefault**

Disabled: 使一个输入型的控件处于灰色禁用状态, 这样可以防止用户的焦点和输入。使用 "[GuiControl](#)([See 19.5](#)) **Enable**" 命令重新使控件为可用。注意: 可添加 **ReadOnly** 选项, 设置 **Edit** 控件为只读。同样, 在 **Disable** 后面加上 0 或 1 表示可用和禁用。也就是说 **Disabled** 和 **Disabled %** 变量% 是一样的。

Hidden: 使控件创建时不可见。使用 "[GuiControl](#)([See 19.5](#)) **Show**" 可使其显示, **Hidden** 后可加上 0 或 1 表示可见和隐藏。也就是说 **Hidden** 和 **Hidden%** 变量% 相同。

Left: 使控件上可见文字部分左对齐。

Right: 使控件上可见文字部分右对其。对于 **checkboxes** 和 **radio buttons** 控件, 该选项会使选择框放在文字的右边而不是左边。

Center: 使控件上可见文字部分居中对齐。

Section: 创建一个新的控件区域, 并且保存该控件的坐标。之后创建的具有 **xs** 和 **ys** 选项控件则会以该坐标作为控件坐标的 0 点, 可参考[上文描述](#)([See 19.3](#))。

Tabstop: 当用户使用 **Tab** 键切换焦点控件时, 可跳过输入型控件。

Wrap: 可以使控件中的内容自动换行。在所有类型的控件中加入 **Wrap** 选项使其自动换行, 使用 **-Wrap** 禁用该功能。

VScroll: 如果该控件支持, 为该控件提供一个垂直的滚动条。

HScroll: 如果改控件支持, 为该控件提供一个水平的滚动条。只有 [ListBox\(See 19.4\)](#) 支持该选项。水平滚动条的宽度默认为 `ListBox` 宽度的 3 倍。如果想要指定不同的宽度, 在 `HScroll` 后加上需要指定的宽度即可。例如 `HScroll500` 表示在 `ListBox` 中加入一个宽度为 500 像素的水平滚动条。但, 如果指定的宽度比 `ListBox` 的宽度小, 则滚动条则不会显示 (但是要在窗体创建后随后再添加滚动条, 只能使用 [GuiControl\(See 19.5\)](#), `+HScroll500, MyScrollBar`, 来添加!)。

BackgroundTrans: 使用透明的背景, 允许任何在一个 `Text`, `Picture`, 或 `GroupBox` 控件之下的控件也可以正常显示。例如, 一个透明的 `Text` 控件显示在一个 `Picture` 控件之上让文字看起来好像是图片的一部分。可以使用 "[GuiControl\(See 19.5\)](#) +background" 来移除该文字。参考图 片控件的提交(See 19.4)了解更多关于图片透明的信息。已知的局限性: 背景透明选项不能够正确的应用于含有 [ListView\(See 19.7\)](#) 的 `Tab(See 19.4)` 控件中。

-Background (减去背景): 使用标准的背景颜色而不是用"Gui Color"命令来设置的颜色。这个经常被用于设置 `Tab` 控件单独的标准颜色。可以通过 "[GuiControl\(See 19.5\)](#) +background" 命令来移除该效果。

Border: 为控件添加一个稍细的边沿。大多数控件不需要设置该选项, 因为他们本身就提供了明确的类型边沿。当向一个已有控件添加一个边沿, 该控件的高和宽增加 1 个像素。

Hwnd 输出变量 [v1.0.46.01+]: 当使用 [Gui Add\(See 19.3\)](#) 时, 该选项会保存新建控件的句柄 (HWND) 到 输出变量 中。例如: `Gui, Add, Edit, vMyEdit HwndMyEditHwnd` (如果是在函数内部, `MyEditHwnd` 将视为一个[函数动态变量\(See 10.\)](#))。一个控件的句柄经常被使用于 [PostMessage\(See 28.1.12\)](#), [SendMessage\(See 28.1.12\)](#), 和 [DIICall\(See 18.3\)](#)。也可以直接用于 `ahk_id` [WinTitle\(See 30.2\)](#) (即使在指定了 [DetectHiddenWindows\(See 28.5\)](#) 为关闭的情况下也可以使用)。如果设置了子窗口, 父窗口的句柄可以通过 [Gui 2:+LastFoundExist\(See 19.3\)](#) 跟 [WinExist\(\)\(See 10.\)](#) 得到。

Theme: 该选项可以忽略窗体当前的主题设置。参见 [Gui +/-Theme\(See 19.3\)](#) 了解更多信息。

(未命名样式): 在一个代表样式的十六进制的[样式数字\(See 30.18\)](#)前加一个正号或负号, 来表示增加或去除改样式。

(未命名扩展样式): 在一个代表扩展样式的十六进制数字前面加一个字母 E 再添加正号或负号来表示增加或去除该扩展样式。如果前面的符号被省略则默认为正号。例如, `E0x200` 表示添加 `WS_EX_CLIENTEDGE` 样式。它可以为图片控件或其他控件提供一个下沉的边沿样式。尽管其他的样式没有在本帮助文档中给出, 但你可以在 www.microsoft.com 查找 `WS_EX_CLIENTEDGE` 得到更多信息。

如果在脚本中存在以下标签 (子过程), 则将自动与对应的窗体事件相关联:

GuiClose: 当窗口被以下事件关闭后会自动调用该子过程: 点击标题栏上的关闭按钮, 在窗体的系统菜单上选择"关闭";, 用 [WinClose\(See 28.14\)](#) 命令关闭窗体。如果该标签不存在, 关闭窗体后仅仅是将窗体隐藏, 与 [Gui Cancel\(See 19.3\)](#) 的效果一样。该命令最常用的就是在窗体关闭后退出脚本, 代码如下:

GuiClose:

ExitApp

GuiEscape: 当窗体在激活状态时，用户点击了 **ESC** 键，会调用改子过程。如果该标签不存在，则点击 **ESC** 后无效果。局限性：如果在窗体中的第一个控件被禁用（可能与控件的类型有关），则 **GuiEscape** 标签不会运行。但有一些情况可以产生这个效果。

GuiSize: 当用户调整大小时、最大化、最小化、或者恢复时调用该子过程。内建的变量将储存新的窗体客户区大小（区域包括标题栏、菜单栏、窗体边框）。[A_EventInfo\(See 9.\)](#) 和 [ErrorLevel\(See 30.8\)](#) 包含以下一种数字：

- 0: 窗体恢复大小，或者用户拖拽窗体的边沿
- 1: 窗体为最小化
- 2: 窗体为最大化

当用户改变窗体大小时，会相应该事件，你可以通过 **GuiSize** 来调整控件的位置及大小来适应窗体的大小。该功能可以通过 [#Include\(See 17.1\)](#) 加载 [Titan](#) 的 [Anchor](#) 脚本

GuiContextMenu: 当用户在窗体上（除去标题栏及菜单栏）鼠标右击或点击 **Apps** 按键或 **Shift+F10** 时，将调用改子过程。不同于其他窗体标签，**GuiContextMenu** 可以有多于一个的[线程\(See 30.19\)](#)。以下内置变量可以在该标签中使用：

1. [A_GuiControl\(See 9.\)](#), 包含收到事件消息的控件的[文字或变量名\(See 9.\)](#) (如果没有则为空)。
2. [A_EventInfo\(See 9.\)](#): 当右键点击到一个 **ListBox**, **ListView**, 或 **TreeView** 时 (可以通过 [A_GuiControl](#) 变量得到), [A_EventInfo](#) 表示控件中的哪个项目被选中:
 - [ListBox\(See 19.4\)](#) 或 [ListView\(See 19.7\)](#): [A_EventInfo](#) 为当前具有焦点的行号 (如果没有则为 0)。
 - [TreeView\(See 19.8\)](#): 对于右击, [A_EventInfo](#) 表示被点击项目的 ID 号 (如果用户没有点击到项目上则为 0)。对于 **AppsKey** 和 **Shift+F10** , [A_EventInfo](#) 表示当前选中项目的 ID 号。
3. [A_GuiX](#) 和 [A_GuiY](#), 表示菜单显示的 X 和 Y 坐标 (例如, [Menu\(See 22.13\)](#), [MyContext](#), [Show](#), [%A_GuiX%](#), [%A_GuiY%](#)) 以窗口的左上角为坐标 0 点。
4. [A_GuiEvent](#), 如果用户点击右击, 则 [A_GuiEvent](#) 为 **RightClick** 字符, 如果用户点击 **AppsKey** 或 **Shift+F10**, 则 [A_GuiEvent](#) 为 **Normal** 字符。

注意：由于 [Edit\(See 19.4\)](#) 和 [MonthCal\(See 19.4\)](#) 控件有他们自己的右键菜单，所以右击在这些控件中不会调用 **GuiContextMenu** 子过程。

GuiDropFiles: 当拖拽文件或文件夹到窗体上时，则调用该子过程（但如果该子过程已经在运行，则拖拽事件将忽略）。以下内置变量可用。

1. [A_GuiControl\(See 9.\)](#), 当用户拖拽文件到控件上时，该变量保存有控件的变量名或文字 (如果没有则为空)
2. [A_EventInfo\(See 9.\)](#) 和 [ErrorLevel\(See 30.8\)](#), 保存用户拖拽的文件数量
3. [A_GuiX](#) 和 [A_GuiY](#), 保存拖拽的 x 和 y 坐标 (相对于窗体，以左上角为零点)
4. [A_GuiEvent](#), 包含拖拽文件的名称，每个文件末尾以换行符(`n)结束。

要分别获取每个文件的名称，可以使用 [parsing loop\(See 17.14\)](#) 命令，例如：

```

; 示例 #1:
Loop, parse, A_GuiEvent, `n
{
    MsgBox, 4,, File number %A_Index% is: `n%A_LoopField%.`n`nContinue?
    IfMsgBox, No, Break
}
; 示例 #2: 只想获取第一个文件的文件名
Loop, parse, A_GuiEvent, `n
{
FirstFile = %A_LoopField%
Break
}
; 示例 #3: 把文件按照字母顺序排列
FileList = %A_GuiEvent%
Sort, FileList
Loop, parse, FileList, `n
MsgBox File number %A_Index% is: `n%A_LoopField%.

```

如想临时关闭窗口的拖拽功能，可通过 `Gui -E0x10` 将窗体的 `WS_EX_ACCEPTFILES` 样式移除。恢复拖拽使用 `use Gui +E0x10`

检测和响应其他的事件:其他的窗体事件可以通过 [OnMessage\(\)](#)(See 18.11) 来检测和响应。例如，当用户移动鼠标到相应的控件上时，脚本可以通过 `ToolTip` 来显示相关的帮助，详细信息可查看这个例子：

[GUI ToolTip example](#)(See 19.3)

对于多于一个窗体的界面，在第 `N` 个窗体中，会使用特定的前缀来区分以上提到的特殊标签。如，`2GuiEscape` 和 `2GuiClose` 是第二个窗体的默认标签。如果要使用自定义标签，请参考 [Gui +Label](#)(See 19.3)

每一个脚本可以同时创建 99 个窗体。若要对非默认窗体进行操作，可在逗号后面加上窗体序号，例如：

```

Gui, 2:Add, Text,, Text for about-box.
Gui, 2>Show

```

Gui 2:Default 若该命令调用后，则可以不要写 “2:” 前缀，因为已经设置为窗体 #2 为默认的窗体

脚本的性能会随着窗体数量的增多而下降。

一个窗体的[线程](#)(See 30.19)用来执行窗体的动作。窗体的动作包括：选择窗体上的[菜单](#)(See 19.3)，或触发了一个 [g-labels](#)(See 19.3)（如点击了一个按钮）。

一个界面线程的**默认窗体数目**是该窗体能够激活的线程数目，而非界面线程则默认为一个线程。

每当一个界面[线程](#)(See 30.19)启动，窗体本身被视为该线程最后发现的窗口(See 30.2)。这允许如 `WinMove`, (See 28.27)`WinHide`(See 28.22), `WinSet`(See 28.29), `WinSetTitle`(See 28.30), 和

ControlGetFocus(See 28.1.5) 这些命令对脚本创建的窗体操作时，可以省略 WinTitle 和 WinText（即使窗体处于隐藏状态）。

如果点击控件的时候，其对应的 **g-label(See 19.3)** 已经由于之前的点击而正在运行，则当前的点击没有效果，所产生的事件也将丢弃。如果想要防止这种情况发生，可以在子过程的第一行声明 **Critical(See 22.7)**（这样会延时热键点击等其他线程的启动）。

内置变量 **A_Gui(See 9.)** 和 **A_GuiControl(See 9.)** 分别表示窗体的序号，和当前运行线程操作的控件 ID。可以点击链接查看详细信息。

如果想要让多个事件运行同一个子过程，可以在子过程上方连续写下事件标签即可，如：

```
GuiEscape:  
GuiClose:  
ButtonCancel:  
ExitApp ; 以上的所有标签都会执行该语句
```

所有的界面线程(See 30.19)会以脚本的默认设置启动，如发送模式 **SendMode(See 20.13)**。这些默认的设置可以在[自动运行片段\(See 8.\)](#)来改变。

一个界面窗体可以通过 TAB 键来使焦点切换到下一个控件（如果控件去除了 **Tabstop(See 19.3)** 样式，则会跳过该控件）。导航的顺序则是由控件添加的顺序所决定。当一个窗体第一次显示时，可输入的控件默认具有 **Tabstop** 样式（大多数控件默认具有该样式）以便能够通过 Tab 键来进行控件焦点的切换。

控件可以添加字符 (&) 来创建该控件的快捷键，同时在控件的名称上会出现一个下划线（取决于系统设置）。用户可以通过 ALT + 相应的字母来完成点击的功能。对于 **buttons**, **checkboxes**, 和 **radio buttons** 控件，点击快捷键与点击该控件效果相同。对于 **GroupBoxes** 控件，点击快捷键则会把焦点自动跳转到 **GroupBoxes** 之后创建的第一个具有 **tabstop(See 19.3)** 样式的控件上。然而如果有多个控件具有相同的快捷键时，点击该快捷键会把轮流点击或激活这些控件。

如果想在控件中显示 & 地址符，可以连续输入两个 & 来表示，如：Save && Exit

当一个窗体创建后，他使用托盘图标作为窗体图标。如果需要不同的图标，可以在创建窗体前改变托盘图标。例如：**Menu(See 22.13)**, **Tray, Icon, MyIcon.ico**。他还可以有不同尺寸的图标（大图标可以用来显示在 alt-tab 菜单中）。这个功能可以通过 **DllCall** 和 **SendMessage** 来完成；例如：

```
hIcon32 := DllCall("LoadImage", uint, 0  
, str, "My Icon.ico" ; 图标文件，该文件可以包含多尺寸的图标  
, uint, 1 ; 图像的类型: IMAGE_ICON  
, int, 32, int, 32 ; 图像的高度和宽度，可以帮助 LoadImage 来确定那个是最佳尺寸。  
, uint, 0x10) ; 标志: LR_LOADFROMFILE  
Gui +LastFound
```

`SendMessage(See 28.1.12), 0x80, 1, hIcon32 ; 0x80 是 WM_SETICON; 1 表示 ICON_BIG , 0 表示 ICON_SMALL).`

`Gui Show`

由于操作系统的限制，在 WindowsXP 及以上版本中，Checkboxes, Radio buttons, 和 GroupBoxes 使用非默认颜色时，他们将变为经典样式。

相关主题: [window's margin.](#)

可使用 [GuiControl\(See 19.5\)](#) 和 [GuiControlGet\(See 19.6\)](#) 来操作窗体上的每个控件。

每个窗体可以包含 11,000 个控件。但当控件超过 5000 个时，请谨慎使用，这可能会因为系统的不稳定产生不确定的控件类型。

任一脚本如果包含了窗体命令，则脚本会自动永久运行 [persistent\(See 29.21\)](#) (即使窗体命令没有执行过)。同时默认以只允许一个实例运行，不过用户可以通过 [#SingleInstance\(See 22.2\)](#) 来设定。

[GuiControl\(See 19.5\)](#), [GuiControlGet\(See 19.6\)](#), [Menu\(See 22.13\)](#), [Control Types\(See 19.4\)](#), [ListView\(See 19.7\)](#), [TreeView\(See 19.8\)](#), [Control\(See 28.1.1\)](#), [ControlGet\(See 28.1.4\)](#), [SplashImage\(See 19.12\)](#), [MsgBox\(See 19.11\)](#), [FileSelectFile\(See 16.23\)](#), [FileSelectFolder\(See 16.24\)](#)

; 示例：完成一个输出文字的窗口：

```
Gui, +AlwaysOnTop +Disabled -SysMenu +Owner ; +Owner 避免出现任务栏窗口
Gui, Add, Text,, AHKBBS.CN
Gui, Show, NoActivate, Title of Window ; NoActivate 选项可以不改变当前激活窗口。
```

; 示例：使用 input-box 的简单例子，要求用户输入姓和名：

```
Gui, Add, Text,, 姓:
Gui, Add, Text,, 名:
Gui, Add, Edit, vFirstName ym ; The ym option starts a new column of controls.
Gui, Add, Edit, vLastName
Gui, Add, Button, default, OK ; 如果标签 ButtonOK 存在，则当用户点击 OK 按钮时会自动调用该标签。
Gui, Show,, 简单的输入框例子
return ; 自动运行片段结束。脚本保持空闲直到用户对窗口操作。
GuiClose:
ButtonOK:
Gui, Submit ; 保存用户输入后每个控件的数据
```

```
MsgBox 你输入了 "%FirstName% %LastName%"。
ExitApp
```

```
; 示例：属性页例子
Gui, Add, Tab2,, 第一页|第二页|第三页 ; 该控件需要 v1.0.47.05 版本
Gui, Add, Checkbox, vMyCheckbox, 一个复选框
Gui, Tab, 2
Gui, Add, Radio, vMyRadio, 单选框 1
Gui, Add, Radio,, 单选框 2
Gui, Tab, 3
Gui, Add, Edit, vMyEdit r5 ; r5 表示总共有 5 行
Gui, Tab ; 表示之后创建的控件不属于属性页控件
Gui, Add, Button, default xm, OK ; xm 表示把该控件显示在窗体的左下角
Gui, Show
return
ButtonOK:
GuiClose:
GuiEscape:
Gui, Submit ; 保存每个与控件相关联的变量
MsgBox 你选择了:`n%MyCheckbox%`n%MyRadio%`n%MyEdit%
ExitApp
```

```
; 示例：用 ListBox 显示一个包含文件名的列表
Gui, Add, Text,, 双击列表执行该文件，按 ESC 或关闭窗体退出程序
Gui, Add, ListBox, vMyListBox gMyListBox w280 r10
Gui, Add, Button,Default,Ok
Loop, C:\*.* ; 可修改为你想要显示的文件夹以及文件通配符
{
  GuiControl,, MyListBox, %A_LoopFileFullPath%
}
Gui, Show
return
MyListBox:
if A_GuiEvent <> DoubleClick
return
; 否则，用户双击列表中的项目，它和点击按钮的效果一样
; 如果 So fall through to the next label.
ButtonOK:
GuiControlGet, MyListBox ; 活动列表框中的选中项目
MsgBox, 4,, 你想要运行下面的文件么 ?`n`n%MyListBox%
IfMsgBox, No
return
```

```
; 否则, 运行该文件
Run, %MyListBox%,, UseErrorLevel
if ErrorLevel = ERROR
    MsgBox 不能够运行该文件。可能是找不到和该文件关联的程序
return
GuiClose:
GuiEscape:
ExitApp
```

```
; 示例: 当鼠标移动到控件上, 显示帮助提示
Gui, Add, Edit, vMyEdit
MyEdit_TT := "这是一个为 MyEdit 控件显示的提示"
Gui, Add, DropDownList, vMyDDL, Red|Green|Blue
MyDDL_TT := "从下拉列表中选择一个颜色"
Gui, Add, Checkbox, vMyCheck, 这个控件没有提示.
Gui, Show
OnMessage(See 18.11)(0x200, "WM_MOUSEMOVE")
return

WM_MOUSEMOVE()
{
    static CurrControl, PrevControl, _TT ; _TT 在下面的 ToolTip 中用到
    CurrControl := A_GuiControl
    If (CurrControl <> PrevControl and not InStr(CurrControl, " "))
    {
        ToolTip ; 关闭之前的 tooltip.
        SetTimer, DisplayToolTip, 1000
        PrevControl := CurrControl
    }
    return
}
DisplayToolTip:
SetTimer, DisplayToolTip, Off
ToolTip(See 19.15) % %CurrControl%_TT ; 第一个百分号表示后面是一个表达式
SetTimer, RemoveToolTip, 3000
return
RemoveToolTip:
SetTimer, RemoveToolTip, Off
ToolTip
return
}
GuiClose:
ExitApp
```

```
; 示例：利用透明窗体进行屏幕显示 (OSD)
CustomColor = EEA99 ; 可以是任何 RGB 值，他将用于后面的透明设置
Gui +LastFound +AlwaysOnTop -Caption +ToolWindow ; +ToolWindow 可以避免在任务栏显示按钮，并且不会出现在 alt-tab 菜单中
Gui, Color, %CustomColor%
Gui, Font, s32 ; 选择 32 号字体
Gui, Add, Text, vMyText cLime, XXXXX YYYYYY ; XX & YY 可以用来让窗体自动调整大小
; 使指定颜色的像素变得透明，并且使字体本身透明度为 150
WinSet, TransColor, %CustomColor% 150
SetTimer, UpdateOSD, 200
Gosub, UpdateOSD ; 使第一次更新立即运行，而不需要等待计时器来运行
Gui, Show, x0 y400 NoActivate ; 不激活窗体避免改变当前激活的窗口
return
UpdateOSD:
MouseGetPos, MouseX, MouseY
GuiControl,, MyText, X%MouseX%, Y%MouseY%
return
```

```
; 示例：一个带背景图像的进度条
Gui, Color, White
Gui, Add, Picture, x0 y0 h350 w450, %A_WinDir%\system32\ntimage.gif
Gui, Add, Button, Default xp+20 yp+250, Start the Bar Moving
Gui, Add, Progress, vMyProgress w416
Gui, Add, Text, vMyText wp ; wp 表示使用之前的宽度
Gui, Show
return
ButtonStartTheBarMoving:
Loop, %A_WinDir%\*.* {
if A_Index > 100
break
GuiControl,, MyProgress, %A_Index%
GuiControl,, MyText, %A_LoopFileName%
Sleep 50
}
GuiControl,, MyText, Bar finished.
return
GuiClose:
ExitApp
```

; 示例：一个简单的图像浏览窗体

```
Gui, +Resize
```

```

Gui, Add, Button, default, &Load New Image
Gui, Add, Radio, ym+5 x+10 vRadio checked, Load &actual size
Gui, Add, Radio, ym+5 x+10, Load to &fit screen
Gui, Add, Pic, xm vPic
Gui, Show
return
ButtonLoadNewImage:
FileSelectFile, file,,, Select an image:, Images (*.gif; *.jpg; *.bmp; *.png; *.tif; *.ico;
*.cur; *.ani; *.exe; *.dll)
if file =
return
Gui, Submit, NoHide ; 保存单选按钮的值
if Radio = 1 ; 按照实际尺寸显示图片
{
Width = 0
Height = 0
}
else ; 第二个选项：按照屏幕的固定尺寸显示
{
Width := A_ScreenWidth - 28 ; 减去 28 为了足够的空间显示边框和内边沿
Height = -1 ; 保持高宽比例
}
GuiControl,, Pic, *w%width% *h%height% %file% ; 加载图片
Gui, Show, xCenter y0 AutoSize, %file% ; 调整窗口大小以适合图片尺寸
return
GuiClose:
ExitApp

```

```

; 示例：一个带菜单栏的文本编辑窗体
; 创建菜单栏
Menu, FileMenu, Add, 新建文件 (&N), FileNew
Menu, FileMenu, Add, 打开文件 (&O)..., FileOpen
Menu, FileMenu, Add, 保存 (&S), FileSave
Menu, FileMenu, Add, 另存为 (&A)..., FileSaveAs
Menu, FileMenu, Add ; 增加一个分隔符
Menu, FileMenu, Add, 退出 (&X), FileExit
Menu, HelpMenu, Add, 关于 (&A), HelpAbout
; 把上面的菜单作为下面的子菜单
Menu, MyMenuBar, Add, 文件 (&F), :FileMenu
Menu, MyMenuBar, Add, 帮助 (&H), :HelpMenu
; 把上面的菜单作为窗体的菜单
Gui, Menu, MyMenuBar
; 创建编辑框控件并显示在窗体上

```

```

Gui, +Resize ; 使用用户可以调整窗体大小
Gui, Add, Edit, vMainEdit WantTab W600 R20
Gui, Show,, Untitled
CurrentFileName = ; 当前文件为空, 表示没有选择文件
return
FileNew:
GuiControl,, MainEdit ; 清空编辑框
return
FileOpen:
Gui +OwnDialogs ; 强制用户在关闭选择文件对话框后才能使用主窗体
FileSelectFile, SelectedFileName, 3,, Open File, Text Documents (*.txt)
if SelectedFileName = ; 如果没有选择文件
return
Gosub FileRead
return
FileRead: ; 读取并调用用户选择的文件
FileRead, MainEdit, %SelectedFileName% ; 把文件的内容读取到变量中
if ErrorLevel
{
    MsgBox Could not open "%SelectedFileName%".
    return
}
GuiControl,, MainEdit, %MainEdit% ; 在控件中显示文本
CurrentFileName = %SelectedFileName%
Gui, Show,, %CurrentFileName% ; 在标题栏中显示文件名
return
FileSave:
if CurrentFileName = ; 如果没有选中文件, 执行另存为
Goto FileSaveAs
Gosub SaveCurrentFile
return
FileSaveAs:
Gui +OwnDialogs ; 强制用户在在关闭选择文件对话框后才能使用主窗体
FileSelectFile, SelectedFileName, S16,, Save File, Text Documents (*.txt)
if SelectedFileName = ; 用户没有选中文件
return
CurrentFileName = %SelectedFileName%
Gosub SaveCurrentFile
return
SaveCurrentFile: ; 判断当前的文件不为空
IfExist %CurrentFileName%
{
    FileDelete %CurrentFileName%
    if ErrorLevel
    {

```

```

MsgBox The attempt to overwrite "%CurrentFileName%" failed.
return
}
}

GuiControlGet, MainEdit ; 获取当前 Edit 控件中的内容
FileAppend, %MainEdit%, %CurrentFileName% ; 保存当前的内容到文件
; 如果成功，在标题栏中显示文件名，与 FileSaveAs 相同
Gui, Show,, %CurrentFileName%
return
HelpAbout:
Gui, 2:+owner1 ; 设置窗体#1 的子窗体为 About Box 即窗体#2
Gui +Disabled ; 使主窗体无效
Gui, 2:Add, Text,, 欢迎访问 AHKBBS.CN
Gui, 2:Add, Button, Default, OK
Gui, 2>Show
return
2ButtonOK: ; 该标签用于
2GuiClose:
2GuiEscape:
Gui, 1:-Disabled ; 重新使主窗体有效，必须在下一步前执行改命令
Gui Destroy ; 销毁关于对话框
return
GuiDropFiles: ; 使窗体支持拖拽
Loop, parse, A_GuiEvent, `n
{
    SelectedFileName = %A_LoopField% ; 如果拖拽了多个文件的话，获取第一个文件
    break
}
Gosub FileRead
return
GuiSize:
if ErrorLevel = 1 ; 窗体是最小化状态则返回
return
; 否则是用户在调整大小或最大化窗体，则调整 Edit 控件的大小
NewWidth := A_GuiWidth - 20
NewHeight := A_GuiHeight - 20
GuiControl, Move, MainEdit, W%NewWidth% H%NewHeight%
return
FileExit: ; 用户从菜单中选择了 Exit
GuiClose: ; 用户关闭了窗体
ExitApp

```

翻译：BLooM.2 FantasyOnLine 2008年9月21日

19.4 Gui control types

- [Text, Edit, UpDown, Picture](#)
- [Button, Checkbox, Radio](#)
- [DropDownList, ComboBox](#)
- [ListBox, ListView\(See 19.7\), TreeView\(See 19.8\)](#)
- [Hotkey, DateTime, MonthCal](#)
- [Slider, Progress](#)
- [GroupBox, Tab2, StatusBar, Internet Explorer Control](#)

Description: A region containing borderless text that the user cannot edit. Often used to label other controls.

Example: Gui, Add, Text,, Please enter your name:

In this case, the last parameter is the string to display. It may contain linefeeds (`n) to start new lines. In addition, a single long line can be broken up into several shorter ones by means of a [continuation section](#)(See 8.).

If a width (W) is specified in *Options* but no [rows \(R\)](#)(See 19.3) or height (H), the text will be word-wrapped as needed, and the control's height will be set automatically.

Since the control's contents are in the last parameter of the Gui command, literal commas do not need to be escaped. This is also true for the last parameter of all other commands.

A [g-label](#)(See 19.3) such as **gMySubroutine** may be listed in the control's options. This would cause the *MySubroutine* label to be launched automatically whenever the user clicks the text. This can be used to simulate an underlined, blue hyperlink as shown in the following working script:

```
Gui, Font, underline
Gui, Add, Text, cBlue gLaunchGoogle, Click here to launch Google.
```

```
Gui, Font, norm
```

```
Gui, Show
```

```
return
```

```
LaunchGoogle:
```

```
Run www.google.com
```

```
return
```

A double-click can be detected by checking whether [A_GuiEvent](#)(See 9.) contains the word DoubleClick.

An ampersand (&) may be used in the text to underline one of its letters. For example:

```
Gui, Add, Text,, &First Name:  
Gui, Add, Edit
```

In the example above, the letter F will be underlined, which allows the user to press the [shortcut key](#)(See 19.3) Alt+F to set keyboard focus to the first input-capable control that was added after the text control. To instead display a literal ampersand, specify two consecutive ampersands (&&). To disable all special treatment of ampersands, include [0x80](#)(See 30.18) in the control's options.

Various [general options](#)(See 19.3) such as *Right*, *Center*, and *Hidden* are applicable to text controls.

Description: An area where free-form text can be typed by the user.

Example: Gui, Add, Edit, r9 vMyEdit, Text to appear inside the edit control (omit this parameter to start off empty).

The control will be multi-line if it has more than one row of text. For example, specifying r3 in *Options* will create a 3-line edit control with the following default properties: a vertical scroll bar, word-wrapping enabled, and the Enter key captured as part of the input rather than triggering the window's [default button](#).

To start a new line in a multi-line edit control, the last parameter (contents) may contain either a solitary linefeed (`n) or a carriage return and linefeed (`r`n). Both methods produce literal `r`n pairs inside the Edit control. However, when the control is saved to its variable via [Gui Submit](#)(See 19.3) or [GuiControlGet](#)(See 19.6), each `r`n in the text is always translated to a plain linefeed (`n). To write the text to a file, follow this example: [FileAppend](#)(See 16.4), %MyEdit%, C:\Saved File.txt

If the control has word-wrapping enabled (which is the default for multi-line edit controls), any wrapping that occurs as the user types will not produce linefeed characters (only the Enter keystroke can do that).

Although multi-line edit controls are limited to 64 KB of text on Windows 95/98/Me, they may have as much as 4 GB of text on Windows NT/2k/XP or later. As the user enters text, more memory is allocated as needed.

A [g-label](#)(See 19.3) such as **gMySubroutine** may be listed in the control's options. This would cause the *MySubroutine* label to be launched automatically whenever the user or the script changes the contents of the control.

TIP: To load a text file into an Edit control, use [FileRead](#)(See 16.19) and [GuiControl](#)(See 19.5).

For example:

```
Gui, Add, Edit, R20 vMyEdit
FileRead, FileContents, C:\My File.txt
GuiControl,, MyEdit, %FileContents%
```

Edit Options (to remove an option rather than adding it, precede it with a minus sign):

Limit: Restricts the user's input to the visible width of the edit field. Alternatively, to limit input to a specific number of characters, include a number immediately afterward. For example, Limit10 would allow no more than 10 characters to be entered.

Lowercase: The characters typed by the user are automatically converted to lowercase.

Multi: Makes it possible to have more than one line of text. However, it is usually not necessary to specify this because it will be auto-detected based on height (H), [rows \(R\)](#)(See 19.3), or contents (*Text*).

Number: Prevents the user from typing anything other than digits into the field (however, it is still possible to paste non-digits into it). An alternate way of forcing a numeric entry is to attach an [UpDown](#) control to the Edit.

Password: Hides the user's input (such as for password entry) by substituting masking characters for what the user types. If a non-default masking character is desired, include it immediately after the word Password. For example, Password* would make the masking character an asterisk rather than the black circle (bullet), which is the default on Windows XP.

Note: This option has no effect for multi-line edit controls.

ReadOnly: Prevents the user from changing the control's contents. However, the text can still be scrolled, selected and copied to the clipboard.

Tn: The letter T may be used to set tab stops inside a [multi-line edit control](#) (since tab stops determine the column positions to which literal TAB characters will jump, they can be used to format the text into columns). If the letter T is not used, tab stops are set at every 32 dialog units (the width of each "dialog unit" is determined by the operating system). If the letter T is used only once, tab stops are set at every n units across the entire width of the control. For example, *Gui, Add, Edit, vMyEdit r16 t64* would double the default distance between tab stops. To have custom tab stops, specify the letter T multiple times as in the following example: *Gui, Add, Edit, vMyEdit r16 t8 t16 t32 t64 t128*. One tab stop is set for each of the absolute column positions in the list, up to a maximum of 50 tab stops. Note: Tab stops require a multiline edit control.

Uppercase: The characters typed by the user are automatically converted to uppercase.

WantCtrlA [v1.0.44+]: Specify -WantCtrlA (minus WantCtrlA) to prevent the user's press of Control-A from selecting all text in the edit control.

WantReturn: Specify -WantReturn (that is, a minus sign followed by WantReturn) to prevent a multi-line edit control from capturing the Enter keystroke. Pressing Enter will then be the same as pressing the window's [default button](#) (if any). In this case, the user may press Control-Enter to start a new line.

WantTab: Causes a tab keystroke to produce a tab character rather than navigating to the next control. Without this option, the user may press Control-Tab to produce a tab character inside a multi-line edit control. Note: Although *WantTab* also works in a single-line edit control, each tab character is displayed as an empty-box character (though it is stored as a real tab).

-Wrap (minus wrap): Turns off word-wrapping in a multi-line edit control. Since this style cannot be changed after the control has been created, use one of the following to change it: 1) [Destroy](#)(See 19.3) then recreate the window and its control; or 2) Create two overlapping edit controls, one with wrapping enabled and the other without it. The one not currently in use can be kept empty and/or hidden.

A more powerful edit control: HiEdit is a free, multitabbed, large-file edit control consuming very little memory. It can edit both text and binary files. For details and a demonstration, see www.autohotkey.com/forum/topic19141.html

Description: A pair of arrow buttons that the user can click to increase or decrease a value. By default, an UpDown control automatically snaps onto the previously added control. This previous control is known as the UpDown's *buddy control*. The most common example is a "spinner", which is an UpDown attached to an [Edit control](#). For example:

```
Gui, Add, Edit
Gui, Add, UpDown, vMyUpDown Range1-10, 5
```

In the example above, the Edit control is the UpDown's buddy control. Whenever the user presses one of the arrow buttons, the number in the Edit control is automatically increased or decreased.

An UpDown's buddy control can also be a [Text control](#) or [ListBox](#). However, due to OS limitations, controls other than than these (such as ComboBox and DropDownList) might not work properly with [g-labels](#)(See 19.3) and other features.

Specify the UpDown's starting position as the last parameter (if omitted, it starts off at 0 or the number in the allowable range that is closest to 0).

When the [Gui Submit](#)(See 19.3) command is used, the control's [associated output variable](#)(See 19.3) (if any) receives the current numeric position of the UpDown. If the UpDown is attached to an Edit control and you do not wish to validate the user's input, it is best to use the UpDown's value rather than the Edit's. This is because the UpDown will always yield an in-range number, even when the user has typed something non-numeric or out-of-range in the Edit control. On a related note, numbers with more than three digits get a [thousands separator](#)(See 30.18)

(such as comma) by default. These separators are stored in the Edit's output variable but not that of the UpDown.

If the UpDown has a [g-label](#)(See 19.3), it will be launched whenever the user clicks one of the arrow buttons or presses an arrow key on the keyboard. Each launch of the g-label also stores the UpDown's position in its [associated output variable](#)(See 19.3) (if any).

UpDown Options:

Horz: Makes the control's buttons point left/right rather than up/down. By default, *Horz* also makes the control isolated (no buddy). This can be overridden by specifying *Horz 16* in the control's options.

Left: Puts the UpDown on the left side of its buddy rather than the right.

Range: Sets the range to be something other than 0 to 100. After the word Range, specify the minimum, a dash, and maximum. For example, Range1-1000 would allow a number between 1 and 1000 to be selected; Range-50-50 would allow a number between -50 and 50; and Range-10--5 would allow a number between -10 and -5. The minimum and maximum may be swapped to cause the arrows to move in the opposite of their normal direction. The broadest allowable range is -2147483648-2147483647. However, Windows 95 and NT4 require Internet Explorer 5.0 or later to support a range broader than -32767-32767. Finally, if the buddy control is a [ListBox](#), the range defaults to 32767-0 for verticals and the inverse for horizontals ([Horz](#)).

Wrap: Causes the control to wrap around to the other end of its range when the user attempts to go beyond the minimum or maximum. Without *Wrap*, the control stops when the minimum or maximum is reached.

-16 (minus 16): Causes a vertical UpDown to be isolated; that is, it will have no buddy. This also causes the control to obey any specified width, height, and position rather than conforming to the size of its buddy control. In addition, an isolated UpDown tracks its own position internally. This position can be retrieved normally by means such as [Gui Submit](#)(See 19.3).

0x80: Include 0x80 in *Options* to omit the thousands separator that is normally present between every three decimal digits in the buddy control. However, this style is normally not used because the separators are omitted from the number whenever the script retrieves it from the UpDown control itself (rather than its buddy control).

Increments other than 1: In [this script](#), NumEric demonstrates how to change an UpDown's increment to a value other than 1 (such as 5 or 0.1).

Description: An area containing an image (see last two paragraphs for supported file types). The last parameter is the filename of the image, which is assumed to be in [A_WorkingDir](#)(See 9.) if an absolute path isn't specified.

Example: Gui, Add, Picture, w300 h-1, C:\My Pictures\Company Logo.gif

To retain the image's actual width and/or height, omit the W and/or H options. Otherwise, the image is scaled to the specified width and/or height (this width and height also determines which icon to load from a multi-icon .ICO file). To shrink or enlarge the image while preserving its aspect ratio, specify -1 for one of the dimensions and a positive number for the other. For example, specifying "w200 h-1" would make the image 200 pixels wide and cause its height to be set automatically. If the picture cannot be loaded or displayed (e.g. file not found), the control is left empty and its width and height are set to zero.

A [g-label](#)(See 19.3) such as **gMySubroutine** may be listed in the control's options. This would cause the *MySubroutine* label to be launched automatically whenever the user clicks the picture. A double-click can be detected by checking whether [A_GuiEvent](#)(See 9.) contains the word DoubleClick.

To use a picture as a background for other controls, the picture should normally be added prior to those controls. However, if those controls are input-capable and the picture has a [g-label](#)(See 19.3), create the picture after the other controls and include 0x4000000 (which is WS_CLIPSIBLINGS) in the picture's *Options*. This trick also allows a picture to be the background behind a [Tab control](#) or [ListView](#)(See 19.7).

Icons, cursors, and animated cursors: Icons and cursors may be loaded from the following types of files: ICO, CUR, ANI, EXE, DLL, CPL, SCR, and other types that contain icon resources. To use an icon group other than the first one in the file, include in *Options* the word Icon followed by the number of the group. In the following example, the default icon from the second icon group would be used: *Gui, Add, Picture, Icon2, C:\My Application.exe*

Specifying the word AltSubmit in *Options* tells the program to use Microsoft's GDIPlus.dll to load the image, which might result in a different appearance for GIF, BMP, and icon images. For example, it would load an ICO/GIF that has a transparent background as a transparent bitmap, which allows the [BackgroundTrans](#)(See 19.3) option to take effect. If GDIPlus is not available (see next paragraph), AltSubmit is ignored and the image is loaded using the normal method.

All operating systems support GIF, JPG, BMP, ICO, CUR, and ANI images. On Windows XP or later, additional image formats such as PNG, TIF, Exif, WMF, and EMF are supported. Operating systems older than XP can be given support by copying Microsoft's free GDI+ DLL into the AutoHotkey.exe folder (but in the case of a [compiled script](#)(See 8.), copy the DLL into the script's folder). To download the DLL, search for the following phrase at www.microsoft.com/gdi redistributable

Due to an OS bug, animated cursors scaled to a size greater than 90x90 on Windows 95/98/Me might crash the script.

Animated GIFs: Although animated GIF files can be displayed in a picture control, they will not actually be animated. To solve this, use the AniGIF DLL (which is free for non-commercial use) as demonstrated at www.autohotkey.com/forum/topic19264.html

Description: A pushbutton, which can be pressed to trigger an action. In this case, the last parameter is the name of the button (shown on the button itself), which may include linefeeds (`n) to start new lines.

Example: Gui, Add, Button, Default, OK

The example above includes the word [Default](#) in its *Options* to make "OK" the default button. The default button's action is automatically triggered whenever the user presses ENTER, except when the keyboard focus is on a different button or a multi-line edit control having the [WantReturn](#) style. To later change the default button to another button, follow this example, which makes the Cancel button become the default: [GuiControl](#)(See 19.5), `+default, Cancel`. To later change the window to have no default button, follow this example: [GuiControl](#), `-default, OK`

An ampersand (&) may be used in the name button to underline one of its letters. For example:

Gui, Add, Button,, &Pause

In the example above, the letter P will be underlined, which allows the user to press Alt+P as [shortcut key](#)(See 19.3). To display a literal ampersand, specify two consecutive ampersands (&&).

If a button lacks an explicit [g-label](#)(See 19.3), an automatic label is assumed. For example, if the first GUI window contains an OK button, the ButtonOK label (if it exists) will be launched when the button is pressed. For GUI windows [other than the first](#)(See 19.3), the window number is included in front of the button's automatic label; for example: 2ButtonOK.

If the text on the button contains spaces or any of the characters in the set &`r`n`t`, its automatic label omits those characters. For example, a button titled "&Pause" would have an automatic label of ButtonPause. Similarly, a button titled "Save && Exit" would have an automatic label of ButtonSaveExit (the double-ampersand is used to display a single, literal ampersand).

Known limitation: Certain desktop themes might not display a button's text properly. If this occurs, try including `-Wrap` (minus Wrap) in the button's options. However, this also prevents having more than one line of text.

Description: A small box that can be checked or unchecked to represent On/Off, Yes/No, etc.

Example: Gui, Add, Checkbox, vShipToBillingAddress, Ship to billing address?

The last parameter is a label displayed next to the box, which is typically used as a prompt or description of what the checkbox does. It may include linefeeds (`n) to start new lines. If a width (W) is specified in *Options* but no [rows \(R\)](#)(See 19.3) or height (H), the control's text will be word-wrapped as needed, and the control's height will be set automatically. The checkbox's [associated output variable](#)(See 19.3) (if any) receives the number 1 for checked, 0 for unchecked, and -1 for gray/indeterminate.

Specify the word `Check3` in *Options* to enable a third state that displays a gray checkmark instead of a black one (the gray state indicates that the checkbox is neither checked nor unchecked). Specify the word `Checked` or `CheckedGray` in *Options* to have the checkbox start off with a black or gray checkmark, respectively. The word `Checked` may optionally be followed immediately by a 0, 1, or -1 to indicate the starting state. In other words, "Checked" and "Checked%VarContainingOne%" are the same.

A [g-label](#)(See 19.3) such as `gMySubroutine` may be listed in the control's options. This would cause the `MySubroutine` label to be launched automatically whenever the user clicks or changes the checkbox.

Known limitation: Certain desktop themes might not display a button's text properly. If this occurs, try including `-Wrap` (minus Wrap) in the button's options. However, this also prevents having more than one line of text.

A Radio button is a small empty circle that can be checked (on) or unchecked (off).

Example: Gui, Add, Radio, vMyRadioGroup, Wait for all items to be in stock before shipping.

These controls usually appear in *radio groups*, each of which contains two or more radio buttons. When the user clicks a radio button to turn it on, any others in its radio group are turned off automatically (the user may also navigate inside a group with the arrow keys). A radio group is created automatically around all consecutively added radio buttons. To start a new group, specify the word `Group` in the *Options* of the first button of the new group -- or simply add a non-radio control in between, since that automatically starts a new group.

For the last parameter, specify the label to display to the right of the radio button. This label is typically used as a prompt or description, and it may include linefeeds (`n) to start new lines. If a width (W) is specified in *Options* but no rows (R) or height (H), the control's text will be word-wrapped as needed, and the control's height will be set automatically.

Specify the word `Checked` in *Options* to have the button start off in the "on" state. The word `Checked` may optionally be followed immediately by a 0 or 1 to indicate the starting state: 0 for unchecked and 1 for checked. In other words, "Checked" and "Checked%VarContainingOne%" are the same.

The radio button's [associated output variable](#)(See 19.3) (if any) receives the number 1 for "on" and 0 for "off". However, if only one button in a radio group has a variable, that variable will instead receive the number of the currently selected button: 1 is the first radio button (according to original creation order), 2 is the second, and so on. If there is no button selected, 0 is stored.

A [g-label](#)(See 19.3) such as `gMySubroutine` may be listed in the control's options. This would cause the `MySubroutine` label to be launched automatically whenever the user turns on the button. Unlike the single-variable mode in the previous paragraph, the g-label must be

specified for each button in a radio group for which the label should be launched. This allows the flexibility to ignore the clicks of certain buttons. Finally, a double-click can be detected by checking whether [A_GuiEvent](#)(See 9.) contains the word DoubleClick.

Known limitation: Certain desktop themes might not display a button's text properly. If this occurs, try including *-Wrap* (minus Wrap) in the button's options. However, this also prevents having more than one line of text.

Description: A list of choices that is displayed in response to pressing a small button. In this case, the last parameter is a pipe-delimited list of choices such as Choice1|Choice2|Choice3.

Example: Gui, Add, DropDownList, vColorChoice, Black|White|Red|Green|Blue

To have one of the items pre-selected when the window first appears, include two pipe characters after it. Alternatively, include in *Options* the word [Choose](#) followed immediately by the number to pre-select. For example, Choose5 would pre-select the fifth item (as with other options, it can also be a variable such as Choose%Var%). To change the choice or add/remove entries from the list after the control has been created, use [GuiControl](#)(See 19.5).

Specify either the word [Uppercase](#) or [Lowercase](#) in *Options* to automatically convert all items in the list to uppercase or lowercase. Specify the word [Sort](#) to automatically sort the contents of the list alphabetically (this also affects any items added later via [GuiControl](#)(See 19.5)). The Sort option also enables incremental searching whenever the list is dropped down; this allows an item to be selected by typing the first few characters of its name.

When the [Gui Submit](#)(See 19.3) command is used, the control's [associated output variable](#)(See 19.3) (if any) receives the text of the currently selected item. However, if the control has the [AltSubmit](#)(See 19.3) property, the output variable will receive the item's position number instead (the first item is 1, the second is 2, etc.).

A [g-label](#)(See 19.3) such as **gMySubroutine** may be listed in the control's options. This would cause the *MySubroutine* label to be launched automatically whenever the user selects a new item.

Use the [R or H option](#)(See 19.3) to control the height of the popup list. For example, specifying R5 would make the list 5 rows tall. If both R and H are omitted, the list will automatically expand to take advantage of the available height of the user's desktop (however, operating systems older than Windows XP will show 3 rows by default).

The separator between fields may be changed to something other than pipe (|). For example [Gui +Delimiter`n](#)(See 19.3) would change it to linefeed and [Gui +DelimiterTab](#) would change it to tab (`t).

Description: Same as DropDownList but also permits free-form text to be entered as an alternative to picking an item from the list.

Example: Gui, Add, ComboBox, vColorChoice, Red|Green|Blue|Black|White

In addition to allowing all the same options as DropDownList above, the word [Limit](#) may be included in *Options* to restrict the user's input to the visible width of the ComboBox's edit field. Also, the word [Simple](#) may be specified to make the ComboBox behave as though it is an Edit field with a ListBox beneath it.

When the [Gui Submit](#)(See 19.3) command is used, the control's [associated output variable](#)(See 19.3) (if any) receives the text of the currently selected item. However, if the control has the [AltSubmit](#)(See 19.3) property, the output variable will receive the item's position number instead (the first item is 1, the second is 2, etc.). If either case, if there is no selected item, the output variable will be set to the contents of the ComboBox's edit field.

A [g-label](#)(See 19.3) such as **gMySubroutine** may be listed in the control's options. This would cause the *MySubroutine* label to be launched automatically whenever the user selects a new item.

Description: A relatively tall box containing a list of choices that can be selected. In this case, the last parameter is a pipe-delimited list of choices such as Choice1|Choice2|Choice3.

Example: Gui, Add, ListBox, vColorChoice, Red|Green|Blue|Black|White

To have list item(s) pre-selected when the window first appears, include two pipe characters after each (the [Multi](#) option is required if more than one item is to be pre-selected).

Alternatively, include in *Options* the word [Choose](#) followed immediately by a single item number to pre-select. For example, Choose5 would pre-select the fifth item. To change the choice or add/remove entries from the list after the control has been created, use [GuiControl](#)(See 19.5).

When the [Gui Submit](#)(See 19.3) command is used, the control's [associated output variable](#)(See 19.3) (if any) receives the text of the currently selected item. However, if the control has the [AltSubmit](#)(See 19.3) property, the output variable instead receives the item's position number (the first item is 1, the second is 2, etc.).

A [g-label](#)(See 19.3) such as **gMySubroutine** may be listed in the control's options. This would cause the *MySubroutine* label to be launched automatically whenever the user selects a new item. If the user double-clicks an item, the built-in variable `A_GuiEvent` will contain the string `DoubleClick` rather than `Normal`. Also, the variable `A_EventInfo` will contain the position of the item that was double-clicked (1 is the first item, 2 is the second, etc.).

When adding a large number of items to a ListBox, performance may be improved by using `GuiControl, -Redraw, MyListBox` prior to the operation, and `GuiControl, +Redraw, MyListBox` afterward.

ListBox Options:

Choose: See [above](#).

Multi: Allows more than one item to be selected simultaneously via shift-click and control-click (to avoid the need for shift/control-click, specify [the number 8](#)(See 30.18) instead of the word Multi). In this case, [Gui Submit](#)(See 19.3) stores a pipe delimited list of item-strings in the control's [output variable](#)(See 19.3). However, if the [AltSubmit](#)(See 19.3) option is in effect, [Gui Submit](#)(See 19.3) stores a pipe-delimited list of item numbers instead. For example, 1|2|3 would indicate that the first three items are selected. To extract the individual items from the string, use a [parsing loop](#)(See 17.14) such as this example:

```
Loop, parse, MyListBox, |
{
    MsgBox Selection number %A_Index% is %A_LoopField%.
}
```

The separator between fields may be changed to something other than pipe (|). For example [Gui +Delimiter`n](#)(See 19.3) would change it to linefeed and [Gui +DelimiterTab](#) would change it to tab (`t).

ReadOnly: Prevents items from being visibly highlighted when they are selected (but [Gui Submit](#)(See 19.3) will still store the selected item).

Sort: Automatically sorts the contents of the list alphabetically (this also affects any items added later via [GuiControl](#)(See 19.5)). The Sort option also enables incremental searching, which allows an item to be selected by typing the first few characters of its name.

Tn: The letter T may be used to set tab stops, which can be used to format the text into columns. If the letter T is not used, tab stops are set at every 32 dialog units (the width of each "dialog unit" is determined by the operating system). If the letter T is used only once, tab stops are set at every n units across the entire width of the control. For example, "Gui, Add, ListBox, vMyListBox t64" would double the default distance between tab stops. To have custom tab stops, specify the letter T multiple times as in the following example: Gui, Add, ListBox, vMyListBox t8 t16 t32 t64 t128. One tab stop is set for each of the absolute column positions in the list, up to a maximum of 50 tab stops.

0x100: Include 0x100 in options to turn on the LBS_NOINTEGRALHEIGHT style. This forces the ListBox to be exactly the height specified rather than a height that prevents a partial row from appearing at the bottom. This also prevents the ListBox from shrinking when its font is changed.

See separate pages [ListView](#)(See 19.7) and [TreeView](#)(See 19.8).

Description: A box that looks like a single-line edit control but instead accepts a keyboard combination pressed by the user. For example, if the user presses Control+Alt+C on an English keyboard layout, the box would display "Ctrl + Alt + C".

Example: Gui, Add, Hotkey, vChosenHotkey

When the [Gui Submit](#)(See 19.3) command is used, the control's [associated output variable](#)(See 19.3) (if any) receives the hotkey modifiers and name, which are compatible with the [Hotkey](#)(See 20.1.10) command. Examples: ^!C, +!Home, +^Down, ^Numpad1, !NumpadEnd. If there is no hotkey in the control, the output variable is made blank. Note: Some keys are displayed the same even though they are retrieved as different names. For example, both ^Numpad7 and ^NumpadHome might be displayed as Ctrl + Num 7.

By default, the control starts off with no hotkey specified. To instead have a default, specify its modifiers and name as the last parameter as in this example: Gui, Add, Hotkey, vChosenHotkey, ^!p

The only modifiers supported are ^ (Control), ! (Alt), and + (Shift). See the [key list](#)(See 7.) for available key names.

A [g-label](#)(See 19.3) such as **gMySubroutine** may be listed in the control's options. This would cause the *MySubroutine* label to be launched automatically whenever the user changes the hotkey. Each launch of the g-label also stores the hotkey in control's [associated output variable](#)(See 19.3) (if any). Note: the g-label is launched even when an incomplete hotkey is present. For example, if the user holds down the Control key, the g-label is launched once and the output variable contains only a circumflex (^). When the user completes the hotkey, the label is launched again and the variable contains the complete hotkey.

To restrict the types of hotkeys the user may enter, include the word [Limit](#) followed by the sum of one or more of the following numbers:

- 1: Prevent unmodified keys
- 2: Prevent Shift-only keys
- 4: Prevent Control-only keys
- 8: Prevent Alt-only keys
- 16: Prevent Shift-Control keys
- 32: Prevent Shift-Alt keys
- 64: This value is not supported (it will not behave correctly).
- 128: Prevent Shift-Control-Alt keys.

For example, Limit1 would prevent unmodified hotkeys such as letters and numbers from being entered, and Limit15 would require at least two modifier keys. If the user types a forbidden modifier combination, the Control+Alt combination is automatically and visibly substituted.

The Hotkey control has limited capabilities. For example, it does not support mouse/joystick hotkeys or the Windows key (LWin and RWin). One way to work around this is to provide one

or more [checkboxes](#) as a means for the user to enable extra modifiers such as the Windows key.

Description: A box that looks like a single-line edit control but instead accepts a date and/or time. A drop-down calendar is also provided.

Example: Gui, Add, DateTime, vMyDateTime, LongDate

The last parameter may be one of the following:

(omitted): When omitted, the locale's short date format is used. For example, in some locales it would look like: 6/1/2005

LongDate: Uses the locale's long date format. For example, in some locales it would look like: Wednesday, June 01, 2005

Time: Shows only the time using the locale's time format. Although the date is not shown, it is still present in the control and will be retrieved along with the time in the [YYYYMMDDHH24MISS](#)(See 16.26) format.

(custom format): Specify any combination of [date and time formats](#)(See 27.1). For example, *M/yy HH:mm* would look like 6/1/05 21:37. Similarly, *ddd MMMM d, yyyy hh:mm:ss tt* would look like Wednesday June 1, 2005 09:37:45 PM. Letters and numbers to be displayed literally should be enclosed in single quotes as in this example: '*Date:*' MM/dd/yy '*Time:*' hh:mm:ss tt. By contrast, non-alphanumeric characters such as spaces, tabs, slashes, colons, commas, and other punctuation do not need to be enclosed in single quotes. The exception to this is the single quote character itself: to produce it literally, use four consecutive single quotes ("'"), or just two if the quote is already inside an outer pair of quotes.

DateTime Usage

To have a date other than today pre-selected, include in *Options* the word [Choose](#) followed immediately by a date in YYYYMMDD format. For example, Choose20050531 would pre-select May 31, 2005 (as with other options, it can also be a variable such as Choose%Var%). To have no date/time selected, specify [ChooseNone](#). [ChooseNone](#) also creates a checkbox inside the control that is unchecked whenever the control has no date. Whenever the control has no date, [Gui Submit](#)(See 19.3) and [GuiControlGet](#)(See 19.6) will retrieve a blank value (empty string).

The time of day may optionally be present. However, it must always be preceded by a date when going into or coming out of the control. The format of the time portion is HH24MISS (hours, minutes, seconds), where HH24 is expressed in 24-hour format; for example, 09 is 9am and 21 is 9pm. Thus, a complete date-time string would have the format [YYYYMMDDHH24MISS](#)(See 16.26).

When specifying dates in the YYYYMMDDHH24MISS format, only the leading part needs to be present. Any remaining element that has been omitted will be supplied with the following default values:

MM: Month 01

DD: Day 01

HH24: Hour 00

MI: Minute 00

SS: Second 00

Within the drop-down calendar, the today-string at the bottom can be clicked to select today's date. In addition, the year and month name are clickable and allow easy navigation to a new month or year.

Keyboard navigation: Use the Up/Down arrow keys, NumpadPlus/Minus, and Home/End to increase or decrease the control's values. Use LeftArrow and RightArrow to move from field to field inside the control. Within the drop-down calendar, use the arrow keys to move from day to day; use PageUp/Down to move backward/forward by one month; use Ctrl-PageUp/Down to move backward/forward by one year; and use Home/End to select the first/last day of the month.

When the [Gui Submit](#)(See 19.3) command is used, the control's [associated output variable](#)(See 19.3) (if any) receives the selected date and time in [YYYYMMDDHH24MISS](#)(See 16.26) format. Both the date and the time are present regardless of whether they were actually visible in the control.

If the control has a [g-label](#)(See 19.3), the label is launched whenever the user changes the date or time. For each launch, the control's [associated output variable](#)(See 19.3) (if any) is automatically updated with the currently selected date/time.

Windows 95 and NT4 require DLL versions at least as new as those distributed with Internet Explorer 3.0 to support DateTime controls.

DateTime Options:

Choose: See [above](#).

Range: Restricts how far back or forward in time the selected date can be. After the word Range, specify the minimum and maximum dates in YYYYMMDD format (with a dash between them). For example, Range20050101-20050615 would restrict the date to the first 5.5 months of 2005. Either the minimum or maximum may be omitted to leave the control unrestricted in that direction. For example, Range20010101 would prevent a date prior to 2001 from being selected and Range-20091231 (leading dash) would prevent a date later than 2009 from being selected. Without the Range option, any date between the years 1601 and 9999 can be selected. The time of day cannot be restricted.

Right: Causes the drop-down calendar to drop down on the right side of the control instead of the left.

1: Specify the number 1 in *Options* to provide an up-down control to the right of the control to modify date-time values, which replaces the of the drop-down month calendar that would otherwise be available.

2: Specify the number 2 in *Options* to provide a checkbox inside the control that the user may uncheck to indicate that no date/time is selected. Once the control is created, this option cannot be changed.

Description: A tall and wide control that displays all the days of the month in calendar format. The user may select a single date or a range of dates.

Example: Gui, Add, MonthCal, vMyCalendar

To have a date other than today pre-selected, specify it as the last parameter in YYYYMMDD format (e.g. 20050531). A range of dates may also be pre-selected by including a dash between two dates (e.g. 20050525-20050531).

It is usually best to omit width (W) and height (H) for a MonthCal because it automatically sizes itself to fit exactly one month. To display more than one month vertically, specify R2 or higher in *Options*. To display more than one month horizontally, specify W-2 (W negative two) or higher. These options may both be present to expand in both directions.

The today-string at the bottom of the control can be clicked to select today's date. In addition, the year and month name are clickable and allow easy selection of a new year or month.

Unlike [DateTime](#)'s drop-down calendar, keyboard navigation is generally not supported in a MonthCal.

When the [Gui Submit](#)(See 19.3) command is used, the control's [associated output variable](#)(See 19.3) (if any) receives the selected date in YYYYMMDD format (without any time portion). However, when the [multi-select](#) option is in effect, the minimum and maximum dates are retrieved with a dash between them (e.g. 20050101-20050108). If only a single date was selected in a multi-select calendar, the minimum and maximum are both present but identical. [StringSplit](#)(See 27.21) can be used to separate the dates. For example, the following would put the minimum in Date1 and the maximum in Date2: *StringSplit, Date, MyMonthCal, -*

If the MonthCal has a [g-label](#)(See 19.3), each launch of it updates the control's [associated output variable](#)(See 19.3) (if any) with the currently selected date or range. By default, the label is launched only when: 1) the user changes the selection; or 2) every two minutes in case a new day has arrived (this behavior is a quirk of the OS). However, if the word AltSubmit is in the control's *Options*, the [g-label](#)(See 19.3) is launched more often and the built-in variable A_GuiEvent will contain the word Normal for a change of the date, the number 1 for a click of a date, and the number 2 when the MonthCal releases "mouse capture". For example, if the user double-clicks a new date, the label would be launched five times: Once with Normal, twice with 1, and twice with 2. This can be used to detect double clicks by [measuring the time](#)(See 9.) between instances of the number 1.

When specifying dates in the YYYYMMDD format, the MM and/or DD portions may be omitted, in which case they are assumed to be 1. For example, 200205 is seen as 20020501, and 2005 is seen as 20050101.

Windows 95 and NT4 require DLL versions at least as new as those distributed with Internet Explorer 3.0 to support MonthCal controls.

MonthCal Options:

Multi: Multi-select. Allows the user to shift-click or click-drag to select a range of adjacent dates (the user may still select a single date too). This option may be specified explicitly or put into effect automatically by means of specifying a selection range when the control is created. For example: *Gui, Add, MonthCal, vMyCal, 20050101-20050108*. Once the control is created, this option cannot be changed.

Range: Restricts how far back or forward in time the calendar can go. After the word Range, specify the minimum and maximum dates in YYYYMMDD format (with a dash between them). For example, *Range20050101-20050615* would restrict the selection to the first 5.5 months of 2005. Either the minimum or maximum may be omitted to leave the calendar unrestricted in that direction. For example, *Range20010101* would prevent a date prior to 2001 from being selected and *Range-20091231* (leading dash) would prevent a date later than 2009 from being selected. Without the Range option, any date between the years 1601 and 9999 can be selected.

4: Specify the number 4 in *Options* to display week numbers (1-52) to the left of each row of days. Week 1 is defined as the first week that contains at least four days.

8: Specify the number 8 in *Options* to prevent the circling of today's date within the control.

16: Specify the number 16 in *Options* to prevent the display of today's date at the bottom of the control.

Description: A sliding bar that the user can move along a vertical or horizontal track. The standard volume control in the taskbar's tray is an example of a slider.

Example: Gui, Add, Slider, vMySlider, 50

Specify the starting position of the slider as the last parameter. If the last parameter is omitted, the slider starts off at 0 or the number in the allowable range that is closest to 0.

The user may slide the control by the following means: 1) dragging the bar with the mouse; 2) clicking inside the bar's track area with the mouse; 3) turning the mouse wheel while the control has focus; or 4) pressing the following keys while the control has focus: Arrow keys, Page-up, Page-down, Home, and End.

When the [Gui Submit](#)(See 19.3) command is used, the control's [associated output variable](#)(See 19.3) (if any) receives the current numeric position of the slider. The position is also stored in the output variable whenever the control's [g-label](#)(See 19.3) is launched.

If the slider has a [g-label](#)(See 19.3), by default it will be launched only when the user has stopped moving the slider (such as by releasing the mouse button after having dragging it).

However, if the word AltSubmit is in the control's *Options*, the g-label is launched for all slider events and the built-in variable A_GuiEvent will contain one of the following digits or strings:

- 0: The user pressed the Left-arrow or Up-arrow key.
- 1: The user pressed the Right-arrow or Down-arrow key.
- 2: The user pressed the Page-up key.
- 3: The user pressed the Page-down key.
- 4: The user moved the slider via the mouse wheel, or finished a drag-and-drop to a new position.
- 5: The user is currently dragging the slider via the mouse; that is, the mouse button is currently down.
- 6: The user pressed the Home key to send the slider to the left or top side.
- 7: The user pressed the End key to send the slider to the right or bottom side.

Normal: The user has finished moving the slider, either via the mouse or the keyboard. Note: With the exception of mouse wheel movement (#4), the **g-label**(See 19.3) is launched again for the "normal" event even though it was already launched for one of the digit-events above.

Slider Options:

Buddy1 and **Buddy2**: Specifies up to two existing controls to automatically reposition at the ends of the slider. Buddy1 is displayed at the left or top side (depending on whether the Vertical option is present). Buddy2 is displayed at the right or bottom side. After the word Buddy1 or Buddy2, specify the **variable name**(See 19.3) of an existing control. For example, Buddy1MyTopText would assign the control whose variable name is MyTopText. Windows 95 and NT4 require Internet Explorer 3.0 or later to support this option.

Center: The thumb (the bar moved by the user) will be blunt on both ends rather than pointed at one end.

Invert: Reverses the control so that the lower value is considered to be on the right/bottom rather than the left/top. This is typically used to make a vertical slider move in the direction of a traditional volume control. Note: The ToolTip option described below will not obey the inversion and therefore should not be used in this case.

Left: The thumb (the bar moved by the user) will point to the top rather than the bottom. But if the Vertical option is in effect, the thumb will point to the left rather than the right.

Line: Specifies the number of positions to move when the user presses one of the arrow keys. After the word Line, specify number of positions to move. For example: Line2

NoTicks: Omits tickmarks alongside the track.

Page: Specifies the number of positions to move when the user presses the Page-up or Page-down key. After the word Page, specify number of positions to move. For example: Page10

Range: Sets the range to be something other than 0 to 100. After the word Range, specify the minimum, a dash, and maximum. For example, Range1-1000 would allow a number between

1 and 1000 to be selected; Range-50-50 would allow a number between -50 and 50; and Range-10--5 would allow a number between -10 and -5.

Thick: Specifies the length of the thumb (the bar moved by the user). After the word Thick, specify the thickness in pixels (e.g. Thick30). To go beyond a certain thickness on Windows XP or later, it is probably necessary to either specify the Center option or remove the theme from the control (which can be done by specifying *-Theme* in the control's options).

TickCount: Provides tickmarks alongside the track at the specified interval. After the word TickInterval, specify the interval at which to display additional tickmarks (if the interval is omitted, it is assumed to be 1). For example, TickInterval10 would display a tickmark once every 10 positions.

ToolTip: Creates a tooltip that reports the numeric position of the slider as the user is dragging it. To have the tooltip appear in a non-default position, specify one of the following instead: ToolTipLeft or ToolTipRight (for horizontal sliders); ToolTipTop or ToolTipBottom (for vertical sliders). Windows 95 and NT4 require Internet Explorer 3.0 or later to support this option.

Vertical: Makes the control slide up and down rather than left and right.

The above options can be changed after the control is created via [GuiControl](#)(See 19.5).

Description: A dual-color bar typically used to indicate how much progress has been made toward the completion of an operation.

Example: Gui, Add, Progress, w300 h20 cBlue vMyProgress

Specify the starting position of the bar as the last parameter (if omitted, the bar starts off at 0 or the number in the allowable range that is closest to 0). To later change the position of the bar, follow these examples, all of which operate upon a progress bar whose [associated variable name](#) (See 19.3) is MyProgress:

`GuiControl`(See 19.5),, MyProgress, +20 ; Increase the current position by 20.

`GuiControl`(See 19.5),, MyProgress, 50 ; Set the current position to 50.

For horizontal Progress Bars, the thickness of the bar is equal to the control's height. For vertical Progress Bars it is equal to the control's width.

Progress Options:

Cn: Changes the bar's color. Specify for **n** one of the 16 primary HTML [color names](#)(See 19.12) or a 6-digit RGB color value. Examples: cRed, cFFFF33, cDefault. If the C option is never used (or cDefault is specified), the system's default bar color will be used.

BackgroundN: Changes the bar's background color. Specify for **n** one of the 16 primary HTML [color names](#)(See 19.12) or a 6-digit RGB color value. Examples: BackgroundGreen, BackgroundFFFF33, BackgroundDefault. If the Background option is never used (or

BackgroundDefault is specified), the background color will be that of the window or tab control behind it.

Range: Sets the range to be something other than 0 to 100. After the word Range, specify the minimum, a dash, and maximum. For example, Range0-1000 would allow a numbers between 0 and 1000; Range-50-50 would allow numbers between -50 and 50; and Range-10--5 would allow numbers between -10 and -5. On Windows 95 and NT4, negative ranges and ranges beyond 65535 will not behave correctly unless Internet Explorer 3.0 or later is installed.

-Smooth (minus Smooth): Displays a length of segments rather than a smooth continuous bar. Specifying -Smooth is also one of the requirements to show a themed progress bar on Windows XP or later. The other requirement is that the bar not have any custom colors; that is, that the C and Background options be omitted. Windows 95 and NT4 require Internet Explorer 3.0 or later to support this option.

Vertical: Makes the bar rise or fall vertically rather than move along horizontally. Windows 95 and NT4 require Internet Explorer 3.0 or later to support this option.

The above options can be changed after the control is created via [GuiControl](#)(See 19.5).

Description: A rectangular border/frame, often used around other controls to indicate they are related. In this case, the last parameter is the title of the box, which if present is displayed at its upper-left edge.

Example: Gui, Add, GroupBox, w400 h300, Geographic Criteria

By default, a GroupBox's title may have only one line of text. This can be overridden by specifying *Wrap* in Options.

Description: A large control containing multiple pages, each of which contains other controls. From this point forward, these pages are referred to as "tabs".

Tab2 vs. Tab: v1.0.47.05 adds the "Tab2" control that fixes rare redrawing problems in the original "Tab" control (e.g. activating a GUI window by clicking certain parts of its controls, such as scrollbars, might redraw improperly). The original Tab control is retained for backward compatibility because Tab2 puts its tab control after its contained controls in the tab-key navigation order. New scripts should use Tab2 whenever possible.

Example: Gui, Add, Tab2,, General|View|Appearance|Settings

The last parameter above is a pipe-delimited list of tab names. To have one of the tabs pre-selected when the window first appears, include two pipe characters after it. Alternatively, include in *Options* the word **Choose** followed immediately by the number to pre-select. For example, Choose5 would pre-select the fifth tab (as with other options, it can also be a variable

such as `Choose%Var%`). To change the selected tab, add tabs, or remove tabs after the control has been created, use [GuiControl](#)(See 19.5).

After creating a Tab control, subsequently added controls automatically belong to its first tab. This can be changed at any time by following these examples:

```
Gui, Tab ; Future controls are not part of any tab control.

Gui, Tab, 3 ; Future controls are owned by the third tab of the current tab control.

Gui, Tab, 3, 2 ; Future controls are owned by the third tab of the second tab control.

Gui, Tab, Name ; Future controls are owned by the tab whose name starts with Name (not case sensitive).

Gui, Tab, Name,, Exact ; Same as above but requires exact match (case sensitive too).
```

It is also possible to use any of the examples above to assign controls to a tab or tab-control that does not yet exist (except in the case of the *Name* method). But in that case, the relative positioning options described below are not supported.

Positioning: When each tab of a Tab control receives its first sub-control, that sub-control will have a special default position under the following conditions: 1) The X and Y coordinates are both omitted, in which case the first sub-control is positioned at the upper-left corner of the tab control's interior (with a standard [margin](#)(See 19.3)), and sub-controls beyond the first are positioned beneath the previous control; 2) The [X+n](#) and/or [Y+n](#) (See 19.3)positioning options are specified, in which case the sub-control is positioned relative to the upper-left corner of the tab control's interior. For example, specifying "x+10 y+10" would position the control 10 pixels right and 10 pixels down from the upper left corner.

Sub-controls do not necessarily need to exist within their Tab control's boundaries: they will still be hidden and shown whenever their tab is selected or de-selected. This behavior is especially appropriate for the "buttons" style described below.

When the [Gui Submit](#)(See 19.3) command is used, the control's [associated output variable](#)(See 19.3) (if any) receives the name of the currently selected tab. However, if the control has the [AltSubmit](#)(See 19.3) property, the output variable will receive the tab's position number instead (the first tab is 1, the second is 2, etc.).

A [g-label](#)(See 19.3) such as `gMySubroutine` may be listed in the control's options. This would cause the *MySubroutine* label to be launched automatically whenever the user changes to a new tab. If the tab control has both a [g-label](#)(See 19.3) and an [output variable](#)(See 19.3), whenever the user switches to a new tab, the output variable will be set to the previously selected tab name (or number in the case of [AltSubmit](#)(See 19.3)).

Keyboard navigation: The user may press Control-PageDown/PageUp to navigate from page to page in a tab control; if the keyboard focus is on a control that does not belong to a Tab control, the window's first Tab control will be navigated. Control-Tab and Control-Shift-Tab may also be used except that they will not work if the currently focused control is a multi-line Edit control.

Each window may have no more than 255 tab controls. Each tab control may have no more than 256 tabs (pages). In addition, a tab control may not contain other tab controls.

Tab Options:

Choose: See [above](#).

-Background (minus followed by the word background): Overrides the [window's custom background color](#)(See 19.3) and uses the system's default Tab control color. Specify +Theme -Background to make the Tab control conform to the current desktop theme. However, most control types will look strange inside such a Tab control because their backgrounds will not match that of the tab control. This can be fixed for some control types (such as [Text](#)) by adding BackgroundTrans to their options.

Buttons: Creates a series of buttons at the top of the control rather than a series of tabs (in this case, there will be no border by default because the display area does not typically contain controls).

Left/Right/Bottom: Specify one of these words to have the tabs on the left, right, or bottom side instead of the top. See [TCS_VERTICAL](#)(See 30.18) for limitations on Left and Right.

-Wrap: Prevents the tabs from taking up more than a single row (in which case if there are too many tabs to fit, arrow buttons are displayed to allow the user to slide more tabs into view).

Icons in Tabs: An icon may be displayed next to each tab's name/text via [SendMessage](#)(See 28.1.12). This is demonstrated in the forum topic [Icons in tabs](#).

Description: A row of text and/or icons attached to the bottom of a window, which is typically used to report changing conditions.

; Example:

```
Gui, Add, StatusBar,, Bar's starting text (omit to start off empty).
SB_SetText("There are " . RowCount . " rows selected.")
```

The simplest use of a status bar is to call [SB_SetText\(\)](#) whenever something changes that should be reported to the user. To report more than one piece of information, divide the bar into sections via [SB_SetParts\(\)](#). To display icon(s) in the bar, call [SB_SetIcon\(\)](#).

All of the following StatusBar functions operate upon the current thread's [default GUI window](#)(See 19.3) (which can be changed via [Gui, 2:Default](#)(See 19.3)). If the default window does not exist or has no status bar, all SB functions return 0 to indicate the problem.

SB_SetText(NewText [, PartNumber, Style]): Displays *NewText* in the specified part of the status bar. If *PartNumber* is omitted, it defaults to 1. Otherwise, specify an integer between 1 and 256. If *Style* is omitted, it defaults to 0, which uses a traditional border that makes that part of the bar look sunken. Otherwise, specify 1 to have no border or 2 to have border that

makes that part of the bar look raised. Finally, up to two tab characters (`t) may be present anywhere in *NewText*: anything to the right of the first tab is centered within the part, and anything to the right of the second tab is right-justified. `SB_SetText()` returns 1 upon success and 0 upon failure.

`SB_SetParts([Width1, Width2, ... Width255])`: Divides the bar into multiple sections according to the specified widths (in pixels). If all parameters are omitted, the bar is restored to having only a single, long part. Otherwise, specify the width of each part except the last (the last will fill the remaining width of the bar). For example, `SB_SetParts(50, 50)` would create three parts: the first two of width 50 and the last one of all the remaining width. Note: Any parts "deleted" by `SB_SetParts()` will start off with no text the next time they are shown (furthermore, their icons are automatically destroyed). Upon success, `SB_SetParts()` returns a non-zero value (the status bar's `HWND`(See 28.1.4)). Upon failure it returns 0.

`SB_SetIcon(Filename [, IconNumber, PartNumber])`: Displays a small icon to the left of the text in the specified part (if *PartNumber* is omitted, it defaults to 1). *Filename* is the name of an icon (.ICO), cursor (.CUR), or animated cursor (.ANI) file (animated cursors will not actually be animated in the bar). Other sources of icons include the following types of files: EXE, DLL, CPL, SCR, and other types that contain icon resources. To use an icon group other than the first one in the file, specify its number for *IconNumber*. For example, `SB_SetIcon("Shell32.dll", 2)` would use the default icon from the second icon group. `SB_SetIcon()` returns the icon's HICON upon success and 0 upon failure. The HICON is a system resource that can be safely ignored by most scripts because it is destroyed automatically when the status bar's window is destroyed. Similarly, any old icon is destroyed when `SB_SetIcon()` replaces it with a new one. This can be avoided via:

```
Gui +LastFound

SendMessage(See 28.1.12), 0x40F, part_number - 1, my_hIcon,
msctls_statusbar321 ; 0x40F is SB_SETICON.
```

`SB_SetProgress()`: Creates and controls a progress bar inside the status bar. This function is available at www.autohotkey.com/forum/topic37754.html

G-Label Notifications: A `g-label`(See 19.3) such as `gMySubroutine` may be listed in the control's options. This would cause the `MySubroutine` label to be launched automatically whenever the user clicks on the bar. This subroutine may consult the built-in variables `A_Gui`(See 9.) and `A_GuiControl`(See 9.). More importantly, it may consult **`A_GuiEvent`**, which contains one of the following strings (for compatibility with future versions, a script should not assume these are the only possible values):

- **Normal**: The user left-clicked the bar. The variable `A_EventInfo` contains the part number (however, the part number might be a very large integer if the user clicks near the sizing grip at the right side of the bar).
- **RightClick**: The user right-clicked the bar. The variable `A_EventInfo` contains the part number. NOTE: `GuiContextMenu`(See 19.3) will not be called for the status bar if it has a g-label. Also, the g-label's RightClick event should be used instead of

[GuiContextMenu](#)(See 19.3) when the script needs to know which part number the user clicked on (A_EventInfo).

- **DoubleClick:** The user double-clicked the bar. The variable A_EventInfo contains the part number.
- **R:** The user *double-right-clicked* the bar. The variable A_EventInfo contains the part number.

Font and Color: Although the font size, face, and style can be set via "[Gui Font](#)(See 19.3)" (just like normal controls), the text color cannot be changed. Also, "[Gui Color](#)(See 19.3)" is not obeyed; instead, the status bar's background color may be changed by specifying in *Options* the word [Background](#) followed immediately by a color name (see [color chart](#)(See 19.12)) or RGB value (the 0x prefix is optional). Examples: BackgroundSilver, BackgroundFFDD99, BackgroundDefault.

Hiding the StatusBar: Upon creation, the bar can be hidden via *Gui, Add, StatusBar, Hidden vMyStatusBar*. To hide it sometime after creation, use *GuiControl, Hide, MyStatusBar*. To show it, use *GuiControl, Show, MyStatusBar*. Note: Hiding the bar does not reduce the height of the window. If that is desired, one easy way is [Gui, Show, AutoSize](#)(See 19.3).

Styles (rarely used): See the [StatusBar styles table](#)(See 30.18).

Known Limitations: 1) Any control that overlaps the status bar might sometimes get drawn on top of it. One way to avoid this is to dynamically shrink such controls via the [GuiSize label](#)(See 19.3). 2) There is a limit of one status bar per window.

Example: The bottom of the [TreeView page](#)(See 19.8) demonstrates a multipart status bar.

An MSIE browser control can be embedded into a GUI window via DllCall. The control can display a web page or other browser-compatible content. This is demonstrated at www.autohotkey.com/wiki/index.php?title=

[ListView](#)(See 19.7), [TreeView](#)(See 19.8), [Gui](#)(See 19.3), [GuiControl](#)(See 19.5), [GuiControlGet](#)(See 19.6), [Menu](#)(See 22.13)

19.5 GuiControl

对一个图形用户界面窗口的控件做出种种改变。

GuiControl, Sub-command, ControlID [, Param3]

参数

Sub-command	参见下面的列表。
-------------	----------

<p>ControlID</p> <p>如果目标控件有一个关联的变量，那么将变量名指定为 <i>ControlID</i> (这个方法优先于后面描述的方法)。由于这个原因，通常最好给每一个之后要被 GuiControl 或者 GuiControlGet 存取的控件指定一个变量名，即使那个控件没有输入功能(例如 GroupBox 或者 Text)。</p> <p>如若不然，<i>ControlID</i> 可以是 ClassNN (控件的类名和实例号) 或者控件的名字/文本，这两者都可以通过 Window Spy(窗体侦探) 来决定。当使用名字/文本时，匹配的行为可以由 TitleMatchMode(See 28.8) 决定。注意：一个图片控件的文件名字 (和控件创建时指定的一样) 可以当做 <i>ControlID</i> 使用。</p>	<p>Param3 除了下面 sub-commands 列表中说明的，这个参数是被省略的。</p>
--	---

ErrorLevel

如果指定的窗体/控件不存在或者其他一些错误导致命令不能工作，**ErrorLevel**(See 30.8) 被设为 1 。否则，它被设为 0 。

Sub-commands

(Blank) (空白): 将 *Sub-command* 留空，然后通过 *Param3* 给予控件新的内容。比如：

Picture(See 19.4): *Param3* 应当是要加载的图片的文件名 (支持的文件类型参见 **Gui Picture**(See 19.4))。在文件名的前面可以指定零个或多个下面的参数: *wN (宽度 N), *hN (高度 N), 和 *IconN (DLL 或者 EXE 文件中的图标的组号)。下面的例子中，第二个图标组中的默认的图标载入的宽度是 100，通过 "keep aspect ratio" 设置自动高度: *GuiControl,, MyPic, *icon2 *w100 *h-1 C:\My Application.exe*。指定 *w0 *h0 来使用图像的实际高度和宽度。如果 *w 和 *h 被省略，图像将被缩放来适应当前控件的尺寸。当从多图标的 .ICO 文件载入图标的时候，指定宽度和高度可以决定哪个图标将被载入。注意：在最后一个选择项和文件名之间只能使用一个空格或者制表符；其他任何空格或者制表符将被当做文件名的一部分。

Text(See 19.4)/**Button**(See 19.4)/**GroupBox**(See 19.4)/**StatusBar**(See 19.4): 对于 *Param3* 指定控件的新文本。因为控件不能自动改变大小，所以如果控件需要变宽，请使用 **GuiControl, Move, MyText, W300**。对于 **StatusBar**(See 19.4)，这个选项仅设置第一部分的文本。(想要更灵活的实现请使用 **SB_SetText()**(See 19.4))。

Edit(See 19.4): 为了正常显示 *Param3* 中缺少之前回车 (`r) 的换行 (`n)，它被自动的转换为 CR+LF (`r` n)。然而，这在通常情况下无需考虑，因为 "Gui Submit" 和 "GuiControlGet OutputVar" 命令能自动撤销这一替换，使 CR+LF 变回 LF (`n)。

Hotkey(See 19.4): *Param3* 可以为空来清除控件，或者一组有键名的修改器。例如: ^!c, ^Numpad1, +Home。支持的修改器仅有 ^ (Control), ! (Alt)，和 + (Shift)。可用键名请参见 **key list**(See 7.)。

Checkbox(See 19.4): *Param3* 可以为 0 来不选择按钮，1 来选择，或者 -1 来给它一个灰色的检查标志。另外，*Param3* 默认是控件的新标题/文本。关于如何重写这一行为，参考下面的 **Text** 。

Radio(See 19.4): 和上面的 **Checkbox** 类似。但是，如果单选按钮是选中的（开启）而且它是一个单选按钮组中的一项，那么组中其他的单选按钮自动即为未选中的。想要选中一个只有一个变量的单选按钮组中的一个新的按钮并且此按钮不是那个变量直接联系的，请指定按钮的 **ControlID** 为它的 名字/文本。

DateTime(See 19.4)/**MonthCal**(See 19.4): 为 **Param3** 指定一个格式为 **YYYYMMDDHH24MISS**(See 16.26) 的日期-时间标记。指定 **%A_Now%** 来使用当前的日期和时间（当天）。对于日期时间控件，**Param3** 可以省略来使控件不选中日期/时间（如果它在创建时有 **that ability**(See 19.4) (这一能力)）。对于 **MonthCal** 控件，如果它是 **multi-select**(See 19.4) (多选择的)，那么可以指定一个范围。

UpDown(See 19.4)/**Slider**(See 19.4)/**Progress**(See 19.4): **Param3** 应当是控件的新位置。如果 **Param3** 的第一个符号是加号，这个数字就被认为是相对当前位置的偏移值，例如，**+10** 把位置增加 10，而 **+ -10** (加负十) 把位置减去 10。如果新位置在控件的范围之外，控件通常被设置在最近的合法值处。

Tab(See 19.4)/**DropDownList**(See 19.4)/**ComboBox**(See 19.4)/**ListBox**(See 19.4): **Param3** 应当包含一个管道符分隔的要加在控件后的条目的列表。要替换（重写）这个列表，在第一个字符前加上一个管道符（例如 **|Red|Green|Blue**）。要将控件置空，只需指定一个管道符（**|**）。要预先选定其中一个条目，在它之后加上两个管道符（例如 **Red|Green||Blue**）。条目之间的分隔符可以为其他字符，不一定是管道符。比如 **Gui +Delimiter`n**(See 19.3) 可以将它改为换行而 **Gui +DelimiterTab** 将它改为制表符。

Tab controls(See 19.4): 除了以上段落描述的行为，一个标签页的子控件和它们原始的标签编号相关联；也就是说，它们永远不与它们的标签页的名字关联。因此，重命名或者删除一个标签页不会改变子控件所属的标签页的编号。例如，有三个标签页 "**Red|Green|Blue**"，第二个标签页通过 "**GuiControl,, MyTab, |Red|Blue**" 被移除，原来和 **Green** 标签页联系的子控件不会和 **Blue** 标签页关联。因为这一行为，通常情况下只有最后一个标签页才能被移除。用这种方法移除的标签随后还可以再被加进来，那时它们会重新声明原来的控件。

ListView(See 19.7) 和 **TreeView**(See 19.8): 当 **Sub-command** 为空时不支持这两个参数。做为替代，请使用内建的 **ListView functions**(See 19.7) 和 **TreeView functions**(See 19.8)。

GuiControl, Text: 除了下面这几个，效果和上面的一样：

Checkbox(See 19.4)/**Radio**(See 19.4): **Param3** 被当做新的文本/标题，即使它是 **-1, 0, 或 1**。

DateTime(See 19.4): **Param3** 被当做控件中显示的新的 **date/time format**(See 19.4) (日期/时间格式)。如果 **Param3** 被省略，任何自定义格式将被移除，短日期格式被使用。

ComboBox(See 19.4): **Param3** 被当做文本而直接放在下拉列表框的编辑控件中。

GuiControl, Move: 移动 和/或 缩放控件。在 **Param3** 中指定下面一个或几个可选字母：X (相对于窗口中用户区域的 x 坐标，即不包括标题栏，菜单栏和边框的区域); Y (y 坐标), W (宽度), H (高度)。例如：

```
GuiControl, Move, MyEdit, x10 y20 w200 h100
```

```
GuiControl, Move, MyEdit, % "x" VarX+10 "y" VarY+5 "w" VarW*2 "h" VarH*1.5 ; 通  
过 "% " 前缀使用 expression(See 9.) (表达式) 。
```

GuiControl, MoveDraw: 除了上面 "Move" 的功能，它还重画控件占据的窗口位置。尽管当重复和快速调用时可能会产生闪烁，它解决了在特定控件中绘画的问题，比如在 **GroupBoxes**(See 19.4) 中。

GuiControl, Focus: 将键盘的焦点设置在控件上。要使有效，窗口通常不能被最小化或者被隐藏。

GuiControl, Enable / Disable: 使控件有效或者失效(灰化)。对于标签页控件，此选项还会使标签的子控件有效或失效。然而，任何通过 "GuiControl Disable" 显示禁止的控件在它的标签页被重新生效后仍然保持失效状态。单词 `Disable` 或者 `Enable` 后面可以可选的紧跟着 `Zero` 或 `One`。`Zero` 产生相反的效果。例如，`Enable` 和 `Enable%VarContainingOne%` 都使控件生效，但是 `Enable%VarContainingZero%` 将使它失效。

GuiControl, Hide / Show: 隐藏或者显示控件。对于标签页控件将会隐藏或显示标签下所有的子控件。另外如果你还想同时禁用控件的快捷键(加下线字母)，通过 "GuiControl Disable" 禁用控件。单词 `Hide` 或者 `Show` 后面可以可选的紧跟着 `0 or 1`。产生相反的效果。例如，`Show` 和 `Show%VarContainingOne%` 都显示控件，但是 `Show%VarContainingZero%` 将隐藏它。

GuiControl, Delete (还没有实现): 这个子命令现在还不存在。对于一个工作区，使用上面的 `Hide` 和 /或 `Disable`，或者通过 [Gui Destroy\(See 19.3\)](#) 销毁或重建整个窗口。

GuiControl, Choose, ControlID, N: 设置列表框，下拉列表，复选框或者标签页中的选区为第 `N` 个入口。第一个入口 `N` 对应为 `1`，第二个入口对应为 `2`，依次类推(如果 `N` 不是整数，下面描述的 `ChooseString` 方法将被作为替代使用)。与 [Control Choose\(See 28.1.1\)](#) 不同，这个子命令不会触发任何与控件关联的 [g-label\(See 19.3\)](#)，除非 `N` 之前有一个管道符(即使这样，`g-label` 仅在新的选区和旧选区不同的情况下被触发，至少对 [Tab controls\(See 19.4\)](#) 是这样)。例如：`GuiControl, Choose, MyListBox, |3.`

此外要产生一个结束的事件，(例如在列表框中的双击)，请包含两个管道符而不是一个(不支持标签页)。

要选择或者取消选择 [multi-select ListBox\(See 19.4\)](#)(多选列表框)中的所有 选项，参考下面的例子：

`Gui +LastFound ; 避免指定下面的 WinTitle 的需要。`

`PostMessage(See 28.1.12), 0x185, 1, -1, ListBox1 ; 选择所有选项。0x185 表示 LB_SETSEL。`

`PostMessage(See 28.1.12), 0x185, 0, -1, ListBox1 ; 取消选择所有选项。`

GuiControl, ChooseString, ControlID, String: 设置列表框，下拉列表，复选框或者标签页中主要部分和 `String` 匹配的选区(选择)为入口。搜索不区分大小写。例如，如果一个控件包含项 "UNIX Text"，指定单词 `unix`(小写)就足以选中它了。支持管道符和双管道符(详细请参考上面的 "Choose")。

GuiControl, Font: 把控件的字体改变为它所在窗口的当前使用的字体，字号，颜色和样式。例如：

`Gui, Font(See 19.3), s18 cRed Bold, Verdana ; 如果需要，使用类似这样的排列方式给窗口指定一个新的默认字体。`

`GuiControl, Font, MyEdit ; 使上面的字体对控件生效。`

GuiControl, +/-Option1 +/-Option2 ... : 增加或者删除不同的选项或样式。所有的 GUI 选项([control-specific\(See 19.4\)](#) 和 [general\(See 19.3\)](#))都将被识别。下面的例子中，[AltSubmit\(See 19.3\)](#) 选项被启用但是控件的 [g-label\(See 19.3\)](#) 被删除：`GuiControl, +AltSubmit -g, MyListBox`

下个例子中，OK 按钮被设为新的默认按钮：

`GuiControl, +Default, OK`

尽管 [styles](#)(See 30.18) (样式) 和 [extended styles](#) (扩展的样式) 也会被识别，它们中一些在控件创建以后不能被应用或者被删除。如果被指定的改变中至少一个被成功应用，`ErrorLevel` 被设为 0 。否则，它被设为 1 来指示没有应用任何改变。即使一个改变被成功应用，控件仍然可能选择忽视它。

注意

想要根据窗口号码而不是默认窗口进行操作(参加下面)，在子命令前面加上数字和一个冒号，如下面的例子所示：

```
GuiControl, 2:Show, MyButton
GuiControl, 2:, MyListBox, Item1|Item2
```

当任何作为 GUI 操作结果的线程被启动时，一个 GUI [thread](#)(See 30.19) (线程) 被定义。GUI 操作包括选择 GUI 窗口的菜单栏的一项，或者触发它的 [g-labels](#)(See 19.3) 中的一项(比如按下一个按钮)。

GUI 线程的默认窗口号码是启动线程的窗口号码。非 GUI 线程使用 1 做为它们的默认窗口号码。

相关命令

[Gui](#)(See 19.3), [GuiControlGet](#)(See 19.6), [Control](#)(See 28.1.1)

示例

```
GuiControl,, MyListBox, |Red|Green|Blue ; 用新列表替换当前列表。
GuiControl,, MyEdit, New text line 1.`nNew text line 2.
GuiControl,, MyRadio2, 1 ; 选择这一单选按钮，不选择组中其他的。
GuiControl, Move, OK, x100 y200 ; 移动 OK 按钮到一个新位置。
GuiControl, Focus, LastName ; 把键盘焦点设置在变量或者文本为 "LastName" 的控件上。
```

19.6 GuiControlGet

Retrieves various types of information about a control in a GUI window.

`GuiControlGet, OutputVar [, Sub-command, ControlID, Param4]`

参数

OutputVar	The name of the variable in which to store the result. If the command cannot complete (see <code>ErrorLevel</code> below), this variable is made blank.
Sub-command	See list below.
ControlID	If blank or omitted, it behaves as though the name of the output variable was specified. For example, <code>GuiControlGet, MyEdit</code> is the same as <code>GuiControlGet, MyEdit,, MyEdit</code> .

	<p>If the target control has an associated variable, specify the variable's name as the <i>ControlID</i> (this method takes precedence over the ones described next). For this reason, it is usually best to assign a variable to any control that will later be accessed via GuiControl or GuiControlGet, even if that control is not input-capable (such as GroupBox or Text).</p> <p>Otherwise, <i>ControlID</i> can be either ClassNN (the classname and instance number of the control) or the name/text of the control, both of which can be determined via Window Spy. When using name/text, the matching behavior is determined by SetTitleMatchMode(See 28.8). Note: a picture control's file name (as it was specified at the time the control was created) may be used as its <i>ControlID</i>.</p>
Param4	This parameter is omitted except where noted in the list of sub-commands below.

ErrorLevel

[ErrorLevel](#)(See 30.8) is set to 1 if the specified window/control does not exist or some other problem prevented the command from working. Otherwise, it is set to 0.

Sub-commands

(Blank): Leave *Sub-command* blank to retrieve the control's contents. All control types are self-explanatory except the following:

[Picture](#)(See 19.4): Retrieves the picture's file name as it was originally specified when the control was created. This name does not change even if a new picture file name is specified.

[Edit](#)(See 19.4): Retrieves the contents but any line breaks in the text will be represented as plain linefeeds (`n) rather than the traditional CR+LF (`r`n) used by non-GUI commands such as [ControlGetText](#)(See 28.1.7) and [ControlSetText](#)(See 28.1.10).

[Hotkey](#)(See 19.4): Retrieves a blank value if there is no hotkey in the control. Otherwise it retrieves the modifiers and key name. Examples: ^!C, ^Home, +^NumpadHome.

[Checkbox](#)(See 19.4)/[Radio](#)(See 19.4): Retrieves 1 if the control is checked, 0 if it is unchecked, or -1 if it has a gray checkmark. To retrieve the control's text/caption instead, specify the word Text for *Param4*. Note: Unlike the [Gui Submit](#)(See 19.3) command, radio buttons are always retrieved individually, regardless of whether they are in a radio group.

[UpDown](#)(See 19.4)/[Slider](#)(See 19.4)/[Progress](#)(See 19.4): Retrieves the control's current position.

[Tab](#)(See 19.4)/[DropDownList](#)(See 19.4)/[ComboBox](#)(See 19.4)/[ListBox](#)(See 19.4): Retrieves the text of the currently selected item/tab (or its position if the control has the [AltSubmit](#)(See 19.3) property). For a ComboBox, if there is no selected item, the text in the control's edit field

is retrieved instead. For a [multi-select ListBox](#)(See 19.4), the output uses the window's [current delimiter](#)(See 19.3).

[ListView](#)(See 19.7) and [TreeView](#)(See 19.8): These are not supported when *Sub-command* is blank. Instead, use the built-in [ListView functions](#)(See 19.7) and [TreeView functions](#)(See 19.8).

[StatusBar](#)(See 19.4): Retrieves only the first part's text.

Note: To unconditionally retrieve any control's text/caption rather than its contents, specify the word *Text* for *Param4*.

GuiControlGet, OutputVar, Pos: Retrieves the position and size of the control. The position is relative to the GUI window's client area, which is the area not including title bar, menu bar, and borders. The information is stored in four variables whose names all start with *OutputVar*. For example:

```
GuiControlGet, MyEdit, Pos
```

```
MsgBox The X coordinate is %MyEditX%. The Y coordinate is %MyEditY%. The width  
is %MyEditW%. The height is %MyEditH%.
```

Within a [function](#)(See 10.), to create a set of variables that is global instead of local, [declare](#)(See 10.) *OutputVar* as a global variable prior to using this command (the converse is true for [assume-global](#)(See 10.) functions).

GuiControlGet, OutputVar, Focus: Retrieves the control identifier (ClassNN) for the control that currently has keyboard focus. Since the specified GUI window must be [active](#)(See 28.12) for one of its controls to have focus, *OutputVar* will be made blank if it is not active. Example usage: GuiControlGet, focused_control, focus

GuiControlGet, OutputVar, FocusV [v1.0.43.06+]: Same as *Focus* (above) except that it retrieves the name of the focused control's [associated variable](#)(See 19.3). If that control lacks an associated variable, the first 63 characters of the control's text/caption is retrieved instead (this is most often used to avoid giving each button a variable name).

GuiControlGet, OutputVar, Enabled: Retrieves 1 if the control is enabled or 0 if it is disabled.

GuiControlGet, OutputVar, Visible: Retrieves 1 if the control is visible or 0 if it is hidden.

GuiControlGet, OutputVar, Hwnd [v1.0.46.16+]: Retrieves the window handle (HWND) of the control. A control's HWND is often used with [PostMessage](#)(See 28.1.12), [SendMessage](#)(See 28.1.12), and [DII Call](#)(See 18.3). Note: [HwndOutputVar](#)(See 19.3) is usually a more concise way to get the HWND.

注意

To operate upon a window number other than the default (see below), include a number followed by a colon in front of the sub-command as in these examples:

```
GuiControlGet, MyEdit, 2:  
GuiControlGet, MyEdit, 2:Pos  
GuiControlGet, Outputvar, 2:Focus
```

A GUI [thread](#)(See 30.19) is defined as any thread launched as a result of a GUI action. GUI actions include selecting an item from a GUI window's menu bar, or triggering one of its [g-labels](#)(See 19.3) (such as by pressing a button).

The default window number for a GUI thread is that of the window that launched the thread. Non-GUI threads use 1 as their default.

相关命令

[Gui](#)(See 19.3), [GuiControl](#)(See 19.5), [ControlGet](#)(See 28.1.4)

示例

```
GuiControlGet, MyEdit  
  
GuiControlGet, CtrlContents,, MyEdit ; Same as the above except uses a non-default  
output variable.  
  
GuiControlGet, MyCheckbox1 ; Retrieves 1 if it is checked, 0 if it is unchecked.  
  
GuiControlGet, MyCheckbox1,,, Text ; Retrieves the caption/text of the checkbox.  
  
GuiControlGet, Pic, Pos, Static4 ; The position/size will be stored in PicX, PicY, PicW, and  
PicH
```

19.7 Gui ListView control

- [Introduction and Simple Example](#)
- [Options and Styles](#)
- [View Modes](#): Report (default), Icon, Tile, Small-Icon, and List.
- [Built-in Functions](#):
 - [Row functions](#) (adding, modifying, and deleting rows)
 - [Column functions](#)
 - [Getting data out of a ListView](#)
- [G-Label Notifications](#)
- [ImageLists](#) (the means by which icons are added to a ListView)
- [ListView Remarks](#)

- Examples

A List-View is one of the most elaborate controls provided by the operating system. In its most recognizable form, it displays a tabular view of rows and columns, the most common example of which is Explorer's list of files and folders (detail view).

Though it may be elaborate, a ListView's basic features are easy to use. The syntax for creating a ListView is:

Gui, Add, ListView, Options, ColumnTitle1|ColumnTitle2|...

Here is a working script that creates and displays a ListView containing a list of files in the user's "My Documents" folder:

```
; Create the ListView with two columns, Name and Size:  
Gui, Add, ListView, r20 w700 gMyListView, Name|Size (KB)  
  
; Gather a list of file names from a folder and put them into the ListView:  
Loop, %A_MyDocuments%\*.*  
    LV_Add("", A_LoopFileName, A_LoopFileSizeKB)  
  
LV_ModifyCol() ; Auto-size each column to fit its contents.  
LV_ModifyCol(2, "Integer") ; For sorting purposes, indicate that column 2 is an integer.  
  
; Display the window and return. The script will be notified whenever the user double-clicks a row.  
Gui, Show  
return  
  
MyListView:  
if A_GuiEvent = DoubleClick  
{  
    LV_GetText(RowText, A_EventInfo) ; Get the text from the row's first field.  
    ToolTip You double-clicked row number %A_EventInfo%. Text: "%RowText%"}
```

```

}

return

GuiClose: ; Indicate that the script should exit automatically when the window is
closed.

ExitApp

```

AltSubmit: Notifies the script for more types of ListView events than normal. In other words, the g-label is launched more often. See [ListView Notifications](#) for details.

Background: Specify the word Background followed immediately by a color name (see [color chart](#)(See 19.12)) or RGB value (the 0x prefix is optional). Examples: BackgroundSilver, BackgroundFFDD99. If this option is not present, the ListView initially defaults to the background color set by the last parameter of [Gui Color](#)(See 19.3) (or if none, the system's default background color). Specifying BackgroundDefault applies the system's default background color (usually white). For example, a ListView can be restored to the default color via *GuiControl*, +BackgroundDefault, *MyListView*.

C: Text color. Specify the letter C followed immediately by a color name (see [color chart](#)(See 19.12)) or RGB value (the 0x prefix is optional). Examples: cRed, cFF2211, c0xFF2211, cDefault

Checked: Provides a checkbox at the left side of each row. When [adding](#) a row, specify the word *Check* in its options to have the box to start off checked instead of unchecked. The user may either click the checkbox or press the spacebar to check or uncheck a row.

Count: Specify the word Count followed immediately by the total number of rows that the ListView will ultimately contain. This is not a limit: rows beyond the count can still be added. Instead, this option serves as a hint to the control that allows it to allocate memory only once rather than each time a row is added, which greatly improves row-adding performance (it may also improve sorting performance). To improve performance even more, use *GuiControl*, -Redraw, *MyListView* prior to adding a large number of rows. Afterward, use *GuiControl*, +Redraw, *MyListView* to re-enable redrawing (which also repaints the control).

Grid: Provides horizontal and vertical lines to visually indicate the boundaries between rows and columns.

Hdr: Specify -Hdr (minus Hdr) to omit the special top row that contains column titles. To make it visible later, use *GuiControl*, +Hdr, *MyListView*.

LV: Specify the string LV followed immediately by the number of an [extended ListView style](#)(See 30.18). These styles are entirely separate from generic extended styles. For example,

specifying **-E0x200** would remove the generic extended style WS_EX_CLIENTEDGE to eliminate the control's default border. By contrast, specifying **-LV0x20** would remove **LVS_EX_FULLROWSELECT**.

LV0x10: Specify **-LV0x10** to prevent the user from dragging column headers to the left or right to reorder them. However, it is usually not necessary to do this because the physical reordering of columns does not affect the column order seen by the script. For example, the first column will always be column 1 from the script's point of view, even if the user has physically moved it to the right of other columns.

LV0x20: Specify **-LV0x20** to require that a row be clicked at its first field to select it (normally, a click on *any* field will select it). The advantage of this is that it makes it easier for the user to drag a rectangle around a group of rows to select them.

Multi: Specify **-Multi** (minus **Multi**) to prevent the user from selecting more than one row at a time.

NoSortHdr: Prevents the header from being clickable. It will take on a flat appearance rather than its normal button-like appearance. Unlike most other ListView styles, this one cannot be changed after the ListView is created.

NoSort: Turns off the automatic sorting that occurs when the user clicks a column header. However, the header will still behave visually like a button (unless **NoSortHdr** has been specified). In addition, the g-label will still receive the [ColClick notification](#), to which it can respond with a custom sort or other action.

ReadOnly: Specify **-ReadOnly** (minus **ReadOnly**) to allow editing of the text in the first column of each row. To edit a row, select it then press the [F2 key](#). Alternatively, you can click a row once to select it, wait at least half a second, then click the same row again to edit it.

R: Rows of height (upon creation). Specify the letter R followed immediately by the number of rows for which to make room inside the control. For example, R10 would make the control 10 rows tall. If the ListView is created with a [view mode](#) other than report view, the control is sized to fit rows of icons instead of rows of text. Note: adding [icons](#) to a ListView's rows will increase the height of each row, which will make this option inaccurate.

Sort: The control is kept alphabetically sorted according to the contents of the first column.

SortDesc: Same as above except in descending order.

WantF2 [v1.0.44+]: Specify **-WantF2** (minus **WantF2**) to prevent an F2 keystroke from [editing](#) the currently focused row. This setting is ignored unless **-ReadOnly** is also in effect. Regardless of this setting, the g-label still receives F2 [notifications](#).

(Unnamed numeric styles): Since styles other than the above are rarely used, they do not have names. See the [ListView styles table](#)(See 30.18) for a list.

A ListView has five viewing modes, of which the most common is report view (which is the default). To use one of the other views, specify its name in the options list. The view can also be changed after the control is created; for example: `GuiControl, +IconSmall, MyListView`

Icon: Shows a large-icon view. In this view and all the others except *Report*, the text in columns other than the first is not visible. To display icons in this mode, the ListView must have a large-icon [ImageList](#) assigned to it.

Tile: Shows a large-icon view but with ergonomic differences such as displaying each item's text to the right of the icon rather than underneath it. [Checkboxes](#) do not function in this view. Also, attempting to show this view on operating systems older than Windows XP has no effect.

IconSmall: Shows a small-icon view.

List: Shows a small-icon view in list format, which displays the icons in columns. The number of columns depends on the width of the control and the width of the widest text item in it.

Report: Switches back to report view, which is the initial default. For example: `GuiControl, +Report, MyListView`

All of the ListView functions operate upon the current thread's [default GUI window](#)(See 19.3) (which can be changed via [Gui, 2:Default](#)(See 19.3)). If the default window does not exist or has no ListView controls, all functions return zero to indicate the problem.

If the window has more than one ListView control, by default the functions operate upon the one most recently added. To change this, specify `Gui, ListView, ListViewName`, where *ListViewName* is the name of the ListView's [associated variable](#)(See 19.3) or its ClassNN as shown by Window Spy. Once changed, all existing and future [threads](#)(See 30.19) will use the indicated ListView.

When the phrase "row number" is used on this page, it refers to a row's current position within the ListView. The top row is 1, the second row is 2, and so on. After a row is added, its row number tends to change due to sorting, deleting, and inserting of other rows. Therefore, to locate specific row(s) based on their contents, it is usually best to use [LV_GetText\(\)](#) in a loop.

LV_Add([Options, Field1, Field2, ...]): Adds a new row to the bottom of the list. The parameters *Field1* and beyond are the columns of the new row, which can be text or numeric (including numeric [expression](#)(See 9.) results). To make any field blank, specify "" or the equivalent. If there are too few fields to fill all the columns, the columns at the end are left blank. If there are too many fields, the fields at the end are completely ignored.

Upon failure, `LV_Add()` returns 0. Upon success, it returns the new [row number](#), which is not necessarily the last row if the ListView has the [Sort](#) or [SortDesc](#) style.

Row Options: The *Options* parameter is a string containing zero or more words from the list below (not case sensitive). Separate each word from the next with a space or tab. To remove an option, precede it with a minus sign. To add an option, a plus sign is permitted but not required.

Check: Shows a checkmark in the row (if the ListView has [checkboxes](#)). To later uncheck it, use `LV_Modify(LineNumber, "-Check")`.

Col: Specify the word *Col* followed immediately by the column number at which to begin applying the parameters *Col1* and beyond. This is most commonly used with [LV_Modify\(\)](#) to alter individual fields in a row without affecting those that lie to their left.

Focus: Sets keyboard focus to the row (often used in conjunction with Select). To later de-focus it, use `LV_Modify(LineNumber, "-Focus")`.

Icon: Specify the word *Icon* followed immediately by the number of this row's icon, which is displayed in the left side of the first column. If this option is absent, the first icon in the [ImageList](#) is used. To display a blank icon, specify a number that is larger than the number of icons in the ImageList. If the control lacks a small-icon ImageList, no icon is displayed nor is any space reserved for one in [report view](#).

Select: Selects the row. To later deselect it, use `LV_Modify(LineNumber, "-Select")`. When selecting rows, it is usually best to ensure that at least one row always has the [focus property](#) because that allows the Apps key to display its [context menu](#)(See 19.3) (if any) near the focused row. The word *Select* may optionally be followed immediately by a 0 or 1 to indicate the starting state. In other words, both "*Select*" and "*Select*".[VarContainingOne](#) are the same (the period used here is the [concatenation operator](#)(See 9.)). This technique also works with *Focus* and *Check* above.

Vis [1.0.44+]: Ensures that the specified row is completely visible by scrolling the ListView, if necessary. This has an effect only for `LV_Modify()`; for example: `LV_Modify(LineNumber, "Vis")`

LV_Insert(LineNumber [, Options, Col1, Col2, ...]): Behaves identically to `LV_Add()` except for its different first parameter, which specifies the row number for the newly inserted row. Any rows at or beneath *LineNumber* are shifted downward to make room for the new row. If *LineNumber* is greater than the number of rows in the list (even as high as 2147483647), the new row is added to the end of the list. For *Options*, see [row options](#).

LV_Modify(LineNumber, Options [, NewCol1, NewCol2, ...]): Modifies the attributes and/or text of a row, and returns 1 upon success and 0 upon failure. If *LineNumber* is 0, [all](#) rows in the control are modified (in this case the function returns 1 on complete success and 0 if any part of the operation failed). When only the first two parameters are present, only the row's attributes and not its text are changed. Similarly, if there are too few parameters to cover all the columns, the columns at the end are not changed. The [ColN option](#) may be used to update specific columns without affecting the others. For other options, see [row options](#).

LV_Delete([RowNumber]): If the parameter is omitted, **all** rows in the ListView are deleted. Otherwise, only the specified *RowNumber* is deleted. It returns 1 upon success and 0 upon failure.

LV_ModifyCol([ColumnNumber, Options, ColumnTitle]): Modifies the attributes and/or text of the specified column and its header. The first column is number 1 (not 0). If all parameters are omitted, the width of every column is adjusted to fit the contents of the rows. If only the first parameter is present, only the specified column is auto-sized. Auto-sizing has no effect when not in Report (Details) view. This function returns 1 upon success and 0 upon failure.

Column Options: The *Options* parameter is a string containing zero or more words from the list below (not case sensitive). Separate each word from the next with a space or tab. To remove an option, precede it with a minus sign. To add an option, a plus sign is permitted but not required.

Column Options: General

N: Specify for N the new width of the column, in pixels. This number can be unquoted if is the only option. For example, the following are both valid: LV_ModifyCol(1, 50) LV_ModifyCol(1, "50 Integer").

Auto: Adjusts the column's width to fit its contents. This has no effect when not in Report (Details) view.

AutoHdr: Adjusts the column's width to fit its contents and the column's header text, whichever is wider. If applied to the last column, it will be made at least as wide as all the remaining space in the ListView. It is usually best to apply this setting only after the rows have been added because that allows any newly-arrived vertical scroll bar to be taken into account when sizing the last column. This option has no effect when not in Report (Details) view.

Icon: Specify the word Icon followed immediately by the number of the [ImageList's](#) icon to display next to the column header's text. Specify -Icon (minus icon) to remove any existing icon. On Windows 95/NT4, column icons require the DLLs distributed with Internet Explorer 3.0 or later.

IconRight: Puts the icon on the right side of the column rather than the left.

Column Options: Data Type

Float: For sorting purposes, indicates that this column contains floating point numbers (hexadecimal format is not supported). Sorting performance for Float and Text columns is up to 25 times slower than it is for integers.

Integer: For sorting purposes, indicates that this column contains integers. To be sorted properly, each integer must be 32-bit; that is, within the range -2147483648 to 2147483647. If any of the values are not integers, they will be considered zero when sorting (unless they

start with a number, in which case that number is used). Numbers may appear in either decimal or hexadecimal format (e.g. 0xF9E0).

Text: Changes the column back to text-mode sorting, which is the initial default for every column. Only the first 8190 characters of text are significant for sorting purposes (except for the *Logical* option, in which case the limit is 4094).

Column Options: Alignment / Justification

Center: Centers the text in the column. To center an Integer or Float column, specify the word Center after the word Integer or Float.

Left: Left-justifies the column's text, which is the initial default for every column. On older operating systems, the first column might have a forced left-justification.

Right: Right-justifies the column's text. This attribute need not be specified for Integer and Float columns because they are right-justified by default. That default can be overridden by specifying something such as "Integer Left" or "Float Center".

Column Options: Sorting

Case: The sorting of the column is case sensitive (affects only `text` columns). If the options *Case*, *CaseLocale*, and *Logical* are all omitted, the uppercase letters A-Z are considered identical to their lowercase counterparts for the purpose of the sort.

CaseLocale [v1.0.43.03+]: The sorting of the column is case insensitive based on the current user's locale (affects only `text` columns). For example, most English and Western European locales treat the letters A-Z and ANSI letters like Ä and Ü as identical to their lowercase counterparts. This method also uses a "word sort", which treats hyphens and apostrophes in such a way that words like "coop" and "co-op" stay together.

Desc: Descending order. The column starts off in descending order the first time the user sorts it.

Logical [v1.0.44.12+]: Same as *CaseLocale* except that any sequences of digits in the text are treated as true numbers rather than mere characters. For example, the string T33 would be considered greater than T4. *Logical* requires Windows XP or later (on older OSes, *CaseLocale* is automatically used instead). In addition, *Logical* and *Case* are currently mutually exclusive: only the one most recently specified will be in effect.

NoSort: Prevents a user's click on this column from having any automatic sorting effect. To disable sorting for all columns rather than only a subset, include `NoSort` in the ListView's options. If the ListView has a g-label, the [ColClick notification](#) will still be received when the user clicks a no-sort column.

Sort: Immediately sorts the column in ascending order (even if it has the `Desc` option).

SortDesc: Immediately sorts the column in descending order.

Uni: Unidirectional sort. This prevents a second click on the same column from reversing the sort direction.

LV_InsertCol(ColumnNumber [, Options, ColumnTitle]): Creates a new column, inserting it as the specified *ColumnNumber* (shifting any other columns to the right to make room). The first column is 1 (not 0). If *ColumnNumber* is larger than the number of columns currently in the control, the new column is added to the end of the list. The newly inserted column starts off with empty contents beneath it unless it is the first column, in which case it inherits the old first column's contents and the old first column acquires blank contents. The new column's attributes -- such as whether or not it uses [integer sorting](#) -- always start off at their defaults unless changed via [Options](#). This function returns the new column's position number (or 0 upon failure). The maximum number of columns in a ListView is 200.

LV_DeleteCol(ColumnNumber): Deletes the specified column and all of the contents beneath it. It returns 1 upon success and 0 upon failure. Once a column is deleted, the column numbers of any that lie to its right are reduced by 1. Consequently, calling LV_DeleteCol(2) twice would delete the second and third columns. On operating systems older than Windows XP, attempting to delete the original first column might fail and return 0.

LV_GetCount(["Selected | Column"]): When the parameter is omitted, the function returns the total number of rows in the control. When the parameter is "S" or "Selected", the count includes only the selected/highlighted rows. When the parameter is "Col" or "Column", the function returns the number of columns in the control. This function is always instantaneous because the control keeps track of these counts.

This function is often used in the top line of a Loop, in which case the function would get called only once (prior to the first iteration). For example:

```
Loop % LV_GetCount()
{
    LV_GetText(RetrievedText, A_Index)
    if InStr(RetrievedText, "some filter text")
        LV_Modify(A_Index, "Select") ; Select each row whose first field contains the
                                    ; filter-text.
}
```

To retrieve the widths of a ListView's columns -- for uses such as saving them to an INI file to be remembered between sessions -- follow this example:

```
Gui +LastFound
Loop % LV_GetCount("Column")
{
```

```

SendMessage, 4125, A_Index - 1, 0, SysListView321 ; 4125 is
LVM_GETCOLUMNWIDTH.

MsgBox Column %A_Index%'s width is %ErrorLevel%.
}

```

LV_GetNext([StartingRowNumber, "Checked | Focused"]): Returns the row number of the next selected, checked, or focused row. If none is found, zero is returned. If *StartingRowNumber* is omitted or less than 1, the search begins at the top of the list. Otherwise, the search begins at the row after *StartingRowNumber*. If the second parameter is omitted, the function searches for the next selected/highlighted row. Otherwise, specify "C" or "Checked" to find the next checked row; or "F" or "Focused" to find the focused row (there is never more than one focused row in the entire list, and sometimes there is none at all). The following example reports all selected rows in the ListView:

```

RowNumber = 0 ; This causes the first loop iteration to start the search at the top of
the list.

Loop

{
    RowNumber := LV_GetNext(RowNumber) ; Resume the search at the row after
that found by the previous iteration.

    if not RowNumber ; The above returned zero, so there are no more selected rows.

        break

    LV_GetText(Text, RowNumber)

    MsgBox The next selected row is #%RowNumber%, whose first field is "%Text%".
}

```

An alternate method to find out if a particular row number is checked is the following:

```

Gui +LastFound

SendMessage, 4140, RowNumber - 1, 0xF000, SysListView321 ; 4140 is
LVM_GETITEMSTATE. 0xF000 is LVIS_STATEIMAGEMASK.

IsChecked := (ErrorLevel >> 12) - 1 ; This sets IsChecked to true if RowNumber is
checked or false otherwise.

```

LV_GetText(OutputVar, RowNumber [, ColumnNumber]): Retrieves the text at the specified *RowNumber* and *ColumnNumber* and stores it in *OutputVar*. If *ColumnNumber* is omitted, it defaults to 1 (the text in the first column). If *RowNumber* is 0, the column header text is retrieved. If the text is longer than 8191, only the first 8191 characters are retrieved. The function returns 1 upon success and 0 upon failure. Upon failure, *OutputVar* is also made blank.

Column numbers seen by the script are not altered by any dragging and dropping of columns the user may have done. For example, the original first column is still number 1 even if the user drags it to the right of other columns.

A **g-label**(See 19.3) such as **gMySubroutine** may be listed in the control's options. This would cause the *MySubroutine* label to be launched automatically whenever the user performs an action in the control. This subroutine may consult the built-in variables **A_Gui**(See 9.) and **A_GuiControl**(See 9.) to find out which window and ListView generated the event. More importantly, it may consult **A_GuiEvent**, which contains one of the following strings or letters (for compatibility with future versions, a script should not assume these are the only possible values):

DoubleClick: The user has double-clicked a row. The variable **A_EventInfo** contains the row number.

R: The user has *double-right*-clicked within the control. The variable **A_EventInfo** contains the focused row number.

ColClick: The user has clicked a column header. The variable **A_EventInfo** contains the column number, which is the original number assigned when the column was created; that is, it does not reflect any dragging and dropping of columns done by the user. One possible response to a column click is to sort by a hidden column (zero width) that contains data in a sort-friendly format (such as a YYYYMMDD integer date). Such a hidden column can mirror some other column that displays the same data in a more friendly format (such as MM/DD/YY). For example, a script could hide column 3 via **LV_ModifyCol(3, 0)**, then disable automatic sorting in the visible column 2 via **LV_ModifyCol(2, "NoSort")**. Then in response to the ColClick notification for column 2, the script would sort the ListView by the hidden column via **LV_ModifyCol(3, "Sort")**.

D: The user has attempted to start dragging a row or icon (there is currently no built-in support for dragging rows or icons). The variable **A_EventInfo** contains the focused row number. In v1.0.44+, this notification occurs even without **AltSubmit**.

d (lowercase D): Same as above except a right-click-drag rather than a left-drag.

e (lowercase E): The user has finished editing the first field of a row (the user may edit it only when the ListView has **-ReadOnly** in its options). The variable **A_EventInfo** contains the row number.

If the ListView has the word **AltSubmit** in its **options**, its g-label is launched more often and **A_GuiEvent** may contain the following additional values:

Normal: The user has left-clicked a row. The variable **A_EventInfo** contains the focused row number.

RightClick: The user has right-clicked a row. The variable A_EventInfo contains the focused row number. In most cases, it is best not to display a menu in response to this. Instead, use the [GuiContextMenu label](#)(See 19.3) because it also recognizes the Apps key. For example:

```
GuiContextMenu: ; Launched in response to a right-click or press of the Apps key.

if A_GuiControl <> ListView ; This check is optional. It displays the menu only for
clicks inside the ListView.

    return

; Show the menu at the provided coordinates, A_GuiX and A_GuiY. These should be
used

; because they provide correct coordinates even if the user pressed the Apps key:

Menu, MyContextMenu, Show, %A_GuiX%, %A_GuiY%

return
```

A: A row has been activated, which by default occurs when it is double clicked. The variable A_EventInfo contains the row number.

C: The ListView has released mouse capture.

E: The user has begun editing the first field of a row (the user may edit it only when the ListView has [-ReadOnly](#) in its options). The variable A_EventInfo contains the row number.

F: The ListView has received keyboard focus.

f (lowercase F): The ListView has lost keyboard focus.

I: Item changed. A row has changed by becoming selected/deselected, checked/unchecked, etc. If the user selects a new row, at least two such notifications are received: one for the de-selection of the previous row, and one for the selection of the new row. In v1.0.44+, the variable A_EventInfo contains the row number. In v1.0.46.10+, ErrorLevel contains zero or more of the following letters to indicate how the item changed: S (select) or s (de-select), and/or F (focus) or f (de-focus), and/or C (checkmark) or c (uncheckmark). For example, SF means that the row has been selected and focused. To detect whether a particular letter is present, use a [parsing loop](#)(See 17.14) or the case-sensitive option of [InStr\(\)](#)(See 10.); for example: *InStr(ErrorLevel, "S", true)*. Note: For compatibility with future versions, a script should not assume that "SsFfCc" are the only possible letters. Also, specifying [Critical](#)(See 22.7) as the [g-label](#)(See 19.3)'s first line ensures that all "I" notifications are received (otherwise, some might be lost if the script cannot keep up with them).

K: The user has pressed a key while the ListView has focus. A_EventInfo contains the virtual key code of the key, which is a number between 1 and 255. If the key is alphabetic, on most keyboard layouts it can be translated to the corresponding character via [Chr\(A_EventInfo\)](#)(See 10.). F2 keystrokes are received regardless of [WantF2](#). However, the Enter keystroke is not received; to receive it, use a default button as described [below](#).

M: Marquee. The user has started to drag a selection-rectangle around a group of rows or icons.

S: The user has begun scrolling the ListView.

s (lowercase S): The user has finished scrolling the ListView.

An Image-List is a group of identically sized icons stored in memory. Upon creation, each ImageList is empty. The script calls `IL_Add()` repeatedly to add icons to the list, and each icon is assigned a sequential number starting at 1. This is the number to which the script refers to display a particular icon in a row or column header. Here is a working example that demonstrates how to put icons into a ListView's rows:

```
Gui, Add, ListView, h200 w180, Icon & Number|Description ; Create a ListView.

ImageListID := IL_Create(10) ; Create an ImageList to hold 10 small icons.

LV_SetImageList(ImageListID) ; Assign the above ImageList to the current ListView.

Loop 10 ; Load the ImageList with a series of icons from the DLL.

    IL_Add(ImageListID, "shell32.dll", A_Index) ; Omits the DLL's path so that it
    works on Windows 9x too.

Loop 10 ; Add rows to the ListView (for demonstration purposes, one for each icon).

    LV_Add("Icon" . A_Index, A_Index, "n/a")

    LV_ModifyCol("Hdr") ; Auto-adjust the column widths.

Gui Show

return

GuiClose: ; Exit the script when the user closes the ListView's GUI window.

ExitApp
```

IL_Create([InitialCount, GrowCount, LargeIcons?]): Creates a new ImageList, initially empty, and returns the unique ID of the ImageList (or 0 upon failure). *InitialCount* is the number of icons you expect to put into the list immediately (if omitted, it defaults to 2). *GrowCount* is the number of icons by which the list will grow each time it exceeds the current list capacity (if omitted, it defaults to 5). *LargeIcons* should be a numeric value: If non-zero, the ImageList will contain large icons. If zero, it will contain small icons (this is the default when omitted). Icons added to the list are scaled automatically to conform to the system's dimensions for small and large icons.

LV_SetImageList(ImageListID [, 0|1|2]): This function is normally called prior to adding any rows to the ListView. It sets the [ImageList](#) whose icons will be displayed by the ListView's rows (and optionally, its columns). ImageListID is the number returned from a previous call to [IL_Create\(\)](#). If the second parameter is omitted, the type of icons in the ImageList is detected automatically as large or small. Otherwise, specify 0 for large icons, 1 for small icons, and 2 for state icons (state icons are not yet directly supported, but they could be used via [SendMessage](#)(See 28.1.12)).

A ListView may have up to two ImageLists: small-icon and/or large-icon. This is useful when the script allows the user to switch to and from the large-icon view. To add more than one ImageList to a ListView, call `LV_SetImageList()` a second time, specifying the ImageListID of the second list. A ListView with both a large-icon and small-icon ImageList should ensure that both lists contain the icons in the same order. This is because the same ID number is used to reference both the large and small versions of a particular icon.

Although it is traditional for all [viewing modes](#) except Icon and Tile to show small icons, this can be overridden by passing a large-icon list to `LV_SetImageList` and specifying 1 (small-icon) for the second parameter. This also increases the height of each row in the ListView to fit the large icon.

If successful, `LV_SetImageList()` returns the ImageListID that was previously associated with the ListView (or 0 if none). Any such detached ImageList should normally be destroyed via [IL_Destroy\(ImageListID\)](#).

IL_Add(ImageListID, Filename [, IconNumber, ResizeNonIcon?]): Adds an icon or picture to the specified *ImageListID* and returns the new icon's index (1 is the first icon, 2 is the second, and so on). *Filename* is the name of an icon (.ICO), cursor (.CUR), or animated cursor (.ANI) file (animated cursors will not actually be animated when displayed in a ListView). Other sources of icons include the following types of files: EXE, DLL, CPL, SCR, and other types that contain icon resources. To use an icon group other than the first one in the file, specify its number for *IconNumber*. In the following example, the default icon from the second icon group would be used: `IL_Add(ImageListID, "C:\My Application.exe", 2)`.

Non-icon images such as BMP, GIF and JPG may also be loaded. However, in this case the last two parameters should be specified to ensure correct behavior: *IconNumber* should be the mask/transparency color number (0xFFFFFFFF [the color white] might be best for most pictures); and *ResizeNonIcon* should be non-zero to cause the picture to be scaled to become a single icon, or zero to divide up the image into however many icons can fit into its actual width.

All operating systems support GIF, JPG, BMP, ICO, CUR, and ANI images. On Windows XP or later, additional image formats such as PNG, TIF, Exif, WMF, and EMF are supported. Operating systems older than XP can be given support by copying Microsoft's free GDI+ DLL into the AutoHotkey.exe folder (but in the case of a [compiled script](#)(See 8.), copy the DLL into the script's folder). To download the DLL, search for the following phrase at www.microsoft.com/gdi redistributable

IL_Destroy(ImageListID): Deletes the specified ImageList and returns 1 upon success and 0 upon failure. It is normally not necessary to destroy ImageLists because once attached to a

ListView, they are destroyed automatically when the ListView or its parent window is destroyed. However, if the ListView shares ImageLists with other ListViews (by having 0x40 in its options), the script should explicitly destroy the ImageList after destroying all the ListViews that use it. Similarly, if the script replaces one of a ListView's old ImageLists with a new one, it should explicitly destroy the old one.

The [Gui Submit](#)(See 19.3) command has no effect on a ListView control. Therefore, the script may use the ListView's [associated variable](#)(See 19.3) (if any) to store other data without concern that it will ever be overwritten.

After a column is sorted -- either by means of the user clicking its header or the script calling [LV_ModifyCol\(1, "Sort"\)](#) -- any subsequently added rows will appear at the bottom of the list rather than obeying the sort order. The exception to this is the [Sort](#) and [SortDesc](#) styles, which move newly added rows into the correct positions.

To detect when the user has pressed Enter while a ListView has focus, use a [default button](#)(See 19.4) (which can be hidden if desired). For example:

```
Gui, Add, Button, Hidden Default, OK
...
ButtonOK:
    GuiControlGet, FocusedControl, FocusV
    if FocusedControl <> ListView
        return
    MsgBox % "Enter was pressed. The focused row number is " . LV_GetNext(0,
    "Focused")
    return
```

In addition to navigating from row to row with the keyboard, the user may also perform incremental search by typing the first few characters of an item in the first column. This causes the selection to jump to the nearest matching row.

Although any length of text can be stored in each field of a ListView, only the first 260 characters are displayed.

Although the maximum number of rows in a ListView is limited only by available system memory, row-adding performance can be greatly improved as described in the [Count](#) option.

A picture may be used as a background around a ListView (that is, to frame the ListView). To do this, create the [picture control](#)(See 19.4) after the ListView and include 0x4000000 (which is [WS_CLIPSIBLINGS](#)) in the picture's *Options*.

A script may create more than one ListView per window. To operate upon a ListView other than the default one, see [built-in functions](#).

It is best not to insert or delete columns directly with [SendMessage](#)(See 28.1.12). This is because the program maintains a collection of [sorting preferences](#) for each column, which would then get out of sync. Instead, use the [built-in column functions](#).

To perform actions such as resizing, hiding, or changing the font of a ListView, use [GuiControl](#)(See 19.5).

To extract text from external ListViews (those not owned by the script), use "[ControlGet List](#)(See 28.1.4)".

Windows 95 and NT4: If the system lacks version 4.70 or later of Comctl32.dll, Shell32.dll, and Shlwapi.dll -- which are distributed with various updates and applications such as Internet Explorer 3.0 or later -- ListViews are more limited and some features might not behave as expected.

[TreeView](#)(See 19.8), [Other Control Types](#)(See 19.4), [Gui](#)(See 19.3), [GuiContextMenu](#)(See 19.3), [GuiControl](#)(See 19.5), [GuiControlGet](#)(See 19.6), [ListView styles table](#)(See 30.18)

; Select or de-select all rows by specifying 0 as the row number:

```
LV_Modify(0, "Select") ; Select all.  
LV_Modify(0, "-Select") ; De-select all.  
LV_Modify(0, "-Check") ; Uncheck all the checkboxes.
```

; Auto-size all columns to fit their contents:

```
LV_ModifyCol() ; There are no parameters in this mode.
```

; MAIN EXAMPLE

; The following is a working script that is more elaborate than the one near the top of this page.

; It displays the files in a folder chosen by the user, with each file assigned the icon associated with

; its type. The user can double-click a file, or right-click one or more files to display a context menu.

```

; Allow the user to maximize or drag-resize the window:
Gui +Resize

; Create some buttons:
Gui, Add, Button, Default gButtonLoadFolder, Load a folder
Gui, Add, Button, x+20 gButtonClear, Clear List
Gui, Add, Button, x+20, Switch View

; Create the ListView and its columns:
Gui, Add, ListView, xm r20 w700 vMyListView gMyListView, Name|In Folder|Size
(KB)|Type
LV_ModifyCol(3, "Integer") ; For sorting, indicate that the Size column is an integer.

; Create an ImageList so that the ListView can display some icons:
ImageListID1 := IL_Create(10)
ImageListID2 := IL_Create(10, 10, true) ; A list of large icons to go with the small ones.

; Attach the ImageLists to the ListView so that it can later display the icons:
LV_SetImageList(ImageListID1)
LV_SetImageList(ImageListID2)

; Create a popup menu to be used as the context menu:
Menu(See 22.13), MyContextMenu, Add, Open, ContextOpenFile
Menu, MyContextMenu, Add, Properties, ContextProperties
Menu, MyContextMenu, Add, Clear from ListView, ContextClearRows
Menu, MyContextMenu, Default, Open ; Make "Open" a bold font to indicate that
double-click does the same thing.

; Display the window and return. The OS will notify the script whenever the user

```

```
; performs an eligible action:  
Gui, Show  
return  
  
ButtonLoadFolder:  
Gui +OwnDialogs ; Forces user to dismiss the following dialog before using main window.  
FileSelectFolder, Folder,, 3, Select a folder to read:  
if not Folder ; The user canceled the dialog.  
    return  
  
; Check if the last character of the folder name is a backslash, which happens for root  
; directories such as C:\. If it is, remove it to prevent a double-backslash later on.  
StringRight, LastChar, Folder, 1  
if LastChar = \  
    StringTrimRight, Folder, Folder, 1 ; Remove the trailing backslash.  
  
; Ensure the variable has enough capacity to hold the longest file path. This is done  
; because ExtractAssociatedIconA() needs to be able to store a new filename in it.  
VarSetCapacity(Filename, 260)  
sfi_size = 352  
VarSetCapacity(sfi, sfi_size)  
  
; Gather a list of file names from the selected folder and append them to the ListView:  
GuiControl, -Redraw, MyListView ; Improve performance by disabling redrawing during  
load.  
Loop %Folder%\*.*  
{  
    FileName := A_LoopFileFullPath ; Must save it to a writable variable for use below.
```

```
; Build a unique extension ID to avoid characters that are illegal in variable names,
; such as dashes. This unique ID method also performs better because finding an item
; in the array does not require search-loop.

SplitPath, FileName,,, FileExt ; Get the file's extension.

if FileExt in EXE,ICO,ANI,CUR

{

    ExtID := FileExt ; Special ID as a placeholder.

    IconNumber = 0 ; Flag it as not found so that these types can each have a unique
icon.

}

else ; Some other extension/file-type, so calculate its unique ID.

{

    ExtID = 0 ; Initialize to handle extensions that are shorter than others.

    Loop 7 ; Limit the extension to 7 characters so that it fits in a 64-bit value.

    {

        StringMid, ExtChar, FileExt, A_Index, 1

        if not ExtChar ; No more characters.

            break

        ; Derive a Unique ID by assigning a different bit position to each character:

        ExtID := ExtID | (Asc(ExtChar) << (8 * (A_Index - 1)))

    }

    ; Check if this file extension already has an icon in the ImageLists. If it does,
    ; several calls can be avoided and loading performance is greatly improved,
    ; especially for a folder containing hundreds of files:

    IconNumber := IconArray%ExtID%

}

if not IconNumber ; There is not yet any icon for this extension, so load it.

{
```

```

; Get the high-quality small-icon associated with this file extension:

if not DllCall("Shell32\SHGetFileInfoA", "str", FileName, "uint", 0, "str", sfi, "uint",
sfi_size, "uint", 0x101) ; 0x101 is SHGFI_ICON+SHGFI_SMALLICON

    IconNumber = 9999999 ; Set it out of bounds to display a blank icon.

else ; Icon successfully loaded.

{

    ; Extract the hIcon member from the structure:

    hIcon = 0

    Loop 4

        hIcon += *(&sfi + A_Index-1) << 8*(A_Index-1)

    ; Add the HICON directly to the small-icon and large-icon lists.

    ; Below uses +1 to convert the returned index from zero-based to one-based:

    IconNumber := DllCall("ImageList_ReplaceIcon", "uint", ImageListID1, "int",
-1, "uint", hIcon) + 1

    DllCall("ImageList_ReplaceIcon", "uint", ImageListID2, "int", -1, "uint", hIcon)

    ; Now that it's been copied into the ImageLists, the original should be
destroyed:

    DllCall("DestroyIcon", "uint", hIcon)

    ; Cache the icon to save memory and improve loading performance:

    IconArray%ExtID% := IconNumber

}

;

; Create the new row in the ListView and assign it the icon number determined above:

LV_Add("Icon" . IconNumber, A_LoopFileName, A_LoopFileDir, A_LoopFileSizeKB,
FileExt)

}

GuiControl, +Redraw, MyListView ; Re-enable redrawing (it was disabled above).

LV_ModifyCol() ; Auto-size each column to fit its contents.

LV_ModifyCol(3, 60) ; Make the Size column a little wider to reveal its header.

return

```

```
ButtonClear:  
    LV_Delete() ; Clear the ListView, but keep icon cache intact for simplicity.  
    return  
  
ButtonSwitchView:  
    if not IconView  
        GuiControl, +Icon, MyListView ; Switch to icon view.  
    else  
        GuiControl, +Report, MyListView ; Switch back to details view.  
    IconView := not IconView ; Invert in preparation for next time.  
    return  
  
MyListView:  
    if A_GuiEvent = DoubleClick ; There are many other possible values the script can check.  
    {  
        LV_GetText(FileName, A_EventInfo, 1) ; Get the text of the first field.  
        LV_GetText(FileDir, A_EventInfo, 2) ; Get the text of the second field.  
        Run %FileDir%\%FileName%, UseErrorLevel  
        if ErrorLevel  
            MsgBox Could not open "%FileDir%\%FileName%".  
    }  
    return  
  
GuiContextMenu(See 19.3): ; Launched in response to a right-click or press of the Apps key.  
    if A_GuiControl <> MyListView ; Display the menu only for clicks inside the ListView.  
        return
```

```

; Show the menu at the provided coordinates, A_GuiX and A_GuiY. These should be used
; because they provide correct coordinates even if the user pressed the Apps key:

Menu, MyContextMenu, Show, %A_GuiX%, %A_GuiY%

return


ContextOpenFile: ; The user selected "Open" in the context menu.

ContextProperties: ; The user selected "Properties" in the context menu.

; For simplicity, operate upon only the focused row rather than all selected rows:

FocusedRowNumber := LV_GetNext(0, "F") ; Find the focused row.

if not FocusedRowNumber ; No row is focused.

    return

LV.GetText(FileName, FocusedRowNumber, 1) ; Get the text of the first field.

LV.GetText(FileDir, FocusedRowNumber, 2) ; Get the text of the second field.

IfInString A_ThisMenuItem, Open ; User selected "Open" from the context menu.

    Run %FileDir%\%FileName%, UseErrorLevel

else ; User selected "Properties" from the context menu.

    Run Properties "%FileDir%\%FileName%", UseErrorLevel

if ErrorLevel

    MsgBox Could not perform requested action on "%FileDir%\%FileName%".

return


ContextClearRows: ; The user selected "Clear" in the context menu.

RowCount = 0 ; This causes the first iteration to start the search at the top.

Loop

{

    ; Since deleting a row reduces the RowCount of all other rows beneath it,
    ; subtract 1 so that the search includes the same row number that was previously
    ; found (in case adjacent rows are selected):

    RowCount := LV_GetNext(RowCount - 1)
}

```

```

if not RowNumber ; The above returned zero, so there are no more selected rows.

break

LV_Delete(RowNumber) ; Clear the row from the ListView.

}

return

GuiSize: ; Expand or shrink the ListView in response to the user's resizing of the window.

if A_EventInfo = 1 ; The window has been minimized. No action needed.

return

; Otherwise, the window has been resized or maximized. Resize the ListView to match.

GuiControl, Move, MyListView, % "W" . (A_GuiWidth - 20) . " H" . (A_GuiHeight - 40)

return

GuiClose: ; When the window is closed, exit the script automatically:

ExitApp

```

19.8 Gui TreeView control

- [Introduction and Simple Example](#)
- [Options and Styles](#)
- [Built-in Functions: 1\) Add/modify/delete items; 2\) Getting data out of a TreeView](#)
- [G-Label Notifications](#)
- [Remarks](#)
- [Longer Example](#)

A Tree-View displays a hierarchy of items by indenting child items beneath their parents. The most common example is Explorer's tree of drives and folders.

The syntax for creating a TreeView is:

Gui, Add, TreeView, Options

Here is a working script that creates and displays a simple hierarchy of items:

```

Gui, Add, TreeView

P1 := TV_Add("First parent")

P1C1 := TV_Add("Parent 1's first child", P1) ; Specify P1 to be this item's parent.

P2 := TV_Add("Second parent")

P2C1 := TV_Add("Parent 2's first child", P2)

P2C2 := TV_Add("Parent 2's second child", P2)

P2C2C1 := TV_Add("Child 2's first child", P2C2)

Gui, Show ; Show the window and its TreeView.

return

GuiClose: ; Exit the script when the user closes the TreeView's GUI window.

ExitApp

```

AltSubmit: Notifies the script for more types of TreeView events than normal. In other words, the g-label is launched more often. See [TreeView Notifications](#) for details.

Background: Specify the word Background followed immediately by a color name (see [color chart](#)(See 19.12)) or RGB value (the 0x prefix is optional). Examples: BackgroundSilver, BackgroundFFDD99. If this option is not present, the TreeView initially defaults to the background color set by the last parameter of [Gui Color](#)(See 19.3) (or if none, the system's default background color). Specifying BackgroundDefault applies the system's default background color (usually white). For example, a TreeView can be restored to the default color via *GuiControl, +BackgroundDefault, MyTreeView*.

Buttons: Specify -Buttons (minus Buttons) to avoid displaying a plus or minus button to the left of each item that has children.

C: Text color. Specify the letter C followed immediately by a color name (see [color chart](#)(See 19.12)) or RGB value (the 0x prefix is optional). Examples: cRed, cFF2211, c0xFF2211, cDefault

Checked: Provides a checkbox at the left side of each item. When [adding](#) an item, specify the word *Check* in its options to have the box to start off checked instead of unchecked. The user may either click the checkbox or press the spacebar to check or uncheck an item. To discover which items in a TreeView are currently checked, call [TV_GetNext\(\)](#) or [TV_Get\(\)](#).

HScroll: Specify -HScroll (minus HScroll) to disable horizontal scrolling in the control (in addition, the control will not display any horizontal scroll bar). On operating systems older than Windows 2000/Me, this option has no effect unless the system has Comctl32.dll 5.8 or greater (distributed with applications such as Internet Explorer 5 or later).

ImageList: This is the means by which icons are added to a TreeView. Specify the word *ImageList* followed immediately by the ImageListID returned from a previous call to [IL_Create\(\)](#)(See 19.7). This option has an effect only when creating a TreeView. Here is a working example:

```
ImageListID := IL_Create(See 19.7)(10) ; Create an ImageList with initial capacity
for 10 icons.

Loop 10 ; Load the ImageList with some standard system icons.

    IL_Add(See 19.7)(ImageListID, "shell32.dll", A_Index) ; Omits the DLL's path so
that it works on Windows 9x too.

Gui, Add, TreeView, ImageList%ImageListID%

TV_Add("Name of Item", 0, "Icon4") ; Add an item to the TreeView and give it a folder
icon.

Gui Show
```

Lines: Specify -Lines (minus Lines) to avoid displaying a network of lines connecting parent items to their children. However, removing these lines also prevents the plus/minus buttons from being shown for top-level items.

ReadOnly: Specify -ReadOnly (minus ReadOnly) to allow editing of the text/name of each item. To edit an item, select it then press the [F2 key](#). Alternatively, you can click an item once to select it, wait at least half a second, then click the same item again to edit it. After being edited, an item can be alphabetically repositioned among its siblings via the following example:

```
Gui, Add, TreeView, -ReadOnly gMyTree

; ...

MyTree:

if (A_GuiEvent == "e") ; The user has finished editing an item (use == for case
sensitive comparison).

    TV_Modify(TV_GetParent(A_EventInfo), "Sort") ; This works even if the item has
no parent.

return
```

R: Rows of height (upon creation). Specify the letter R followed immediately by the number of rows for which to make room inside the control. For example, R10 would make the control 10 items tall.

WantF2: Specify -WantF2 (minus WantF2) to prevent an F2 keystroke from [editing](#) the currently selected item. This setting is ignored unless [-ReadOnly](#) is also in effect. Regardless of this setting, the g-label still receives F2 [notifications](#).

(Unnamed numeric styles): Since styles other than the above are rarely used, they do not have names. See the [TreeView styles table](#)(See 30.18) for a list.

All of the TreeView functions operate upon the current thread's [default GUI window](#)(See 19.3) (which can be changed via [Gui, 2:Default](#)(See 19.3)). If the default window does not exist or has no TreeView controls, all functions return zero to indicate the problem.

If the window has more than one TreeView control, by default the functions operate upon the one most recently added. To change this, specify *Gui, TreeView, TreeViewName*, where *TreeViewName* is the name of the TreeView's [associated variable](#)(See 19.3) or its ClassNN as shown by Window Spy. Once changed, all existing and future [threads](#)(See 30.19) will use the indicated TreeView.

TV_Add(*Name*, [*ParentItemID*, *Options*]): Adds a new item to the TreeView and returns its unique Item ID number (or 0 upon failure). *Name* is the displayed text of the item, which can be text or numeric (including numeric [expression](#)(See 9.) results). *ParentItemID* is the ID number of the new item's parent (omit it or specify 0 to add the item at the top level). When adding a large number of items, performance can be improved by using *GuiControl*, [-Redraw](#), *MyTreeView* before adding the items, and *GuiControl*, [+Redraw](#), *MyTreeView* afterward.

Options for TV_Add() and TV_Modify(): The *Options* parameter is a string containing zero or more words from the list below (not case sensitive). Separate each word from the next with a space or tab. To remove an option, precede it with a minus sign. To add an option, a plus sign is permitted but not required.

Bold: Displays the item's name in a bold font. To later un-bold the item, use *TV_Modify(ItemID, "-Bold")*.

Check: Shows a checkmark to the left of the item (if the TreeView has [checkboxes](#)). To later uncheck it, use *TV_Modify(ItemID, "-Check")*. The word *Check* may optionally be followed immediately by a 0 or 1 to indicate the starting state. In other words, both "*Check*" and "*Check*". *VarContainingOne* are the same (the period used here is the [concatenation operator](#)(See 9.)).

Expand: Expands the item to reveal its children (if any). To later collapse the item, use *TV_Modify(ItemID, "-Expand")*. If there are no children, [TV_Modify\(\)](#) returns 0 instead of the item's ID. By contrast, [TV_Add\(\)](#) marks the item as expanded in case children are added to it later. Unlike "Select" (below), expanding an item does not automatically expand its parent.

Finally, the word *Expand* may optionally be followed immediately by a 0 or 1 to indicate the starting state. In other words, both "*Expand*" and "*Expand*".*VarContainingOne* are the same.

First | Sort | N: These options apply only to [TV_Add\(\)](#). They specify the new item's position relative to its siblings (a *sibling* is any other item on the same level). If none of these options is present, the new item is added as the last/bottom sibling. Otherwise, specify *First* to add the item as the first/top sibling, or specify *Sort* to insert it among its siblings in alphabetical order. If a plain integer (**N**) is specified, it is assumed to be ID number of the sibling after which to insert the new item (if integer N is the only option present, it does not have to be enclosed in quotes).

Icon: Specify the word *Icon* followed immediately by the number of this item's icon, which is displayed to the left of the item's name. If this option is absent, the first icon in the [ImageList](#) is used. To display a blank icon, specify a number that is larger than the number of icons in the ImageList. If the control lacks an ImageList, no icon is displayed nor is any space reserved for one.

Select: Selects the item. Since only one item at a time can be selected, any previously selected item is automatically de-selected. In addition, this option reveals the newly selected item by expanding its parent(s), if necessary. To find out the current selection, call [TV_GetSelection\(\)](#).

Sort: For [TV_Modify\(\)](#), this option alphabetically sorts the children of the specified item. To instead sort all top-level items, use *TV_Modify(0, "Sort")*. If there are no children, 0 is returned instead of the ID of the modified item.

Vis: Ensures that the item is completely visible by scrolling the TreeView and/or expanding its parent, if necessary.

VisFirst: Same as above except that the TreeView is also scrolled so that the item appears at the top, if possible. This option is typically more effective when used with [TV_Modify\(\)](#) than with [TV_Add\(\)](#).

TV_Modify([ItemID [, Options, NewName]]): Modifies the attributes and/or name of an item. It returns the item's own ID upon success or 0 upon failure (or partial failure). When only the first parameter is present, the specified item is [selected](#). When *NewName* is omitted, the current name is left unchanged. For *Options*, see the list above.

TV_Delete([ItemID]): If *ItemID* is omitted, **all** items in the TreeView are deleted. Otherwise, only the specified *ItemID* is deleted. It returns 1 upon success and 0 upon failure.

TV_GetSelection(): Returns the selected item's ID number.

TV_GetCount(): Returns the total number of items in the control. This function is always instantaneous because the control keeps track of the count.

TV_GetParent(*ItemID*): Returns the specified item's parent as an item ID. Items at the top level have no parent and thus return 0.

TV_GetChild(*ParentItemID*): Returns the ID number of the specified item's first/top child (or 0 if none).

TV_GetPrev(*ItemID*): Returns the ID number of the sibling above the specified item (or 0 if none).

TV_GetNext([*ItemID*, "Checked | Full"]): This has the following modes:

1. When all parameters are omitted, it returns the ID number of the first/top item in the TreeView (or 0 if none).
2. When the only first parameter (*ItemID*) is present, it returns the ID number of the sibling below the specified item (or 0 if none). If the first parameter is 0, it returns the ID number of the first/top item in the TreeView (or 0 if none).
3. When the second parameter is "Full" or "F", the next item is retrieved regardless of its relationship to the specified item. This allows the script to easily traverse the entire tree, item by item. For example:

```
4. ItemID = 0 ; Causes the loop's first iteration to start at the top of the
tree.

5. Loop

6. {

7.     ItemID := TV_GetNext(ItemID, "Full") ; Replace "Full" with "Checked" to
find all checkmarked items.

8.     if not ItemID ; No more items in tree.

9.         break

10.    TV_GetText(ItemText, ItemID)

11.    MsgBox The next Item is %ItemID%, whose text is "%ItemText%".

}
```

12. When the second parameter is either "Check", "Checked", or "C", the same behavior as above is used except that any item without a checkmark is skipped over. This allows all checkmarked items in the TreeView to be retrieved, one by one.

TV.GetText(*OutputVar*, *ItemID*): Retrieves the text/name of the specified *ItemID* and stores it in *OutputVar*. If the text is longer than 8191, only the first 8191 characters are retrieved. Upon success, the function returns the item's own ID. Upon failure, it returns 0 (and *OutputVar* is also made blank).

TV.Get(*ItemID*, "Expand | Check | Bold"): If the specified item has the specified attribute, its own *ItemID* is returned. Otherwise, 0 is returned. For the second parameter, specify "E", "Expand", or "Expanded" to determine if the item is currently [expanded](#) (that is, its children are

being displayed); specify "C", "Check", or "Checked" to determine if the item has a [checkmark](#); or specify "B" or "Bold" to determine if the item is currently [bold](#) in font.

Tip: Since an IF-statement sees any non-zero value as "true", the following two lines are functionally identical:

1. if TV_Get(ItemID, "Checked") = ItemID
2. if TV_Get(ItemID, "Checked")

A [g-label](#)(See 19.3) such as **gMySubroutine** may be listed in the control's options. This would cause the *MySubroutine* label to be launched automatically whenever the user performs an action in the control. This subroutine may consult the built-in variables [A_Gui](#)(See 9.) and [A_GuiControl](#)(See 9.) to find out which window and TreeView generated the event. More importantly, it may consult [A_GuiEvent](#), which contains one of the following strings or letters (for compatibility with future versions, a script should not assume these are the only possible values):

DoubleClick: The user has double-clicked an item. The variable [A_EventInfo](#) contains the item ID.

D: The user has attempted to start dragging an item (there is currently no built-in support for this). The variable [A_EventInfo](#) contains the item ID.

d (lowercase D): Same as above except a right-click-drag rather than a left-drag.

e (lowercase E): The user has finished editing an item (the user may edit items only when the TreeView has [-ReadOnly](#) in its options). The variable [A_EventInfo](#) contains the item ID.

S: A new item has been selected, either by the user or the script itself. The variable [A_EventInfo](#) contains the newly selected item ID.

If the TreeView has the word AltSubmit in its [options](#), its g-label is launched more often and [A_GuiEvent](#) may contain the following additional values:

Normal: The user has left-clicked an item. The variable [A_EventInfo](#) contains the item ID.

RightClick: The user has right-clicked an item. The variable [A_EventInfo](#) contains the item ID. In most cases, it is best not to display a menu in response to this. Instead, use the [GuiContextMenu label](#)(See 19.3) because it also recognizes the Apps key. For example:

```
GuiContextMenu: ; Launched in response to a right-click or press of the Apps key.  
if A_GuiControl <> MyTreeView ; This check is optional. It displays the menu only for  
clicks inside the TreeView.  
return
```

```
; Show the menu at the provided coordinates, A_GuiX and A_GuiY. These should be
; used
; because they provide correct coordinates even if the user pressed the Apps key:
Menu, MyContextMenu, Show, %A_GuiX%, %A_GuiY%
return
```

E: The user has begun editing an item (the user may edit items only when the TreeView has [-ReadOnly](#) in its options). The variable A_EventInfo contains the item ID.

F: The TreeView has received keyboard focus.

f (lowercase F): The TreeView has lost keyboard focus.

K: The user has pressed a key while the TreeView has focus. A_EventInfo contains the virtual key code of the key, which is a number between 1 and 255. If the key is alphabetic, on most keyboard layouts it can be translated to the corresponding character via [Chr\(A_EventInfo\)](#)(See 10.). F2 keystrokes are received regardless of [WantF2](#). However, the Enter keystroke is not received; to receive it, use a default button as described [below](#).

+ (plus sign): An item has been expanded to reveal its children. The variable A_EventInfo contains the item ID.

- (minus sign): An item has been collapsed to hide its children. The variable A_EventInfo contains the item ID.

The [Gui Submit](#)(See 19.3) command has no effect on a TreeView control. Therefore, the script may use the TreeView's [associated variable](#)(See 19.3) (if any) to store other data without concern that it will ever be overwritten.

To detect when the user has pressed Enter while a TreeView has focus, use a [default button](#)(See 19.4) (which can be hidden if desired). For example:

```
Gui, Add, Button, Hidden Default, OK
...
ButtonOK:
GuiControlGet, FocusedControl, FocusV
if FocusedControl <> MyTreeView
    return
MsgBox % "Enter was pressed. The selected item ID is " . TV_GetSelection()
return
```

In addition to navigating from item to item with the keyboard, the user may also perform incremental search by typing the first few characters of an item's name. This causes the selection to jump to the nearest matching item.

Although any length of text can be stored in each item of a TreeView, only the first 260 characters are displayed.

Although the theoretical maximum number of items in a TreeView is 65536, item-adding performance will noticeably decrease long before then. This can be alleviated somewhat by using the redraw tip described in [TV_Add\(\)](#).

Unlike [ListViews](#)(See 19.7), a TreeView's ImageList is not automatically destroyed when the TreeView is destroyed. Therefore, a script should call [IL_Destroy\(ImageListID\)](#)(See 19.7) after destroying a TreeView's window if the ImageList will not be used for anything else. However, this is not necessary if the script will soon be exiting because all ImageLists are automatically destroyed at that time. On a related note, a TreeView's original ImageList can be replaced with a new one via the following example:

```
Gui +LastFound(See 19.3)

SendMessage(See 28.1.12), 0x1109, 0, NewImageListID, SysTreeView321 ; 0x1109
is TVM_SETIMAGELIST

if ErrorLevel ; The TreeView had a previous ImageList.

    IL_Destroy(See 19.7)(ErrorLevel) ; Destroying it is the most typical action.
```

A script may create more than one TreeView per window. To operate upon a TreeView other than the default one, see [built-in functions](#).

To perform actions such as resizing, hiding, or changing the font of a TreeView, use [GuiControl](#)(See 19.5).

Tree View eXtension (TVX) extends TreeViews to support moving, inserting and deleting. It is demonstrated at www.autohotkey.com/forum/topic19021.html

Windows 95 and NT4: If the system lacks version 4.70 or later of Comctl32.dll, Shell32.dll, and Shlwapi.dll -- which are distributed with various updates and applications such as Internet Explorer 3.0 or later -- TreeViews are more limited and some features might not behave as expected.

[ListView](#)(See 19.7), [Other Control Types](#)(See 19.4), [Gui](#)(See 19.3), [GuiContextMenu](#)(See 19.3), [GuiControl](#)(See 19.5), [GuiControlGet](#)(See 19.6), [TreeView styles table](#)(See 30.18)

; The following is a working script that is more elaborate than the one near the top of this page.

; It creates and displays a TreeView containing all folders in the all-users Start Menu.
When the

; user selects a folder, its contents are shown in a ListView to the right (like Windows Explorer).

; In addition, a [StatusBar](#)(See 19.4) control shows information about the currently selected folder.

; The following folder will be the root folder for the TreeView. Note that loading might take a long

; time if an entire drive such as C:\ is specified:

```
TreeRoot = %A_StartMenuCommon%
```

```
TreeViewWidth := 280
```

```
ListViewWidth := A_ScreenWidth - TreeViewWidth - 30
```

; Allow the user to maximize or drag-resize the window:

```
Gui +Resize
```

; Create an ImageList and put some standard system icons into it:

```
ImageListID := IL\_Create(See 19.7)(5)
```

Loop 5 ; Below omits the DLL's path so that it works on Windows 9x too:

```
    IL\_Add(See 19.7)(ImageListID, "shell32.dll", A_Index)
```

; Create a TreeView and a ListView side-by-side to behave like Windows Explorer:

```
Gui, Add, TreeView, vMyTree r20 w%TreeViewWidth% gMyTree
```

```
ImageList%ImageListID%
```

```
Gui, Add, ListView, vMyList r20 w%ListViewWidth% x+10, Name|Modified
```

; Set the ListView's column widths (this is optional):

```
Col2Width = 70 ; Narrow to reveal only the YYYYMMDD part.
```

```

LV_ModifyCol(1, ListViewWidth - Col2Width - 30) ; Allows room for vertical scrollbar.

LV_ModifyCol(2, Col2Width)

; Create a Status Bar to give info about the number of files and their total size:

Gui, Add, StatusBar(See 19.4)

SB_SetParts(See 19.4)(60, 85) ; Create three parts in the bar (the third part fills all the
remaining width).

; Add folders and their subfolders to the tree. Display the status in case loading takes a long
time:

SplashTextOn, 200, 25, TreeView and StatusBar Example, Loading the tree...

AddSubFoldersToTree(TreeRoot)

SplashTextOff

; Display the window and return. The OS will notify the script whenever the user performs
an eligible action:

Gui, Show,, %TreeRoot% ; Display the source directory (TreeRoot) in the title bar.

return

AddSubFoldersToTree(Folder, ParentItemID = 0)

{
    ; This function adds to the TreeView all subfolders in the specified folder.
    ; It also calls itself recursively to gather nested folders to any depth.

    Loop %Folder%\*.* , 2 ; Retrieve all of Folder's sub-folders.

        AddSubFoldersToTree(A_LoopFilePath, TV_Add(A_LoopFileName,
ParentItemID, "Icon4"))

}

MyTree: ; This subroutine handles user actions (such as clicking).

if A_GuiEvent <> S ; i.e. an event other than "select new tree item".

    return ; Do nothing.

```

```

; Otherwise, populate the ListView with the contents of the selected folder.

; First determine the full path of the selected folder:

TV_GetText(SelectedItemText, A_EventInfo)

ParentID := A_EventInfo

Loop ; Build the full path to the selected folder.

{
    ParentID := TV_GetParent(ParentID)

    if not ParentID ; No more ancestors.

        break

    TV_GetText(ParentText, ParentID)

    SelectedItemText = %ParentText%\%SelectedItemText%

}

SelectedFullPath = %TreeRoot%\%SelectedItemText%


; Put the files into the ListView:

LV_Delete() ; Clear all rows.

GuiControl, -Redraw, MyListView ; Improve performance by disabling redrawing during
load.

FileCount = 0 ; Init prior to loop below.

TotalSize = 0

Loop %SelectedFullPath%\*.* ; For simplicity, this omits folders so that only files are
shown in the ListView.

{
    LV_Add("", A_LoopFileName, A_LoopFileTimeModified)

    FileCount += 1

    TotalSize += A_LoopFileSize

}

GuiControl, +Redraw, MyListView


; Update the three parts of the status bar to show info about the currently selected folder:

```

```

SB_SetText(See 19.4)(FileCount . " files", 1)

SB_SetText(Round(TotalSize / 1024, 1) . " KB", 2)

SB_SetText(SelectedFullPath, 3)

return

GuiSize: ; Expand/shrink the ListView and TreeView in response to user's resizing of
window.

if A_EventInfo = 1 ; The window has been minimized. No action needed.

    return

; Otherwise, the window has been resized or maximized. Resize the controls to match.

GuiControl, Move, MyTree, % "H" . (A_GuiHeight - 30) ; -30 for StatusBar and margins.

GuiControl, Move, MyList, % "H" . (A_GuiHeight - 30) . " W" . (A_GuiWidth - TreeViewWidth
- 30)

return

GuiClose: ; Exit the script when the user closes the TreeView's GUI window.

ExitApp

```

19.9 IfMsgBox

检测用户在最近的 [MsgBox/消息框\(See 19.11\)](#) 命令点击了哪个按钮。

IfMsgBox, ButtonName

参数

	下方的字符串代表用户在最近的 MsgBox/消息框(See 19.11) 命令中点击的按钮。
ButtonName	Yes No OK Cancel Abort Ignore Retry Continue [v1.0.44.08+] TryAgain [v1.0.44.08+]

	Timeout (也就是说，单词“timeout”表示消息框 timed out/超时 (See 19.11))
--	--

相关命令

[MsgBox](#)(See 19.11)

示例

```
MsgBox, 4, , Would you like to continue?, 5 ; 5 秒超时时间。
```

```
IfMsgBox, No
```

```
    Return ; 用户点击了“No/否”按钮。
```

```
IfMsgBox, Timeout
```

```
    Return ; 即假定超时和“No/否”一样对待。
```

```
; 否则，继续：
```

```
...
```

19.10 InputBox

Displays an input box to ask the user to enter a string.

`InputBox, OutputVar [, Title, Prompt, HIDE, Width, Height, X, Y, Font, Timeout, Default]`

参数

OutputVar	The name of the variable in which to store the text entered by the user.
Title	The title of the input box. If blank or omitted, it defaults to the name of the script.
Prompt	The text of the input box, which is usually a message to the user indicating what kind of input is expected. If <i>Prompt</i> is long, it can be broken up into several shorter lines by means of a continuation section (See 8.), which might improve readability and maintainability.
HIDE	If this parameter is the word HIDE, the user's input will be masked, which is useful for passwords.
Width	If this parameter is blank or omitted, the starting width of the window will be 375. This parameter can be an expression (See 9.).
Height	If this parameter is blank or omitted, the starting height of the window will be 189. This parameter can be an expression (See 9.).
X, Y	The X and Y coordinates of the window (use 0,0 to move it to the upper left corner of the desktop), which can be expressions (See 9.). If either coordinate is blank or omitted, the dialog will be centered in that dimension. Either coordinate can be

	negative to position the window partially or entirely off the desktop.
Font	Not yet implemented (leave blank). In the future it might accept something like verdana:8
Timeout	Timeout in seconds (can contain a decimal point or be an expression (See 9.)). If this value exceeds 2147483 (24.8 days), it will be set to 2147483. After the timeout has elapsed, the InputBox window will be automatically closed and ErrorLevel (See 30.8) will be set to 2. <i>OutputVar</i> will still be set to what the user entered.
Default	A string that will appear in the InputBox's edit field when the dialog first appears. The user can change it by backspacing or other means.

ErrorLevel

See below.

注意

The dialog allows the user to enter text and then press OK or CANCEL. The user can resize the dialog window by dragging its borders.

If the user presses the CANCEL button: For AutoIt v2 (.aut) scripts, *OutputVar* is set to be blank and [ErrorLevel](#)(See 30.8) is not changed. For all other script types (e.g. .ahk, .ini) [ErrorLevel](#)(See 30.8) is set to 1 (or 0 for OK) and *OutputVar* to the value entered. This allows the CANCEL button to perform a function other than CANCEL should the script designer wish it.

If the dialog times out, [ErrorLevel](#)(See 30.8) will be set to 2 for all types of scripts, including AutoIt v2 scripts. For this reason, AutoIt v2 scripts should explicitly set ErrorLevel to 0 prior to using this command with a timeout.

A GUI window may display a modal InputBox by means of [Gui +OwnDialogs](#)(See 19.3). A modal InputBox prevents the user from interacting with the GUI window until the InputBox is dismissed.

相关命令

[GUI](#)(See 19.3), [Input](#)(See 20.11), [MsgBox](#)(See 19.11), [FileSelectFile](#)(See 16.23), [FileSelectFolder](#)(See 16.24), [SplashTextOn](#)(See 19.14), [ToolTip](#)(See 19.15)

示例

```
InputBox, password, Enter Password, (your input will be hidden), hide
InputBox, UserInput, Phone Number, Please enter a phone number., , 640, 480
if ErrorLevel
    MsgBox, CANCEL was pressed.
```

```
else
```

```
    MsgBox, You entered "%UserInput%"
```

19.11 MsgBox

在一个窗口中显示指定的文本，并包含一个或多个按钮（比如 Yes/是 和 No/否）。

MsgBox, Text

MsgBox [, Options, Title, Text, Timeout]

参数

	如果省略所有的参数， MsgBox 会显示“Press OK to continue.”否则，将显示该参数所指定的文本，一般用来显示提示信息或当前状态信息。 如果 Text/文本 太长，可以使用 continuation section/字符串分段(See 8.) 的方法将它分为几个短小的段落，这样可以增加代码的可读性和可维护性。
Options	指定消息框的类型和它所包含的按钮。如果留空或省略，默认为 0。其它允许的值可以查看下方的表格。 这个参数不能使用表达式或者类似 %option% 这样的变量引用，请直接使用数字。
Title	消息框的标题。如果留空或省略，默认为脚本名（不带路径）。
Timeout	（可选）超时时间，单位是秒（可以包含小数点但是不能是表达式）。如果它的值超过 2147483（24.8 天），它将会被设置为 2147483。超过超时时间以后消息框会自动关闭，并且可以使用 IfMsgBox(See 19.9) 命令的参数 TIMEOUT 来捕获超时状态。 已知局限：如果 MsgBox 只包含一个 OK/确定按钮，则当它超时后，并且本身的 thread/线程(See 30.19) 被其它脚本中断的时候， IfMsgBox(See 19.9) 会认为它是按下了 OK/确定按钮。

Options 参数可以是以下一个或多个值的和。

Function	Value
OK/确定（也就是只显示 OK/确定按钮）	0
OK/确定 Cancel/取消	1
Abort/终止 Retry/重试 Ignore/忽略	2
Yes/是 No/否 Cancel/取消	3

Yes/是 No/否	4
Retry/重试 Cancel/取消	5
Cancel/取消 Try Again/重试 Continue/继续 (2000/XP 以上)	6
添加一个帮助按钮 (看下方的注解)	16384
错误图标 (停止/错误)	16
询问图标	32
感叹图标	48
信息图标	64
使第二个按钮为默认按钮	256
使第三个按钮为默认按钮	512
系统模式对话框 (总在最前)	4096
任务模式对话框	8192
在默认的桌面上显示消息框 (Windows NT/2000/XP 或之后的系统)	131072
总在最前 (WS_EX_TOPMOST 风格) (和系统模式消息框相似但是省略了标题栏图标)	262144
文字右对齐	524288
从右往左的阅读顺序	1048576

注意

上方表格中的值在使用的时候可以进行相加。比如，要显示一个包含 是/否 按钮的 系统模式对话框，则 *Options* 的值就是 4096+4 (4100)。

MsgBox 对逗号的处理比较智能，所以一般在 *Text* 参数中不需要对逗号进行 [escape/转义\(See 29.5\)](#)。

要判断用户在最近一次消息框中按下了哪个按钮，可以使用 [If MsgBox\(See 19.9\)](#) 命令，例如：

```
 MsgBox, 4,, Would you like to continue? (press Yes or No)
```

```
If MsgBox Yes
```

```
    MsgBox You pressed Yes.
```

```
else
```

```
    MsgBox You pressed No.
```

按钮的名字是可以自定义的，参看 [这个例子\(See 30.30\)](#)。

提示：使用 **Ctrl+C** 可以复制当前激活的消息框中的文字。不仅是 **AutoHotkey** 提供的消息框，这个操作对所有的消息框都适用。

在 GUI/图形界面 中使用消息框：在 **GUI/图形界面** 窗口中，可以使用 [Gui +OwnDialogs\(See 19.3\)](#) 来显示一个 模式 消息框。模式 消息框可以避免用户在消息框消失之前对原来的 **GUI/图形界面** 窗口进行修改。注意：在这种情况下，在 *Options* 参数中可以不必指定 系统模式消息框 或者 任务模式消息框。

当 [Gui +OwnDialogs\(See 19.3\)](#) 无效 的时候，任务模式消息框（8192）可以屏蔽所有的脚本窗口，直到用户关闭了消息框。

帮助按钮：在 *Options* 参数中使用了帮助按钮（83）的时候，点击帮助按钮不会有任何效果，除非满足以下两个条件：

1. 消息框是 **GUI/图形界面** 窗口使用 [Gui +OwnDialogs\(See 19.3\)](#) 显示的。
 2. 脚本监听了 **WM_HELP** 消息（0x53）。例如：[OnMessage\(0x53, "WM_HELP"\)\(See 18.11\)](#)。
- 当 **WM_HELP()** 函数被调用的时候，可以显示另外一个窗口或者消息框来对用户进行帮助引导。

关闭按钮（在消息框的标题栏上）：因为消息框是操作系统内置功能，所以它的 **X/关闭** 按钮只有在能够使用的时候才会是启用状态。如果消息框只包含一个 **OK/确定** 按钮，点击 **X/关闭** 按钮等同于点击 **OK/确定** 按钮。否则，**X/关闭** 按钮处于停用状态，除非消息框包含了 **Cancel/取消** 按钮，这时点击 **X/关闭** 按钮就等同于点击 **Cancel/取消** 按钮。

相关命令

[IfMsgBox\(See 19.9\)](#), [InputBox\(See 19.10\)](#), [FileSelectFile\(See 16.23\)](#), [FileSelectFolder\(See 16.24\)](#), [ToolTip\(See 19.15\)](#), [GUI\(See 19.3\)](#)

示例

```

 MsgBox This is the 1-parameter method. Commas (,) do not need to be escaped.

 MsgBox, 4, , This is the 3-parameter method. Commas (,) do not need to be escaped.

 MsgBox, 4, , Do you want to continue? (Press YES or NO)

 IfMsgBox No

    return

 MsgBox, 4, , 4-parameter method: this MsgBox will time out in 5 seconds. Continue?, 5

 IfMsgBox Timeout

    MsgBox You didn't press YES or NO within the 5-second period.

 else IfMsgBox No

    return

;
```

； 在参数的前面加上“%”，它后面就成了一个表达式

; 在下面的例子中，进行了数学计算，读取了内置变量，调用了函数
; 它们之间使用“.”联接成一条单独的字符串进行显示。

```
MsgBox % "New width for object #" . A_Index . " is: " .
RestrictWidth(ObjectWidth%A_Index% * ScalingFactor)
```

; 下面的例子提醒用户一个消息框将要获取焦点（用户正在打字的时候）。

SplashTextOn,,, A MsgBox is about to appear.

Sleep 3000

SplashTextOff

MsgBox The backup process has completed.

19.12 Progress

Creates or updates a window containing a progress bar or an image.

SplashImage, Off

SplashImage [, ImageFile, Options, SubText, MainText, WinTitle, FontName]

Progress, Off

Progress, ProgressParam1 [, SubText, MainText, WinTitle, FontName]

参数

	<p>If this is the word OFF, the window is destroyed. If this is the word SHOW, the window is shown if it is currently hidden.</p> <p>Otherwise, this is the file name of the BMP, GIF, or JPG image to display (to display other file formats such as PNG, TIF, and ICO, consider using the Gui(See 19.3) command to create a window containing a picture control).</p> <p><i>ImageFile</i> is assumed to be in %A_WorkingDir%(See 9.) if an absolute path isn't specified. If <i>ImageFile</i> and <i>Options</i> are blank and the window already exists, its image will be unchanged but its text will be updated to reflect any new strings provided in <i>SubText</i>, <i>MainText</i>, and <i>WinTitle</i>.</p> <p>For newly created windows, if <i>ImageFile</i> is blank or there is a problem loading the image, the window will be displayed without the picture.</p>
ProgressParam1	<p><u>If the progress window already exists:</u> If <i>Param1</i> is the word OFF, the window is destroyed. If <i>Param1</i> is the word SHOW, the window is shown if it is currently hidden.</p>

	<p>Otherwise, if <i>Param1</i> is a pure number, its bar's position is changed to that value. If <i>Param1</i> is blank, its bar position will be unchanged but its text will be updated to reflect any new strings provided in <i>SubText</i>, <i>MainText</i>, and <i>WinTitle</i>. In both of these modes, if the window doesn't yet exist, it will be created with the defaults for all options.</p> <p><u>If the progress window does not exist:</u> A new progress window is created (replacing any old one), and <i>Param1</i> is a string of zero or more options from the list below.</p>
Options	A string of zero or more options from the list further below.
SubText	The text to display below the image or bar indicator. Although word-wrapping will occur, to begin a new line explicitly, use linefeed (`n). To set an existing window's text to be blank, specify %A_Space% (See 9.). For the purpose of auto-calculating the window's height, blank lines can be reserved in a way similar to <i>MainText</i> below.
MainText	<p>The text to display above the image or bar indicator (its font is semi-bold). Although word-wrapping will occur, to begin a new line explicitly, use linefeed (`n).</p> <p>If blank or omitted, no space will be reserved in the window for <i>MainText</i>. To reserve space for single line to be added later, or to set an existing window's text to be blank, specify %A_Space%(See 9.). To reserve extra lines beyond the first, append one or more linefeeds (`n).</p> <p>Once the height of <i>MainText</i>'s control area has been set, it cannot be changed without recreating the window.</p>
WinTitle	The title of the window. If omitted and the window is being newly created, the title defaults to the name of the script (without path). If the B (borderless) option has been specified, there will be no visible title bar but the window can still be referred to by this title in commands such as WinMove (See 28.27).
FontName	<p>The name of the font to use for both <i>MainText</i> and <i>SubText</i>. The font table (See 30.9) lists the fonts included with the various versions of Windows. If unspecified or if the font cannot be found, the system's default GUI font will be used.</p> <p>See the options section below for how to change the size, weight, and color of the font.</p>

A: The window will not be always-on-top.

B: Borderless: The window will have no border and no title bar. To have a border but no title bar, use **B1** for a thin border or **B2** for a dialog style border.

M: The window will be movable by the user (except if borderless). To additionally make the window resizable (by means of dragging its borders), specify **M1**. To additionally include a system menu and a set of minimize/maximize/close buttons in the title bar, specify **M2**.

Pn: For Progress windows, specify for **n** the starting position of the bar (the default is 0 or the number in the allowable range that is closest to 0). To later change the position of the bar, follow this example: Progress, 50

Rx-y: For Progress windows, specify for **x-y** the numerical range of the bar (if the **R** option is not present, the default is 0-100). For example, R0-1000 would allow numbers between 0 and 1000; R-50-50 would allow numbers between -50 and 50; and R-10--5 would allow numbers between -10 and -5. On Windows 95 and NT4, negative ranges and ranges beyond 65535 will not behave correctly unless Internet Explorer 3.0 or later is installed.

T: The window will be given a button in the task bar and it will be unowned. Normally, there is no button because the window is owned by the script's main window. This setting also prevents a GUI window's [Gui +OwnDialogs](#)(See 19.3) setting from taking ownership of a Splash or Progress window.

Hn: Specify for **n** the height of the window's client area (which is the area not including title bar and borders). If unspecified, the height will be calculated automatically based on the height of the image/bar and text in the window.

Wn: Specify for **n** the width of the window's client area. If unspecified, the width will be calculated automatically for SplashImage (if there is an image). Otherwise, it will default to 300.

Xn: Specify for **n** the x-coordinate of the window's upper left corner. If unspecified, the window will be horizontally centered on the screen.

Yn: Specify for **n** the y-coordinate of the window's upper left corner. If unspecified, the window will be vertically centered on the screen.

Hide: The window will be initially hidden. Use *Progress Show* or *SplashImage Show* to show it later.

Cxy: Centered: If this option is absent, both *SubText* and *MainText* will be centered inside the window. Specify 0 for **x** to make *SubText* left-justified. Specify a 1 to keep it centered. The same is true for **y** except that it applies to *MainText* (**y** can be omitted). For example: c10

ZHn: Height of object: For Progress windows, specify for **n** the thickness of the progress bar (default 20). For SplashImage windows, specify for **n** the height to which to scale image. Specify -1 to make the height proportional to the width specified in ZWn (i.e. "keep aspect ratio"). If unspecified, the actual image height will be used. In either case, a value of 0 can be

specified to omit the object entirely, which allows the window to be used to display only text in custom fonts, sizes, and colors.

ZWn: Width of object (for SplashImage windows only): Specify for **n** the width to which to scale the image. Specify -1 to make the width proportional to the height specified in ZHn (i.e. "keep aspect ratio"). If unspecified, the actual image width will be used.

ZXn: Specify for **n** the amount of margin space to leave on the left/right sides of the window. The default is 10 except for SplashImage windows with no text, which have a default of 0.

ZYn: Specify for **n** the amount of vertical blank space to leave at the top and bottom of the window and between each control. The default is 5 except for SplashImage windows with no text, which have a default of 0.

Note: To create a vertical progress bar or to have more layout flexibility, use the [Gui](#)(See 19.3) command such as this example:

```
Gui, Add, Progress, Vertical vMyProgress
Gui, Show
return
; ... later...
GuiControl,, MyProgress, +10 ; Move the bar upward by 10 percent. Omit the + to set
an absolute position.
```

FMn: Specify for **n** the font size for *MainText*. Default 0, which causes 10 to be used on most systems. This default is not affected by the system's selected font size in Control Panel > Display settings.

FSn: Specify for **n** the font size for *SubText*. Default 0, which causes 8 to be used on most systems.

WMn: Specify for **n** the font weight of *MainText*. The weight should be between 1 and 1000. If unspecified, 600 (semi-bold) will be used.

WSn: Specify for **n** the font weight of *SubText*. The weight should be between 1 and 1000 (700 is traditionally considered to be "bold"). If unspecified, 400 (normal) will be used.

A color can be one of the names from the list below or a 6-digit hexadecimal RGB value. For example, specifying cw1A00FF would set a window background color with red component 1A, green component 00, and blue component FF.

Add a space after each color option if there are any more options that follow it. For example:
cbRed ct900000 cwBlue

CBn: Color of progress bar: Specify for **n** one of the 16 primary HTML color names or a 6-digit RGB color value. If unspecified, the system's default bar color will be used. Specify the word Default to return to the system's default progress bar color.

CTn: Color of text: Specify for **n** one of the 16 primary HTML color names or a 6-digit RGB color value. If unspecified, the system's default text color (usually black) will be used. Specify the word Default to return to the system's default text color.

CWn: Color of window (background): Specify for **n** one of the 16 primary HTML color names or a 6-digit RGB color value. If unspecified, the system's color for the face of buttons will be used (specify the word Default to later return to this color). To make the background transparent on Windows 2000/XP or later, use [WinSet TransColor](#)(See 28.29).

Color names and RGB values

	Black = 000000		Green = 008000
	Silver = C0C0C0		Lime = 00FF00
	Gray = 808080		Olive = 808000
	White = FFFFFF		Yellow = FFFF00
	Maroon = 800000		Navy = 000080
	Red = FF0000		Blue = 0000FF
	Purple = 800080		Teal = 008080
	Fuchsia = FF00FF		Aqua = 00FFFF

If the first parameter is the word **OFF**, the window is destroyed.

Each script can display up to 10 Progress windows and 10 SplashImage windows. Each window has a number assigned to it at the time it is created. If unspecified, that number is 1 (the first). Otherwise, precede the first parameter with the number of the window followed by a colon. For example, the Progress command with 2:Off would turn off the number-2 Progress window,

2:75 would set its bar to 75%, 2: would change one or more of its text fields, and 2:B would create a new borderless Progress window. Similarly, the SplashImage command with 2:Off would turn off the number-2 SplashImage window, 2: would change one or more of its text fields, and 2:C:\My Images\Picture1.jpg would create a new number-2 SplashImage window.

Upon creation, a window that would be larger than the desktop is automatically shrunk to fit.

A naked progress bar can be displayed by specifying no *SubText*, no *MainText*, and including the following options: b zx0 zy0. A naked image can be displayed the same way except that only the B option is needed.

On Windows XP or later, if there is a non-Classic theme is in effect, the interior of a progress bar may appear as a series of segments rather than a smooth continuous bar. To avoid this, specify a bar color explicitly such as cbBlue.

Use decimal (not hexadecimal) numbers for option letters that need them, except where noted.

Commands such as [WinSet](#)(See 28.29) and [WinMove](#)(See 28.27) can be used to change the attributes of an existing window without having to recreate it.

A GUI window may take ownership of a Progress or Splash window by means of [Gui +OwnDialogs](#)(See 19.3). This causes the Splash or Progress to stay always on top of its owner. In addition, the Progress or Splash is automatically destroyed when its GUI window is destroyed.

[GUI](#)(See 19.3), [SplashTextOn](#)(See 19.14), [ToolTip](#)(See 19.15)

```
Progress, b w200, My SubText, My MainText, My Title
```

```
Progress, 50 ; Set the position of the bar to 50%.
```

```
Sleep, 4000
```

```
Progress, Off
```

```
; Create a window just to display some 18-point Courier text:
```

```
Progress, m2 b fs18 zh0, This is the Text.`nThis is a 2nd line., , , Courier New
```

```
; Create a simple SplashImage window:
```

```
SplashImage, C:\My Pictures\Company Logo.gif
```

```

; Create a borderless SplashImage window with some large text beneath the image:

SplashImage, C:\My Pictures\Company Logo.gif, b fs18, This is our company logo.

Sleep, 4000

SplashImage, Off


; Here is a working example that demonstrates how a Progress window can be
; overlayed on a SplashImage to make a professional looking Installer screen:

IfExist, C:\WINDOWS\system32\ntimage.gif,
SplashImage, %A_WinDir%\system32\ntimage.gif, A,,, Installation

Loop, %A_WinDir%\system32\*.*

{
    Progress, %a_index%, %a_loopfilename%, Installing..., Draft Installation

    Sleep, 50

    if a_index = 100
        break
}

; There is similar example at the bottom of the GUI page(See 19.3). Its advantage is that
; it uses only a single

; window and it gives you more control over window layout.

```

19.13 SplashImage

Creates or updates a window containing a progress bar or an image.

SplashImage, Off

SplashImage [, ImageFile, Options, SubText, MainText, WinTitle, FontName]

Progress, Off

Progress, ProgressParam1 [, SubText, MainText, WinTitle, FontName]

参数

ImageFile	If this is the word OFF, the window is destroyed. If this is the word SHOW, the window is shown if it is currently hidden. Otherwise, this is the file name of the BMP, GIF, or JPG image to display (to
-----------	---

	<p>display other file formats such as PNG, TIF, and ICO, consider using the Gui(See 19.3) command to create a window containing a picture control).</p> <p><i>ImageFile</i> is assumed to be in %A_WorkingDir%(See 9.) if an absolute path isn't specified. If <i>ImageFile</i> and <i>Options</i> are blank and the window already exists, its image will be unchanged but its text will be updated to reflect any new strings provided in <i>SubText</i>, <i>MainText</i>, and <i>WinTitle</i>.</p> <p>For newly created windows, if <i>ImageFile</i> is blank or there is a problem loading the image, the window will be displayed without the picture.</p>
ProgressParam1	<p><u>If the progress window already exists</u>: If <i>Param1</i> is the word OFF, the window is destroyed. If <i>Param1</i> is the word SHOW, the window is shown if it is currently hidden.</p> <p>Otherwise, if <i>Param1</i> is a pure number, its bar's position is changed to that value. If <i>Param1</i> is blank, its bar position will be unchanged but its text will be updated to reflect any new strings provided in <i>SubText</i>, <i>MainText</i>, and <i>WinTitle</i>. In both of these modes, if the window doesn't yet exist, it will be created with the defaults for all options.</p> <p><u>If the progress window does not exist</u>: A new progress window is created (replacing any old one), and <i>Param1</i> is a string of zero or more options from the list below.</p>
Options	A string of zero or more options from the list further below.
SubText	The text to display below the image or bar indicator. Although word-wrapping will occur, to begin a new line explicitly, use linefeed (`n). To set an existing window's text to be blank, specify %A_Space% (See 9.). For the purpose of auto-calculating the window's height, blank lines can be reserved in a way similar to <i>MainText</i> below.
MainText	<p>The text to display above the image or bar indicator (its font is semi-bold). Although word-wrapping will occur, to begin a new line explicitly, use linefeed (`n).</p> <p>If blank or omitted, no space will be reserved in the window for <i>MainText</i>. To reserve space for single line to be added later, or to set an existing window's text to be blank, specify %A_Space%(See 9.). To reserve extra lines beyond the first, append one or more linefeeds (`n).</p> <p>Once the height of <i>MainText</i>'s control area has been set, it cannot be changed without recreating the window.</p>
WinTitle	The title of the window. If omitted and the window is being newly created, the title defaults to the name of the script (without path). If the B (borderless)

	option has been specified, there will be no visible title bar but the window can still be referred to by this title in commands such as WinMove (See 28.27).
FontName	<p>The name of the font to use for both <i>MainText</i> and <i>SubText</i>. The font table (See 30.9) lists the fonts included with the various versions of Windows. If unspecified or if the font cannot be found, the system's default GUI font will be used.</p> <p>See the options section below for how to change the size, weight, and color of the font.</p>

A: The window will not be always-on-top.

B: Borderless: The window will have no border and no title bar. To have a border but no title bar, use **B1** for a thin border or **B2** for a dialog style border.

M: The window will be movable by the user (except if borderless). To additionally make the window resizable (by means of dragging its borders), specify **M1**. To additionally include a system menu and a set of minimize/maximize/close buttons in the title bar, specify **M2**.

Pn: For Progress windows, specify for **n** the starting position of the bar (the default is 0 or the number in the allowable range that is closest to 0). To later change the position of the bar, follow this example: Progress, 50

Rx-y: For Progress windows, specify for **x-y** the numerical range of the bar (if the **R** option is not present, the default is 0-100). For example, R0-1000 would allow a numbers between 0 and 1000; R-50-50 would allow numbers between -50 and 50; and R-10--5 would allow numbers between -10 and -5. On Windows 95 and NT4, negative ranges and ranges beyond 65535 will not behave correctly unless Internet Explorer 3.0 or later is installed.

T: The window will be given a button in the task bar and it will be unowned. Normally, there is no button because the window is owned by the script's main window. This setting also prevents a GUI window's [Gui +OwnDialogs](#)(See 19.3) setting from taking ownership of a Splash or Progress window.

Hn: Specify for **n** the height of the window's client area (which is the area not including title bar and borders). If unspecified, the height will be calculated automatically based on the height of the image/bar and text in the window.

Wn: Specify for **n** the width of the window's client area. If unspecified, the width will be calculated automatically for SplashScreen (if there is an image). Otherwise, it will default to 300.

Xn: Specify for **n** the x-coordinate of the window's upper left corner. If unspecified, the window will be horizontally centered on the screen.

Yn: Specify for **n** the y-coordinate of the window's upper left corner. If unspecified, the window will be vertically centered on the screen.

Hide: The window will be initially hidden. Use *Progress Show* or *SplashImage Show* to show it later.

Cxy: Centered: If this option is absent, both *SubText* and *MainText* will be centered inside the window. Specify 0 for **x** to make *SubText* left-justified. Specify a 1 to keep it centered. The same is true for **y** except that it applies to *MainText* (**y** can be omitted). For example: c10

ZHn: Height of object: For Progress windows, specify for **n** the thickness of the progress bar (default 20). For SplashImage windows, specify for **n** the height to which to scale image. Specify -1 to make the height proportional to the width specified in ZWn (i.e. "keep aspect ratio"). If unspecified, the actual image height will be used. In either case, a value of 0 can be specified to omit the object entirely, which allows the window to be used to display only text in custom fonts, sizes, and colors.

ZWn: Width of object (for SplashImage windows only): Specify for **n** the width to which to scale the image. Specify -1 to make the width proportional to the height specified in ZHn (i.e. "keep aspect ratio"). If unspecified, the actual image width will be used.

ZXn: Specify for **n** the amount of margin space to leave on the left/right sides of the window. The default is 10 except for SplashImage windows with no text, which have a default of 0.

ZYn: Specify for **n** the amount of vertical blank space to leave at the top and bottom of the window and between each control. The default is 5 except for SplashImage windows with no text, which have a default of 0.

Note: To create a vertical progress bar or to have more layout flexibility, use the [Gui\(See 19.3\)](#) command such as this example:

```
Gui, Add, Progress, Vertical vMyProgress
Gui, Show
return
; ... later...
GuiControl,, MyProgress, +10 ; Move the bar upward by 10 percent. Omit the + to set
an absolute position.
```

FMn: Specify for **n** the font size for *MainText*. Default 0, which causes 10 to be used on most systems. This default is not affected by the system's selected font size in Control Panel > Display settings.

FSn: Specify for **n** the font size for *SubText*. Default 0, which causes 8 to be used on most systems.

WMn: Specify for **n** the font weight of *MainText*. The weight should be between 1 and 1000. If unspecified, 600 (semi-bold) will be used.

WSn: Specify for **n** the font weight of *SubText*. The weight should be between 1 and 1000 (700 is traditionally considered to be "bold"). If unspecified, 400 (normal) will be used.

A color can be one of the names from the list below or a 6-digit hexadecimal RGB value. For example, specifying cw1A00FF would set a window background color with red component 1A, green component 00, and blue component FF.

Add a space after each color option if there are any more options that follow it. For example:
cbRed ct900000 cwBlue

CBn: Color of progress bar: Specify for **n** one of the 16 primary HTML color names or a 6-digit RGB color value. If unspecified, the system's default bar color will be used. Specify the word Default to return to the system's default progress bar color.

CTn: Color of text: Specify for **n** one of the 16 primary HTML color names or a 6-digit RGB color value. If unspecified, the system's default text color (usually black) will be used. Specify the word Default to return to the system's default text color.

CWn: Color of window (background): Specify for **n** one of the 16 primary HTML color names or a 6-digit RGB color value. If unspecified, the system's color for the face of buttons will be used (specify the word Default to later return to this color). To make the background transparent on Windows 2000/XP or later, use [WinSet TransColor](#)(See 28.29).

Color names and RGB values



Black = 000000



Green = 008000



Silver = C0C0C0



Lime = 00FF00



Gray = 808080



Olive = 808000



White = FFFFFF



Yellow = FFFF00



Maroon = 800000



Navy = 000080



Red = FF0000



Blue = 0000FF



Purple = 800080



Teal = 008080



Fuchsia = FF00FF



Aqua = 00FFFF

If the first parameter is the word **OFF**, the window is destroyed.

Each script can display up to 10 Progress windows and 10 SplashImage windows. Each window has a number assigned to it at the time it is created. If unspecified, that number is 1 (the first). Otherwise, precede the first parameter with the number of the window followed by a colon. For example, the Progress command with 2:Off would turn off the number-2 Progress window, 2:75 would set its bar to 75%, 2: would change one or more of its text fields, and 2:B would create a new borderless Progress window. Similarly, the SplashImage command with 2:Off would turn off the number-2 SplashImage window, 2: would change one or more of its text fields, and 2:C:\My Images\Picture1.jpg would create a new number-2 SplashImage window.

Upon creation, a window that would be larger than the desktop is automatically shrunk to fit.

A naked progress bar can be displayed by specifying no *SubText*, no *MainText*, and including the following options: b zx0 zy0. A naked image can be displayed the same way except that only the B option is needed.

On Windows XP or later, if there is a non-Classic theme is in effect, the interior of a progress bar may appear as a series of segments rather than a smooth continuous bar. To avoid this, specify a bar color explicitly such as cbBlue.

Use decimal (not hexadecimal) numbers for option letters that need them, except where noted.

Commands such as [WinSet\(See 28.29\)](#) and [WinMove\(See 28.27\)](#) can be used to change the attributes of an existing window without having to recreate it.

A GUI window may take ownership of a Progress or Splash window by means of [Gui +OwnDialogs\(See 19.3\)](#). This causes the Splash or Progress to stay always on top of its owner. In addition, the Progress or Splash is automatically destroyed when its GUI window is destroyed.

[GUI\(See 19.3\)](#), [SplashTextOn\(See 19.14\)](#), [ToolTip\(See 19.15\)](#)

Progress, b w200, My SubText, My MainText, My Title

Progress, 50 ; Set the position of the bar to 50%.

```

Sleep, 4000

Progress, Off

; Create a window just to display some 18-point Courier text:

Progress, m2 b fs18 zh0, This is the Text.`nThis is a 2nd line., , , Courier New

; Create a simple SplashImage window:

SplashImage, C:\My Pictures\Company Logo.gif

; Create a borderless SplashImage window with some large text beneath the image:

SplashImage, C:\My Pictures\Company Logo.gif, b fs18, This is our company logo.

Sleep, 4000

SplashImage, Off

; Here is a working example that demonstrates how a Progress window can be
; overlaid on a SplashImage to make a professional looking Installer screen:

IfExist, C:\WINDOWS\system32\ntimage.gif,
SplashImage, %A_WinDir%\system32\ntimage.gif, A,,, Installation

Loop, %A_WinDir%\system32\*.*

{
    Progress, %a_index%, %a_loopfilename%, Installing..., Draft Installation

    Sleep, 50

    if a_index = 100
        break
}

; There is similar example at the bottom of the GUI page(See 19.3). Its advantage is that
; it uses only a single

; window and it gives you more control over window layout.

```

19.14 SplashTextOn/SplashTextOff

创建可定制的文本弹出窗口。

SplashTextOff

SplashTextOn [, Width, Height, Title, Text]

参数

Width	窗口的像素级宽度。默认是 200。此参数可以是一个 表达式(See 9.) 。
Height	窗口的像素级高度 (如果脚本的文件扩展名不是 .aut, 那么不包括它的标题栏)。默认是 0 (也就是只显示标题栏)。此参数可以是一个 表达式(See 9.) 。
Title	窗口的标题。默认是空的 (空白)。
Text	窗口的文本。默认是空的 (空白)。如果 <i>Text</i> 很长, 它能通过一种 continuation section(See 8.) (连续章节) 的方法被分解为多个更短的行, 这样可以改善可读性和可维护性。

注意

要进一步控制布局和字体名称/颜色/大小, 请用 [Progress\(See 19.12\)](#) 命令和 zh0 选项, 其省略栏目而只显示文本。例如: Progress, zh0 fs18, 显示一些 18-point(磅) 的文本。

可用 **SplashTextOff** 命令来消除已存在的弹出窗口。

弹出窗口是“总在最上面”的, 意味着它停留在其他所有普通窗口的上面。要改变这种状态, 可用 [WinSet\(See 28.29\)](#), *AlwaysOnTop, Off, <插入弹出窗口的标题>*。[WinSet\(See 28.29\)](#) 同样能使弹出窗口变透明。

在此命令显示弹出文本窗口后 [WinMove\(See 28.27\)](#) 可被用来将其重新定位以及调整大小。

和 [Progress\(See 19.12\)](#), [SplashImage\(See 19.12\)](#), [MsgBox\(See 19.11\)](#), [InputBox\(See 19.10\)](#), [FileSelectFile\(See 16.23\)](#) 以及 [FileSelectFolder\(See 16.24\)](#) 不同, 每个脚本只能有一个弹出文本窗口。

如果在弹出窗口已经显示时使用 **SplashTextOn**, 窗口将用新的参数值来重新创建。不过, 与其在你想要改变弹出窗口的标题或文本时去重新创建它, 倒不如用下面的方法来获得更好的性能, 特别是在窗口需要被频繁地改变时:

[WinSetTitle\(See 28.30\)](#), <插入弹出窗口的标题>, , 新标题

[ControlSetText\(See 28.1.10\)](#), Static1, 新文本, <插入弹出窗口的标题>

相关命令

[Progress\(See 19.12\)](#), [SplashImage\(See 19.12\)](#), [ToolTip\(See 19.15\)](#), [MsgBox\(See 19.11\)](#), [InputBox\(See 19.10\)](#), [FileSelectFile\(See 16.23\)](#), [FileSelectFolder\(See 16.24\)](#), [WinMove\(See 28.27\)](#), [WinSet\(See 28.29\)](#)

示例

SplashTextOn, , , 仅显示标题栏。

Sleep, 2000

```
SplashTextOn, 400, 300, Clipboard, 剪贴板包含: `n%clipboard%
WinMove, Clipboard, , 0, 0 ; 移动弹出窗口到左上角。
```

Msgbox, 按确定后消除弹出文本

SplashTextOff

翻译: 天堂之门 menk33@163.com 2008 年 9 月 13 日

19.15 ToolTip

创建一个在屏幕任何位置上总在最前端的窗口。

ToolTip [, Text, X, Y, WhichToolTip]

参数

Text	如果留空或省略, 已存在的 tooltip (如果有的话) 将被隐藏。否则, 此参数是要显示在 tooltip 中的文本。要创建一个多行 tooltip , 在每行中使用换行符(`n), 例如 行 1`n 行 2 。
X, Y	相对于激活的窗口的 tooltip 的 X 和 Y 坐标位置(使用 "CoordMode(See 22.6), ToolTip" 来改变成屏幕坐标)。如果省略坐标, tooltip 将显示在鼠标指针附近。X 和 Y 坐标可以是 expressions(See 9.) (表达式) 。
WhichToolTip	如果你不需要同时显示多个 tooltips , 省略此参数。否则, 在 1 到 20 之间的一个数字指示哪个 tooltip 窗口运作在上面。如果未指定, 这数字是 1 (首个)。此参数可以是一个 expression(See 9.) 。

注意

如果 X 和 Y 坐标会导致 tooltip 脱离屏幕, 它会重新定位来使其整个可见。

tooltip 会显示, 直到发生了下列情况之一:

- 脚本终止。
- ToolTip 命令用一个空的 Text 参数再次执行。
- 用户在 tooltip 上点击 (此特性会根据操作系统的版本而出现不同)。

一个 GUI(图形用户接口) 窗口可通过 [Gui +OwnDialogs\(See 19.3\)](#) 来成为一个 tooltip 的主人。这样的一个 tooltip 会在它的主人毁灭后自动地毁灭。

相关命令

[CoordMode](#)(See 22.6), [TrayTip](#)(See 19.16), [GUI](#)(See 19.3), [Progress](#)(See 19.12),
[SplashTextOn](#)(See 19.14), [MsgBox](#)(See 19.11), [InputBox](#)(See 19.10), [FileSelectFile](#)(See
16.23), [FileSelectFolder](#)(See 16.24)

示例

```
ToolTip, 多行`nTooltip, 100, 150

; 要使一个 ToolTip 不使用 Sleep(其停止当前线程) 而在某个时间段后消失:

#Persistent

ToolTip, 定时的 ToolTip`n 它将显示 5 秒。

SetTimer, RemoveToolTip, 5000

return

RemoveToolTip:

SetTimer, RemoveToolTip, Off

ToolTip

return
```

翻译：天堂之门 menk33@163.com 2008 年 8 月 12 日

19.16 TrayTip

在托盘图标处创建一个气泡消息窗口。要求 Windows 2000/XP 或以上操作系统。

TrayTip [, Title, Text, Seconds, Options]

参数

Title	如果省略所有参数，当前显示的任何气泡消息窗口将被移除。 否则，此参数是窗体的标题，最长 73 字符（超出此长度将不显示）。 如果 <i>Title</i> 为空，标题行将完全不显示，使窗口垂直距离缩短。
Text	如果省略此参数或留空，当前显示的任何气泡消息窗口将被移除。 否则，指定要显示的信息，会在 <i>Title</i> 下方显示。只显示 <i>Text</i> 的前 265 字符。回车符(`r) 或换行符(`n)可以被用来创建多行文本。例如：行 1`n 行 2

	如果 <i>Text</i> 较长，可通过 continuation section (See 8.)(连 续章节) 分割成若干较短的行，来提高程序的可读性和可维护性。
Seconds	<p>窗体显示的大概的秒数，之后它将被操作系统自动地移除。指定一个小于 10 或大于 30 的数字，通常会被最小值 (10) 或最大值 (30) 的显示时间代替。如果省略或留空，通常会采用最长时间。此参数可以是 expression(See 9.)(表达 式)。</p> <p>真实的时间可能不同于指定的时间。Microsoft 解释说，“如果没有显示用户正在使用计算机，系统会不将此段时间计入。”(技术细节请看 这 里。)因此，为了精确控制托盘窗口的显示，可使用 Sleep(See 17.22) 命令紧跟不带参数的 TrayTip，或像下方示例中使用 SetTimer(See 17.21)。</p>
Options	<p>指定一下数字中的一个，会在 <i>Title</i> 左侧显示小图标：</p> <ul style="list-style-type: none"> 1: 消息图标 2: 警告图标 3: 错误图标 <p>如果省略，默认为 0，无图标显示。此参数可以是 expression(See 9.)。</p>

注意

TrayTip 要求 Windows 2000/XP 或以上的操作系统。在 Windows 9x/NT 上此命令不起作用。

如果脚本没有托盘图标(通过 [#NoTrayIcon](#)(See 22.1) 或 [Menu](#)(See 22.13), *tray, NoIcon*)，那么 **TrayTip** 气泡窗体不能被显示。同样地，如果下列的 REG_DWORD 值已存在，并已被设为 0，那么 **TrayTip** 不会起作用：

HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced >> EnableBalloonTips

当显示一个气泡窗体时，操作系统通常会播放一个声音。可通过加上 16 到 *Options* 参数来关闭此声音(例如 16 是无声音无图标，17 是无声音有信息图标，等等。)

做一个相关提示，每当用户将鼠标悬停在脚本托盘图标上时，会有提示显示。提示内容可以通过：[Menu](#)(See 22.13), *Tray, Tip, 我的新文本* 来改变。

相关命令

[ToolTip](#)(See 19.15), [SetTimer](#)(See 17.21), [Menu](#)(See 22.13), [SplashTextOn](#)(See 19.14),
[MsgBox](#)(See 19.11), [InputBox](#)(See 19.10), [FileSelectFile](#)(See 16.23), [FileSelectFolder](#)(See 16.24)

示例

```
TrayTip, My Title, Multiline`nText, 20, 17
```

; 为了更精确的控制显示时间

```
; 而不使用 Sleep (其会停止当前线程):
#Persistent
TrayTip, Timed TrayTip, 此文本将被显示 5 秒。
SetTimer, RemoveTrayTip, 5000
return
RemoveTrayTip:
SetTimer, RemoveTrayTip, Off
TrayTip
return
; 为了让 TrayTip 永久显示, 使用 timer 来周期性地进行刷新:
SetTimer, RefreshTrayTip, 1000
Gosub, RefreshTrayTip ; 调用一次来让它马上开始。
return
RefreshTrayTip:
TrayTip, Refreshed TrayTip, 这是一个更持久的 TrayTip . , , 16
return
```

翻译: Xiangee 修正: 天堂之门 menk33@163.com 2008 年 8 月 7 日

20. 键盘控制

20.1 Hotkeys and Hotstrings

- 介绍和简单的例子
- 热键前缀符号(修饰键)
- 上下文相关的热键
- 自定义组合键和其他特性
- 鼠标滚轮热键
- 热键技巧和备注

热键有时也被称作快捷键, 因为它们能轻易地触发一个动作(例如启动一个程序或[键盘宏](#)(See 30.11))。在下面的例子中, 热键 Win+N 被设定为启动记事本。[pound sign](#) [#] 表示 Windows 键, 其被称作修饰键:

```
#n::  
Run Notepad  
return
```

在上面的最后一行, "[return](#)(See 17.19)" 用来结束热键。不过, 如果一个热键仅仅需要执行一行, 那么这行可以列在双冒号的右边。换句话说, [return](#)(See 17.19) 可以省略:

```
#n::Run Notepad
```

要对一个热键使用多个修饰键，把它们连续地列出来(顺序没有关系)。下面的例子用 **^!s** 来表示 Control+Alt+S:

```
^!s::  
Send(See 20.12) Sincerely,{enter}John Smith ;这行发送键击到激活的(最前面的)窗口。  
return
```

你可以使用下列的修饰键符号来定义热键：

符号	描述
#	Win (Windows 标识键)
!	Alt
^	Control
+	Shift
&	一个连接符可以用在任何两个键或者鼠标按键的中间，从而将它们组合成一个自定义热键。详见 下面 。这样的热键在 Windows 95/98/Me 上被忽略(未被激活)。
<	使用成对按键中左边的按键。例如 <!a 和 !a 一样，只是仅有左边的 Alt 键会触发它。这个符号在 Windows 95/98/ME 上被忽略。
>	使用成对按键中右边的按键。这个符号在 Windows 95/98/ME 上被忽略。
<^>!	<p>AltGr (alternate graving)(译注：传说此键出现在大部分欧洲键盘上)。如果你的键盘布局中有一个代替右 Alt 键的 AltGr 键，这一系列的符号一般能用来表示 AltGr (需要 Windows NT/2k/XP 或之后版本)。例如：</p> <pre><^>!m::MsgBox 你按下了 AltGr+m. <^<!m::MsgBox 你按下了 LeftControl+LeftAlt+m.</pre> <p>或者，要使 AltGr 它自己成为一个热键，使用下面的热键(不用像上面出现的任何一个热键)：</p> <pre>LControl & RAlt::MsgBox 你按了 AltGr 它本身。</pre>
*	<p>通配符：即使附加的修饰键被按住也激发热键。这常被用来协同重映射(See 6.)按键或按钮。例如：</p> <pre>*#c::Run Calc.exe ; Win+C, Shift+Win+C, Ctrl+Win+C 等全部会触发这个热键。 *ScrollLock::Run Notepad ;即使当修饰键按住时，按下 Scrolllock 键也将触发这个热键。</pre> <p>这个符号在 Windows 95/98/ME 上被忽略。</p>
~	<p>当激发热键时，按键的原来的功能不会被屏蔽(被操作系统隐藏)。在下面的两个例子中，用户的鼠标按钮点击将被发送到激活的窗口：</p> <pre>~RButton::MsgBox 你点击了鼠标右键。</pre>

	<p>~RButton & C::MsgBox 你在按住鼠标右键的同时按下了 C 键。</p> <p>注意: 1) 和其他前缀符号不同, 波浪符前缀被允许在一个热键的某些变体(See 20.1.4)上出现但在其他部分不存在; 2) 那些代替 alt-tab 的特殊热键永远忽略波浪符前缀; 3) 波浪符前缀在 Windows 95/98/ME 上被忽略。</p>
\$	<p>这符号通常仅仅在脚本使用 Send(See 20.12) 命令发送包含了热键自身的按键时才有必要使用, 否则这可能导致发送热键触发它自己。\$ 前缀确切的特性变化取决于操作系统:</p> <p>在 Windows NT4/2k/XP 或之后版本上: \$ 前缀强制使用键盘钩子(See 20.2)来执行这个热键, 作为一个副作用它防止了 Send(See 20.12) 命令触发热键。\$ 前缀等效于在此热键定义的上面某处指定了 #UseHook(See 20.1.9)。</p> <p>在 Windows 95/98/Me 上: 热键在它的线程(See 30.19)执行期间被禁用, 之后重新启用。作为一个副作用, 如果 #MaxThreadsPerHotkey(See 20.1.8) 设成高于 1, 它也将表现得像被这个热键设为 1 一样。</p>
UP	<p>单词 UP 可以跟在一个热键名称后面来促使热键在松开按键时激发而不是在按下按键时。下面的例子将 LWin 重映射(See 6.)成 LControl:</p> <pre>*LWin::Send {LControl Down} *LWin Up::Send {LControl Up}</pre> <p>"Up" 也可以和一般的热键一起使用比如在这个例子中: ^!r Up::MsgBox 你按下并松开了 Ctrl+Alt+R 。它也可以和 组合热键 一起用(例如 F1 & e Up::)</p> <p>限制: 1) "Up" 不能和游戏杆按钮(See 7.)一起用; 2) "Up" 需要 Windows NT4/2000/XP 或之后版本; 3) 不和一个普通/按下的配对热键一起用的 "Up" 热键将完全接管那个热键从而防止它卡在按下的状态。唯一能防止这种情况发生的就是添加一个波浪符前缀 (例如 ~LControl up::)</p> <p>作个相关的提示, 和上面相似的一个技术是让一个热键成为一个前缀键。尽管热键会在松开时激发是个有利情况, 但热键仅在你如果按住它的同时不去按其他任何键的时候才会激发。例如:</p> <p>LControl & F1::return ;通过让左 control 键在 "&" 前面至少使用一次来把它变成一个前缀。</p> <p>LControl::MsgBox 你在没有用 LControl 去修饰其他任何键的情况下松开了它。</p>

(请看[按键列表](#)(See 7.)获取一个完整的键盘按键和鼠标/操纵杆按钮列表)

多个热键可以被垂直地堆放起来让它们执行同样的动作。例如:

```
^Numpad0::
^Numpad1::
MsgBox 按了 Control+Numpad0 或 Control+Numpad1 都会显示这个消息。
return
```

通过让热键不作任何动作可以在整个操作系统中禁用一个键或者组合键。下面的例子禁用了右边的 Windows 键：

```
RWin::return
```

#IfWinActive/Exist(See 20.1.4) 指令可以被用来让一个热键视激活或存在的窗口的类型而执行一个不同的动作(或根本无动静)。例如：

```
#IfWinActive, ahk_class Notepad
^a::MsgBox 你在记事本激活时按了 Ctrl-A 。在其他任何窗口按 Ctrl-A 将只传递 Ctrl-A 键击到那个窗口。
#c::MsgBox 你在记事本激活时按了 Win-C 。
#IfWinActive
#c::MsgBox 你在除了记事本外的任意窗口激活时按了 Win-C 。
```

你可以通过在两个键之间使用 " & " 来自定义一个组合键(除了操纵杆按钮)。在下面的例子中，你能够按住 Numpad0 之后再按第二个键来触发热键：

```
Numpad0 & Numpad1::MsgBox 你在按住 Numpad0 的同时按下了 Numpad1 。
Numpad0 & Numpad2::Run Notepad
```

在上面的例子中，Numpad0 变成一个**前缀键**；而且也导致了通过它去按 Numpad0 它自己的时候失去了它原本/天生的功能。要避免这种情况，脚本可以像下面中的一种那样来配置 Numpad0 去执行一个新的动作：

```
Numpad0::WinMaximize A ;最大化激活的/前台的窗口。
Numpad0::Send {Numpad0} ;让 Numpad0 松开后去再生一个 Numpad0 键击。请见下面的内容。
```

上面的热键之一的存在都促使 Numpad0 松开后去执行指示的动作，但仅在 Numpad0 被按住时如果你没有去按其他任何键的情况下。

Numlock, Capslock 和 Scrolllock: 这些按键可以被强制变成 "AlwaysOn" 或 "AlwaysOff"。例如：
SetNumlockState(See 20.15) *AlwaysOn*

覆盖 Explorer 的热键: Windows 的内置热键例如 Win-E (#e) 和 Win-R (#r) 可以通过在脚本里将它们指定为一个动作来简单地单独覆盖。详见**覆盖页面**(See 30.12)。

替换 Alt-Tab: 热键能通过 alt-tab 提供一种交替的效果。例如，下面的两个热键可让你用右手来 alt-tab：

```
RControl & RShift::AltTab ;按住右-control 然后反复地按右-shift 来向前移动。
RControl & Enter::ShiftAltTab ;甚至不需要释放右-control，直接按 Enter 键来反向移动。
```

详见 [Alt-Tab](#)。

通过键名 `WheelDown` 和 `WheelUp` 可以支持在调节鼠标滚轮的时候激发热键。`WheelLeft` 和 `WheelRight` 需要 v1.0.48+ 以上支持，但是在 Windows Vista 或以上没影响。以下是 鼠标滚轮热键的例子：

```
MButton & WheelDown::MsgBox 你在按住鼠标中键的时候向后滚动了滚轮。
^!WheelUp::MsgBox 你在按住 Control+Alt 的时候向前旋转了滚轮。
```

在 v1.0.43.03+，内置变量 `A_EventInfo` 包含了通过调节滚轮得到的触点次数，它一般情况是 1。然而，在下面的情况下，`A_EventInfo` 可能大于或小于 1：如果鼠标硬件报告的距离不足一个刻痕(你滚动滚轮感觉到的一下)，`A_EventInfo` 可能包含 0；当滚轮被快速地滚动(取决于鼠标类型)，`A_EventInfo` 可能大于 1。例子用法：`~WheelDown::ToolTip %A_EventInfo%`

鼠标滚轮的一些最有用的热键涉及交替窗口的文本滚动模式。例如，下面的一对热键在你按住左边的 `Control` 键并调节滚轮时用水平地滚动代替了垂直地滚动：

```
~LControl & WheelUp:: ;向左滚动。
ControlGetFocus, fcontrol, A
Loop 2 ;<-- 调大这个数值来快速滚动。
SendMessage, 0x114, 0, 0, %fcontrol%, A ;0x114 是 WM_HSCROLL, 它之后的 0 是
SB_LINELEFT。
return

~LControl & WheelDown:: ;向右滚动。
ControlGetFocus, fcontrol, A
Loop 2 ;<-- 调大这个数值来快速滚动。
SendMessage, 0x114, 1, 0, %fcontrol%, A ;0x114 是 WM_HSCROLL, 它之后的 1 是
SB_LINERIGHT。
return
```

最后提下，由于鼠标滚轮热键只产生按下事件(从来没有弹起事件)，它们不能被用作弹起的按键热键。

依靠 `Numlock` 的状态，每个数字键区的按键能启动两种不同的热键子程序。或者，一个数字键区按键能不管 `Numlock` 的状态而启动相同的子程序。例如：

```
NumpadEnd::
Numpad1::
MsgBox, 不管 Numlock 是否打开，这个热键都将被启用。
return
```

如果波浪操作符 (~) 和一个前缀键即使一起使用了一次，前缀键也总是会被发送到激活的窗口。例如，下面两个热键中，激活的窗口会接收到所有的右键点击即使只有一个热键定义包含了波浪符：

`~RButton & LButton::MsgBox` 你在按住鼠标右键的同时按了鼠标左键。

`RButton & WheelUp::MsgBox` 你在按住鼠标右键的同时向前推动了滚轮。

Suspend(See 17.23) 命令能临时禁用所有热键除了你要免除的那些外。要得到更多的选择性，使用 **#IfWinActive/Exist(See 20.1.4)**。

通过使用 **Hotkey(See 20.1.10)** 命令，热键能在脚本运行过程中被动态地创建。 **Hotkey** 命令也能单独地修改、禁用或启用已存在的脚本热键。

操纵杆热键目前不支持修饰键前缀例如 `^ (Control)` 和 `# (Win)`。不过，你可以使用 **GetKeyState(See 20.7)** 模仿下面例子中展示的这个效果：

```
Joy2::  
if not GetKeyState("Control");左边或右边的 Control 都没有按下。  
    return ;也就是什么也不做。  
 MsgBox 你在按住 Control 键的同时按了首个操纵杆的第二个按钮。  
return
```

当一个热键在继续前要等它的修饰键被松开的时候可能需要一些时间。参考下面的例子：

```
^!s::Send {Delete}
```

按下 `Control-Alt-S` 会导致系统表现得像你按了 `Control-Alt-Delete` 一样(由于系统的侵略性的 `Ctrl-Alt-Delete` 探测)。要绕弯解决这种情况，使用 **KeyWait(See 20.10)** 来等待那些键被松开；例如：

```
^!s::  
KeyWait Control  
KeyWait Alt  
Send {Delete}  
return
```

一个热键标记能被用作 **Gosub(See 17.8)** 或 **Goto(See 17.9)** 要跳转的目标。例如： `Gosub ^!s`

一个热键常用来开始和结束一个重复的动作，比如一系列的键击或鼠标点击。要得到这样一个例子，请看这个 **FAQ 主题(See 3.)**。

最后，每个脚本都是**类似的多线程(See 30.19)**，其允许在前一个热键子程序还在运行时，一个新的热键也能被启动。例如，即使当一个 **MessageBox(See 19.11)** 被当前热键调用显示时，新的热键也能被启动。

每个 `Alt-Tab` 热键必须是一个双键组合，其往往通过连接符 (`&`) 实现。在下面的例子中，你将按住右边的 `Alt` 键并按下 `J` 或 `K` 来浏览 `alt-tab` 菜单：

```
RAlt & j::AltTab  
RAlt & k::ShiftAltTab
```

`AltTab` 和 `ShiftAltTab` 是两个特殊的命令，它们仅当和热键使用在同一行时才被识别。这里有一个完整的列表：

AltTab: 如果 `alt-tab` 菜单可见，在菜单中前移。否则，显示菜单(仅在热键是一个用 "&" 连接的双键组合才行；否则，它什么也不做)。

ShiftAltTab: 和上面的一样，除了在菜单中是向后移动。

AltTabAndMenu: 如果 `alt-tab` 菜单可见，在菜单中前移。否则，显示菜单。

AltTabMenuDismiss: 关闭 `Alt-tab` 菜单。

要阐明上面的命令，鼠标滚轮可以来整个代替 `Alt-tab`。让下面的热键生效时，点击鼠标中键显示菜单并且调节滚轮可在它里面来导航：

```
MButton::AltTabMenu
WheelDown::AltTab
WheelUp::ShiftAltTab
```

要取消一个热键调用的 `Alt-tab` 菜单而不激活选中的窗口，使用一个像下面那样的热键。它可能需要依赖后面的条件来调整：1) 借助原本显示的 `alt-tab` 菜单；2) 脚本是否安装了[键盘钩子](#)(See 20.2)。

```
LCtrl & CapsLock::AltTab
!MButton:: ;鼠标中键。前缀 ! 使它在按住 Alt 键时激发(如果 alt-tab 菜单可见，Alt 键就是按住的)。
IfWinExist ahk_class #32771 ;指示 alt-tab 菜单当前是否出现在屏幕上。
    Send !{Escape}{Alt up}
return
```

目前，所有特殊的 `Alt-tab` 动作必须像上面的例子一样直接指定到一个热键(也就是它们不能像命令那样使用)。另外，`alt-tab` 菜单的存在能通过 `IfWinExist ahk_class #32771` 探测到。

自定义 `alt-tab` 动作也能通过热键来创建。在下面的例子中，你将按 `F1` 来显示菜单并原先在里面前移。然后你将按 `F2` 来激活选中的窗口(或按 `Escape` 来取消)：

```
*F1::Send {Alt down}{tab} ;在这里需要星号。译注：因为按了后 Alt 处于 Down 状态，再次激发热键只能靠星号匹配 Alt 这种修饰键。
!F2::Send {Alt up} ;松开 Alt 键激活选中的窗口。
~*Escape::
IfWinExist ahk_class #32771
    Send {Escape}{Alt up} ;取消菜单而不激活选中的窗口。
return
```

翻译：天堂之门 menk33@163.com 2008 年 11 月 22 日

20.1.1 #HotkeyInterval

随同 [#MaxHotkeysPerInterval](#)(See 20.1.5) 一起，指定 [热键](#)(See 4.) 激活的速率，当超过这一速率时，将会显示一个警告对话框。

`#HotkeyInterval Milliseconds`

参数

Milliseconds	间隔的长度，以毫秒为单位。
--------------	---------------

注意

如果在脚本里没有指定该指令，它会像设为 2000 毫秒一样运转。

更多细节和注意事项，请参考：[#MaxHotkeysPerInterval\(See 20.1.5\)](#)。

相关命令

[#MaxHotkeysPerInterval\(See 20.1.5\)](#)

示例

```
#HotkeyInterval 2000 ; 这是默认值(毫秒)。
```

```
#MaxHotkeysPerInterval 200
```

翻译：坛友 lwjiej 修正：天堂之门 menk33@163.com 2008 年 8 月 3 日

20.1.2 #HotkeyModifierTimeout

Affects the behavior of [hotkey\(See 4.\)](#) modifiers: CTRL, ALT, WIN, and SHIFT.

`#HotkeyModifierTimeout Milliseconds`

参数

Milliseconds	The length of the interval in milliseconds. The value can be -1 so that it never times out (modifier keys are always pushed back down after the Send), or 0 so that it always times out (modifier keys are never pushed back down).
--------------	---

注意

This directive **need not** be used when:

- Hotkeys send their keystrokes via the [SendInput\(See 20.12\)](#) or [SendPlay\(See 20.12\)](#) methods. This is because those methods postpone the user's physical pressing and releasing of keys until after the Send completes.
- The script has the keyboard hook installed (you can see if your script uses the hook via the "View->Key history" menu item in the main window, or via the [KeyHistory\(See 20.9\)](#) command). This is because the hook can keep track of which modifier keys (ALT/CTRL/WIN/SUPER) the user is physically holding down and doesn't need to use the timeout.

To illustrate the effect of this directive, consider this example:

```
^!a:::Send, abc
```

When the [Send](#)(See 20.12) command executes, the first thing it does is release the CTRL and ALT keys so that the characters get sent properly. After sending all the keys, the command doesn't know whether it can safely push back down CTRL and ALT (to match whether the user is still holding them down). But if less than the specified number of milliseconds have elapsed, it will assume that the user hasn't had a chance to release the keys yet and it will thus push them back down to match their physical state. Otherwise, the modifier keys will not be pushed back down and the user will have to release and press them again to get them to modify the same or another key.

The timeout should be set to a value less than the amount of time that the user typically holds down a hotkey's modifiers before releasing them. Otherwise, the modifiers may be restored to the down position (get stuck down) even when the user isn't physically holding them down.

You can reduce or eliminate the need for this directive with one of the following:

- Install the keyboard hook by adding the line [#InstallKeybdHook](#)(See 20.2) anywhere in the script (however, the hook is not supported on Win9x).
- Use the [SendInput](#)(See 20.12) or [SendPlay](#)(See 20.12) methods rather than the traditional [SendEvent](#)(See 20.12) method.
- When using the traditional [SendEvent](#)(See 20.12) method, reduce [SetKeyDelay](#)(See 20.14) to 0 or -1, which should help because it sends the keystrokes more quickly.

If this is directive is unspecified in a script, it behaves as though set to 50.

相关命令

[GetKeyState](#)(See 20.7)

示例

```
#HotkeyModifierTimeout 100
```

20.1.3 #Hotstring

改变 [热字符串](#)(See 5.) 的选项或结尾字符。

```
#Hotstring NoMouse
#Hotstring EndChars NewChars
#Hotstring NewOptions
```

参数

NoMouse	防止鼠标点击会像 这里 (See 5.) 描述的那样重设热字符串识别器。作为副作用，这也阻止了热字符串需要的 鼠标钩子 (See 20.3) (然而如果脚本因为其他目的需要它时，它仍会被安装，例如鼠标热键)。 #Hotstring NoMouse 出现在脚本的任何位置都将影响所有的热字符串，不仅仅是那些物理上在它下面的热字符串。
---------	--

EndChars NewChars	<p>指定 <code>EndChars</code> 单词，紧跟单独空格然后是新的结尾字符。例如：</p> <pre>#Hotstring EndChars -()[]{}';"/\,.?!`n `t</pre> <p>由于新的结尾字符对整个脚本是全局生效的 -- 不管 <code>EndChars</code> 指令出现在哪里 -- 故没有必要多次指定 <code>EndChars</code>。</p> <p>结尾字符的最大数量是 100 个。超出此长度的字符将被忽略。</p> <p>要让 <code>tab</code> 成为一个结尾字符，在列表里包括 <code>'t</code>。要让空格成为一个结尾字符，在列表的其他两个字符中间包含它（或在列表开头，如果列表仅包含一个其他字符或没有其他字符）。</p>
NewOptions	<p>指定新选项给 热字符串选项(See 5.) 里描述的选项。例如： <code>#Hotstring r s k0 c0</code></p> <p>与上面的 <code>EndChars</code> 不同，<code>#Hotstring</code> 指令按这种方法使用时是位置相关的。换句话说，整个热字符串部分可以有不同的默认选项，像在这个例子中一样：</p> <pre>::btw::by the way</pre> <pre>#Hotstring r c ; 下面所有的热字符串将使用 "send raw" 并且默认为区分大小写。</pre> <pre>::al::airline</pre> <pre>::CEO::Chief Executive Officer</pre> <pre>#Hotstring c0 ; 让这下面的所有热字符串不区分大小写。</pre>

相关命令

[Hotstrings\(See 5.\)](#)

翻译：天堂之门 menk33@163.com 2008 年 9 月 4 日

20.1.4 #IfWinActive/Exist

创建上下文相关的 [热键\(See 4.\)](#) 和 [热字符串\(See 5.\)](#)。这样的热键视某类窗口的激活或存在而执行一个不同的动作（或根本不执行）。

```
#IfWinActive [, WinTitle, WinText]
#IfWinExist [, WinTitle, WinText]
#IfWinNotActive [, WinTitle, WinText]
#IfWinNotExist [, WinTitle, WinText]
```

参数

WinTitle	目标窗口的标题或部分标题（匹配模式由设置在 自动执行部分(See 8.) 里的
----------	---

	SetTitleMatchMode (See 28.8) 决定(See 8.)。要使用一个窗口类, 指定 <code>ahk_class</code> 确切的 <code>ClassName</code> (通过 Window Spy 显示的)。要使用一个 窗口组(See 28.2.2), 指定 <code>ahk_group</code> <code>GroupName</code> 。也支持 <code>ahk_pid</code> 和 <code>ahk_id</code> (See 30.2) 关键词, 但对 <code>#IfWin</code> 来说更常见的是通过 <code>GroupAdd</code> (See 28.2.2) 间接地使用它们(或者, 使用 " <code>Hotkey IfWin</code> "(See 20.1.10))。最后, 通过指定 多个条件(See 30.2) 可以缩小搜索范围。例如: <i>My File.txt ahk_class Notepad</i>
WinText	如果有的话, 此参数必须是目标窗口的单独文本对象中的子字符串 (像内含的 Window Spy 工具显示的那样)。如果 <code>DetectHiddenText</code> (See 28.4) 在自动执行部分(脚本的顶部)已被开启, 隐藏的文本对象会被探测到。
ExcludeTitle ExcludeText	虽然这些不被支持, 但它们能间接地通过指定 <code>ahk_group MyGroup</code> 给 <code>WinTitle</code> 来使用 (在那通过 <code>GroupAdd</code> (See 28.2.2) 创建 <code>MyGroup</code> , 它支持 <code>ExcludeTitle/Text</code>)。

基本的操作

`#IfWin` 指令能简单地创建上下文相关的 `hotkeys`(See 4.) 和 `hotstrings`(See 5.)。例如:

```
#IfWinActive ahk_class Notepad
#space::MsgBox 你在记事本中按了 Win+空格 。
```

`#IfWin` 指令是位置相关的: 它们影响脚本中在它们下面的所有热键和热字符串。它们也是互斥的; 也就是说, 只有最近的一个才会生效。

要关闭上下文制约, 指定任意 `#IfWin` 指令但省略它所有的参数。例如:

```
#IfWinActive
```

当 `#IfWin` 关闭时 (或没用在脚本中), 所有的 `热键`(See 4.) 和 `热字符串`(See 5.) 对所有的窗口都有效 (除了被 `Suspend`(See 17.23) 或 `Hotkey 命令`(See 20.1.10) 禁用)。

当一个鼠标或键盘热键被 `#IfWin` 条件限制时, 它执行它本来的功能; 也就是说, 它传递到激活的窗口时就仿佛没有那么一个热键。有两个例外: 1) Windows 95/98/Me: 按一个 IfWin-禁用的 热键没有任何效果 (甚至没有它本来的功能); 2) 操纵杆热键: 虽然 `#IfWin` 可用, 但它不能阻止其他程序探测按钮的按动。

`#IfWin` 也能用来改变一个普通的按键像 `Enter` 或 空格 的表现。这在一个特殊的窗口忽略那个按键或执行一些你发觉不想要的动作时很有用。例如:

```
#IfWinActive Reminders ahk_class #32770 ; 在 Outlook 里的 "reminders" 。
Enter::Send !o ; 让 "Enter" 键击打开选择的 reminder 而不是让它睡觉。
#IfWinActive
```

变体 (重复的) 热键

一个特定的 `热键`(See 4.) 或 `热字符串`(See 5.) 能在脚本中定义多次, 假如每个定义都有不同的 `#IfWin` 条件。这些被称作 *hotkey variants*(热键变体)。例如:

```
#IfWinActive ahk_class Notepad
^!c::MsgBox 你在记事本中按了 Control+Alt+C 。
#IfWinActive ahk_class WordPadClass
^!c::MsgBox 你在写字板中按了 Control+Alt+C 。
#IfWinActive
^!c::MsgBox 你在不是记事本/写字板的窗口中按了 Control+Alt+C 。
```

如果有多个([译注：即条件不同但热键相同](#))变体适合被激发，那么只有最接近脚本顶部的那个会被激发。除此之外是全局的变体(没有 `#IfWin` 条件)：它总是最低的优先级；因此，它只在没有其他合适的变体的时候才会激发(这个例外不适用于 [热字符串\(See 5.\)](#))。

在创建重复的热键时，[修饰键符号\(See 4.\)](#) 例如 `^!+#` 的顺序没有关系。例如，`^!c` 和 `!^c` 是一样的。不过按键必须拼写一致。例如，为了这种目的，`Esc` 和 `Escape` 是不一样的(然而大小写没关系)。同样，任何带[通配符前缀 \(*\) \(See 4.\)](#)的热键完全和一个没有通配符的有区别；例如，`*F1` 和 `F1` 每个都将会有自己的变体。

要让同个热键的子程序通过多个变体执行，最简单的方法就是创建一堆相同的热键，每个热键都有不同的 `#IfWin` 指令在它上面。例如：

```
#IfWinActive ahk_class Notepad
#z::
#IfWinActive ahk_class WordPadClass
#z::
MsgBox 你在记事本或写字板中按了 Win+Z 。
return
```

或者，通过 `#IfWinActive ahk_group MyGroup` 使用一个[窗口组\(See 28.2.2\)](#)。

要动态地创建热键变种(当脚本在运行时)，请看 "[Hotkey IfWin](#)"([See 20.1.10](#))。

一般说明

`#IfWin` 也能在前缀按键被占用时还原它们本来的功能(在一个热键像 "a & b" 中[前缀按键\(See 4.\)](#)是 "a" 键)。这用在没有给一个特定的前缀指定可用的热键时。

当 `Gosub` 或 `Goto` 被用来跳转到一个热键或热字符串标识时，它跳转到最接近脚本顶部的那个变体。

当一个热键当前被 `#IfWin` 限制时，它的按键或鼠标按钮会用一个 "#" 符号出现在 `KeyHistory` 的([See 20.9](#)) "Type" 列里。这能帮助调试一个脚本。

当前不支持例如 `%Var%` 的变量引用。因此，百分号必须通过`%`[转义\(See 29.5\)](#)来允许将来支持它们。同样，原义的逗号必须被转义(通过 `,)` 来允许附加的变量在将来被添加。如果你需要绕弯解决这种限制，使用 [GroupAdd\(See 28.2.2\)](#) 和 [ahk_group\(See 30.2\)](#)。

[Hotkey 命令\(See 20.1.10\)](#) 指定热键的 `label` 不直接被 `#IfWin` 影响。取而代之的是，在最接近脚本底部使用的 `#IfWin` (如果有的话) 将影响所有由 `Hotkey` 命令创建的热键(除非使用 "[Hotkey IfWin](#)"([See 20.1.10](#)) 来改变)。

[Alt-tab 热键\(See 4.\)](#) 不受 `#IfWin` 影响：它们在所有的窗口有效。

最后找到的窗口(See 30.2) 可被 `#IfWinActive/Exist` 设置(但是不被 `#IfWinNotActive/NotExist` 设置)。例如：

```
#IfWinExist ahk_class Notepad  
#n::WinActivate ; 激活被 #IfWin 找到的窗口。
```

如果在 `#IfWin` 的某个变量里需要开头或结尾空格, 转义顺序(See 29.5) `s 和 `t 可以被使用。

由于性能的原因, `#IfWin` 不会持续监视指定窗口的激活或存在。取而代之的是, 它只在你输入一个热键或热字串时才去核查匹配窗口。如果匹配的窗口不存在, 你的键击或鼠标点击会被允许传递给没变的激活窗口(除了在 Windows 95/98/Me 上)。

Windows 95/98/Me: 如果热键的首个变体有一个 \$ 前缀, 所有的变体将被允许 "发送它们本身"。这为一个热键去执行它原来的功能提供了一种方法而不是完全什么也不做。例如：

```
$^a::Send ^a ; 首个变体必须有一个 $ 前缀来允许它在 Windows 9x 上 "发送自身"。  
  
#IfWinActive ahk_class Notepad  
  
^a::MsgBox 你在记事本激活时按了 Control-A 。
```

窗口标题和文本是区别大小写的。隐藏的窗口不被探测, 除非 `DetectHiddenWindows`(See 28.5) 在自动执行部分(脚本顶部)被开启。

相关命令

[Hotkey command](#)(See 20.1.10), [Hotkeys](#)(See 4.), [Hotstrings](#)(See 5.), [Suspend](#)(See 17.23), [IfWinActive](#)(See 28.6), [IfWinExist](#)(See 28.7), [SetTitleMatchMode](#)(See 28.8), [DetectHiddenWindows](#)(See 28.5)

示例

```
#IfWinActive ahk_class Notepad  
^!a::MsgBox 在记事本激活时你按了 Ctrl-Alt-A 。 ; 此热键如果在别的窗口按是无效的 (它将 "pass through"[经过] )。  
  
#c::MsgBox 在记事本激活时你按了 Win-C 。  
  
::btw::此 "btw" 的替换文本将只在记事本里出现。  
  
#IfWinActive  
  
#c::MsgBox 在一个不是记事本的窗口里你按了 Win-C 。
```

翻译：天堂之门 menk33@163.com 2008 年 8 月 27 日

20.1.5 #MaxHotkeysPerInterval

随同 `#`(See 20.1.1)[HotkeyInterval](#)(See 20.1.1) 一起, 指定 热键(See 4.) 激活的速率, 当超过这一速率时, 将会显示一个警告对话框。

#MaxHotkeysPerInterval Value

参数

Value	在 #(See 20.1.1)HotkeyInterval(See 20.1.1) 指定的时间里，不会触发一个警告对话框而能被按下的热键数量的上限。
-------	--

注意

注意不要将上述数值设定得过于宽松，因为如果你不经意地引入一个键击的无限循环(通过一个 [Send\(See 20.12\)](#) 命令意外地触发其它热键)的话，由于快速而泛滥的键盘事件，你的电脑可能会变得反应迟钝。

举一个非常简单的例子，热键 `^c::Send ^c` 将导致一个无限键击循环。要避免这种情况，为热键定义添加 [\\$ 前缀\(See 4.\)](#) (例如 `$^c::`)，使得热键不能被 `Send` 命令触发。

如果没有在脚本中指定该指令，它会像设成 `70` 一样来运转。

相关命令

[#HotkeyInterval\(See 20.1.1\)](#)

示例

```
#MaxHotkeysPerInterval 200
```

翻译：坛友 lwjiej 修正：天堂之门 menk33@163.com 2008年8月3日

20.1.6 #MaxThreads

设置同时启动的[线程\(See 30.19\)](#)的最大数量。

#MaxThreads Value

参数

Value	可同时存在的 线程(See 30.19) 的最大总数。指定大于 255 的数值等同与指定 255(在 1.0.48 以前的版本中，这个值的上限是 20)。
-------	---

注意

此设置是全局性的，这就意味着只需要将它指定一次（在脚本任何位置）就能影响整个脚本的表现。

虽然允许但是不推荐将值设为 1，因为这样会在每次脚本显示一个 [MsgBox\(See 19.11\)](#) 或者其他对话框时阻止新的[热键\(See 4.\)](#)运行。也会在每次另一个[线程\(See 30.19\)](#)休眠或等待时阻止[定时器\(See 17.21\)](#)运行。

如果一个线程子程序的首行是 [ExitApp\(See 17.7\)](#), [Pause\(See 17.18\)](#), [Edit\(See 22.9\)](#), [Reload\(See 20.1.13\)](#), [KeyHistory\(See 20.9\)](#), [ListLines\(See 22.11\)](#), [ListVars\(See 22.12\)](#) 或 [ListHotkeys\(See 20.1.11\)](#)，那么至多两种接下来的线程类型即使在 `#MaxThread` 达到的情况下也可以

被创建: [hotkey](#)(See 4.), [hotstring](#)(See 5.), [OnClipboardChange](#)(See 30.6), [GUI event](#)(See 19.3)。

还有, 不论有多少线程存在, [OnExit subroutine](#)(See 17.17)(子程序)总是可以启动。

如果此设置低于 [#MaxThreadsPerHotkey](#)(See 20.1.8), 那么它将有效地取代那个设置。

如果脚本中没有指定此指令, 那么它将表现得好像被设为 10 那样。

相关命令

[#MaxThreadsPerHotkey](#)(See 20.1.8), [Threads](#)(See 30.19), [#MaxHotkeysPerInterval](#)(See 20.1.5), [#HotkeyInterval](#)(See 20.1.1), [ListHotkeys](#)(See 20.1.11), [#MaxMem](#)(See 29.15)

示例

```
#MaxThreads 2
```

翻译: 天堂之门 menk33@163.com 2008 年 11 月 24 日

20.1.7 #MaxThreadsBuffer

Causes some or all [hotkeys](#)(See 4.) to buffer rather than ignore keypresses when their [#MaxThreadsPerHotkey](#)(See 20.1.8) limit has been reached.

`#MaxThreadsBuffer On|Off`

参数

On Off	<p>On: All hotkey subroutines between here and the next <code>#MaxThreadsBuffer ON</code> directive will buffer rather than ignore presses of their hotkeys whenever their subroutines are at their #MaxThreadsPerHotkey(See 20.1.8) limit.</p> <p>Off: This is the default behavior. A hotkey press will be ignored whenever that hotkey is already running its maximum number of threads (usually 1, but this can be changed with #MaxThreadsPerHotkey(See 20.1.8)).</p>
--------	--

注意

This directive is rarely used because this type of buffering, which is OFF by default, usually does more harm than good. For example, if you accidentally press a hotkey twice, having this setting ON would cause that hotkey's subroutine to automatically run a second time if its first [thread](#)(See 30.19) takes less than 1 second to finish (this type of buffer expires after 1 second, by design). Note that AutoHotkey buffers hotkeys in several other ways (such as "[Thread Interrupt](#)(See 22.22)" and "[Critical](#)"(See 22.7)). It's just that this particular way can be detrimental, thus it is OFF by default.

The main use for this directive is to increase the responsiveness of the keyboard's auto-repeat feature. For example, when you hold down a hotkey whose [#MaxThreadsPerHotkey](#)(See

20.1.8) setting is 1 (the default), incoming keypresses are ignored if that hotkey subroutine is already running. Thus, when the subroutine finishes, it must wait for the next auto-repeat keypress to come in, which might take 50ms or more due to being caught in between keystrokes of the auto-repeat cycle. This 50ms delay can be avoided by enabling this directive for any hotkey that needs the best possible response time while it is being auto-repeated.

As with all # directives, this one should not be positioned in the script as though it were a command (i.e. it is not necessary to have it contained within a subroutine). Instead, position it immediately before the first hotkey label you wish to have affected by it.

相关命令

[#MaxThreads](#)(See 20.1.6), [#MaxThreadsPerHotkey](#)(See 20.1.8), [\(See 30.19\)](#)[Critical](#)(See 22.7), [Thread \(command\)](#)(See 22.22), [Threads](#)(See 30.19), [Hotkey](#)(See 20.1.10), [#MaxHotkeysPerInterval](#)(See 20.1.5), [#HotkeyInterval](#)(See 20.1.1), [ListHotkeys](#)(See 20.1.11)

示例

```
#MaxThreadsBuffer on
#x::MsgBox, This hotkey will use this type of buffering.
#y::MsgBox, And this one too.
#MaxThreadsBuffer off
#z::MsgBox, But not this one.
```

20. 1. 8 #MaxThreadsPerHotkey

设置每个[热键](#)(See 4.)或[热字串](#)(See 5.)能同时启动的[线程](#)(See 30.19)的最大数量。

#MaxThreadsPerHotkey Value

参数

Value	一个给出的热键/热字串子程序能启动的 线程 (See 30.19)的最大数量 (限制 20)。
-------	---

注意

此设置被用来控制一个给出的[热键](#)(See 4.)或[热字串](#)(See 5.)子程序允许同时存在多少“实例”。例如，假设一个热键的最大限制是 1，在它的子程序已经运行的情况下，热键被再次按下，那么第二次按键将被忽略。这对防止意外的重复按键很有帮助。不过，如果你希望这些按键被缓冲而不是被忽略，也许是要增加键盘的自动重复特征的响应性，那么可以使用 [#MaxThreadsBuffer](#)(See 20.1.7)。

与 [#MaxThreads](#)(See 20.1.6) 不同，此设置不是全局性的。将它置于你希望影响的首个热键标签的前面，这样它后面所有的热键都将使用它的值，直到遇上此指令的另一个实例。

任何首行是 [ExitApp\(See 17.7\)](#), [Pause\(See 17.18\)](#), [Edit\(See 22.9\)](#), [Reload\(See 20.1.13\)](#), [KeyHistory\(See 20.9\)](#), [ListLines\(See 22.11\)](#), [ListVars\(See 22.12\)](#) 或 [ListHotkeys\(See 20.1.11\)](#) 的热键(See 4.)子程序将总是忽略此设置而运行。

如果 [#MaxThreads\(See 20.1.6\)](#) 的设置低于此设置, 那么会被优先采用。

如果脚本中没有指定此指令, 它将表现得好像被设为 1 那样。

相关命令

[#MaxThreads\(See 20.1.6\)](#), [#MaxThreadsBuffer\(See 20.1.7\)](#), [Critical\(See 22.7\)](#), [Threads\(See 30.19\)](#), [Hotkey\(See 20.1.10\)](#), [#MaxHotkeysPerInterval\(See 20.1.5\)](#), [#HotkeyInterval\(See 20.1.1\)](#), [ListHotkeys\(See 20.1.11\)](#)

示例

```
#MaxThreadsPerHotkey 3
```

翻译: 天堂之门 menk33@163.com 2008 年 11 月 24 日

20.1.9 #UseHook

强制使用钩子来实现部分或全部键盘热键(See 4.)。

[#UseHook \[On|Off\]](#)

参数

On Off	<p>#UseHook 后不带下列单词的话就等同于 #UseHook On。</p> <p>On: 将用键盘钩子(See 20.2)来实现此处和下个 #UseHook OFF (如果有的话)之间所有的键盘热键。</p> <p>Off: 将用默认的方法(如果可用 RegisterHotkey() 的话; 否则会用键盘钩子)实现热键。</p>
--------	--

注意

通常只要可能, 都会用 windows API 函数 [RegisterHotkey\(\)](#) 来实现键盘热键。不过在某些条件下, 如果用[键盘钩子\(See 20.2\)](#)来代替的话, 热键的响应性可能会更好。

将此指令调为 ON 就相当于在每个受影响热键的定义中使用 [\\$ 前缀\(See 4.\)](#)。Windows 95/98/Me 例外, 它们忽略 [#UseHook](#) (虽然 [\\$ 前缀\(See 4.\)](#)能起有限的作用)。

和所有在脚本启动时仅执行一次的 # 指令一样, [#UseHook](#) 在脚本中不应该像命令那样被放置(也就是说, 没必要将它包含在子程序中)。而是将它放在你想要影响的首个热键标签前。

使用[键盘钩子](#)(See 20.2)的热键不能被 [Send 命令](#)(See 20.12)触发。相似地，鼠标热键也不能被比如 [Click](#)(See 23.1) 命令触发，因为所有的鼠标热键都使用[鼠标钩子](#)(See 20.3)。要绕弯解决这种情况，可用 [Gosub](#)(See 17.8) 直接跳转到热键的子程序。例如：`Gosub #LButton`

如果脚本中没有出现此指令，那么将表现得像设成了 OFF。

相关命令

[#InstallKeybdHook](#)(See 20.2), [#InstallMouseHook](#)(See 20.3), [ListHotkeys](#)(See 20.1.11)

示例

```
#UseHook ;在此点后强制给热键使用钩子。  
#x::MsgBox, 此热键将用钩子实现。  
#y::MsgBox, 这个也一样。  
#UseHook off  
#z::MsgBox, 但这个没用。
```

翻译：天堂之门 menk33@163.com 2008 年 11 月 22 日

20.1.10 Hotkey

参数

KeyName	<p>热键的触发键的名称，包括任何 修饰键符号(modifier symbols)(See 4.)。例如，用 <code>#c</code> 来代替 <code>Win+C</code> 的热键。</p> <p>如果 <code>KeyName</code> 是一个现有的热键，那么该热键将被此命令的其它参数值修改。</p> <p><code>KeyName</code> 也可以是一个现有的热键标记名字(也就是一个双冒号标记)，这将导致原来的热键被此命令的其他参数值修改。</p> <p>操作一个 现有的热键时，<code>KeyName</code> 是大小写敏感的。不过，按键名称必须与现存热键的相同(比如，这种情况下 <code>Esc</code> 和 <code>Escape</code> 就是不同的)，另外，修饰键符号(See 4.) 的顺序是无关紧要的。</p> <p>当前的 IfWin 设置 决定了 <code>Hotkey</code> 命令要处理一个热键的哪个 变体，如果该变体尚不存在，将会被创建出来。</p> <p>一个热键第一次被创建出来的时候——不论是通过 <code>Hotkey</code> 命令还是脚本中的 双冒号标记(See 4.) ——它的键名和修饰键顺序就成为该热键的永久名称。就算此后 <code>Hotkey</code> 命令处理一个不同修饰键顺序的该热键，热键名称也不会发生改变。</p>
Label	热键按下的时候，将要被执行(作为一个新的 线程 (See 30.19))的那个标记名字。普通标

	<p>记和 热键(See 4.)/热字符串(See 5.) 标记都可以。不要包含后面的冒号。如果使用可变的 <i>Label</i> (比如 <code>%VarContainingLabelName%</code> 这样包含标签名的变量), 请事先调用 IsLabel(VarContainingLabelName)(See 10.) 以确保该标签的存在性。</p> <p>如果 <i>KeyName</i> 对应一个现存的热键, 本参数可以留空, 这种情况下标记不会发生变化。这在仅仅修改热键的 <i>Options</i> 的情况下很有用。</p> <p>如果该标记此前已经用此命令禁用了, 它将继续禁用下去。要启用它, 在 <i>Options</i> 中包含 ON 即可。</p> <p>本参数可以是下列特定值中的一个:</p> <p>On: 热键将被启用。如果该热键已经处于可用状态, 就不会有效果。</p> <p>Off: 热键将被禁用。如果该热键已经处于禁用状态, 就不会有效果。</p> <p>Toggle: 该热键的状态将被切换(启用或禁用)。</p> <p>AltTab (或其它): 这些是在 这里(See 4.) 提到的 Alt-Tab 特殊热键。</p> <p>备注: 当前的 IfWin 设置 决定了将要被 On/Off/Toggle 的热键是哪个 变体。</p>
Options	<p>由任意数量的下列内容构成的字符串, 可以用空格将其分割(或不分割)。例如: <code>UseErrorLevel B0</code>。</p> <p>UseErrorLevel: 如果此命令遇到一个问题, , 这个选项将会忽略警告对话框, 将 ErrorLevel(See 30.8) 设置为 下表 中的一个代码, 然后让 当前线程(See 30.19) 继续执行。</p> <p>On: 如果该热键当前被禁用, 此选项将启动它。</p> <p>Off: 如果该热键当前处于可用状态, 此选项将禁用它。这个选项通常用来以禁用状态创建一个热键。</p> <p>B 或者 B0: 指定 B 将会把该热键以 #MaxThreadsBuffer(See 20.1.7) 中描述的方式设置缓冲。指定 B0(B 后面跟一个零)禁用这种缓冲。</p> <p>Pn: 指定 P 跟随该热键的 线程优先级。如果创建热键的时候 P 选项被省略, 优先级将被设置为 0。</p> <p>Tn: 指定 T 后跟随 #MaxThreadsPerHotkey(See 20.1.8) 中描述的线程数, 例如: <code>T5</code></p> <p>如果 同时/部分 省略了 B 和 T 两个选项并且该热键已经存在, 这些设置值将不会改变。但如果该热键还没有被创建——就是说要用当前命令创建它的时候——它们将被指定为最近使用过的值。例如, 最靠近脚本结尾的 #MaxThreadsBuffer(See 20.1.7) 的值将会被采用。如果当前脚本中没有出现过 #MaxThreadsBuffer(See 20.1.7), 它的缺省值 (OFF) 将会被采用。对于 #IfWin(See 20.1.4) 的表现也是这样: 除非脚本启动之后 "Hotkey IfWin" 被执行过, 否则最靠近脚本结尾的那个结果将在创建热键的时候被采用。</p> <p>备注: 当前的 IfWin 设置 决定了将要被处理的热键是哪个 变体。如果该变体尚未被创建,</p>

	将在这时被创建。
IfWinActive IfWinExist	(IfWinNotActive 和 IfWinNotExist 都可以使用)这些子命令将会使此后创建的热键都是上下文相关的。详细信息见 下面 。
WinTitle WinText	<p>在这两个参数中, 用 <code>%var%</code> 这种方式引用的变量将在命令执行结束时被固定下来。换句话说, 此后的命令对该变量内容的修改不会影响现存的 IfWin 热键。</p> <p>类似于 #IfWinActive/Exist(See 20.1.4), WinTitle 和 WinText 使用自动执行部分中对 SetTitleMatchMode(See 28.8) 和 DetectHiddenWindows(See 28.5) 的缺省设置。详细内容请参阅 #IfWinActive/Exist(See 20.1.4)。</p>

ErrorLevel

只有在下列情况时, [ErrorLevel\(See 30.8\)](#) 才会被改变: 1)如果第一个参数是 IfWin[Not]Active/Exist, 这种情况下遇到问题的话置 1, 否则置 0; 2) UseErrorLevel 出现在 Options 参数中。

Error	描述
1	在 Label 中制定了一个不存在的标记的名字。
2	在 KeyName 中指定了当前的 键盘布局/语言 既不能识别也不能支持的按键。(译注: 这应该就是热键名/热字串名 中不能使用中文的原因。)
3	不被支持的前缀键。例如, 使用鼠标滚轮作为热键的前缀, WheelDown & Enter 这样的热键就不会被支持。
4	KeyName 不适合用在 AltTab or ShiftAltTab(See 4.) 功能上。这种情况需要两个键组合。例如: RControl & RShift::AltTab
5	该命令试图对不存在的热键进行修改。
6	该命令试图对一个热键的不存在的 变体 进行修改。要避免这个问题, 用 "IsLabel(VarContainingLabelName)(See 10.)" 确保操作目标的存在。
50	Windows 95/98/Me: 命令成功执行, 但操作系统拒绝激活该热键。导致这个问题的通常是该热键正在被其它脚本或者应用程序(或者是操作系统自身)使用。这个问题仅仅出现在 Windows 95/98/Me 中, 因为其它操作系统中, (AutoHotkey)程序会借助 键盘钩子(See 20.2) 来 取代 (See 20.2) 这个优先权。
51	Windows 95/98/Me: 命令成功执行, 但是 Windows 95/98/Me 不支持该热键。例如, 鼠标热键和 "a & b" 这种热键都不会被支持。
98	如果创建了这个热键将会突破 每脚本 1000 个热键 的限制(不过, 每个热键的 变体 数量不限, 热字串(See 5.) 的数量也没有限制)。
99	内存耗尽。这种情况极少发生, 通常仅仅在操作系统变得不稳定的时候才发生。

提示: UseErrorLevel 选项可以用来检验一个热键变体的存在性。例如:

```
Hotkey, ^!p,, UseErrorLevel
```

```
if ErrorLevel in 5,6
```

MsgBox 该热键不存在，或者不存在当前 **IfWin** 条件下的变体。

注意

如果仅仅为了根据当前处于激活状态的窗口来自动禁用某个热键，**Hotkey, ^!c, Off** 这种方式就不如 **#IfWinActive/Exist**(See 20.1.4)(或者下面提到的改良等价版本 "Hotkey IfWinActive/Exist")来得方便了。

通过**双冒号标记**(See 4.)创建的热键的执行效率高于通过 **Hotkey** 命令创建的热键，因为它们在脚本启动的时候会被批量地全部启用(而非一个个地启动)。所以，最好用本命令来创建那些在脚本运行后才能知道名称的热键。这样的情形之一是：脚本文件通过 **INI 文件**(See 16.29) 为热键配置多个不同的行为。

一个特定的标记可以被多个热键指向。如果知道某个标记是被其它热键呼叫的，可以查询内建变量 **A_ThisHotkey**(See 9.) 来判断它哪个热键。

如果脚本被 挂起 了，新 添加/启用 的热键也会被挂起，直到挂起状态被关闭(除非它们像 **Suspend**(See 17.23) 章节描述的那样被排除了)。

通过本命令做出的改变可能导致 **键盘**(See 20.2) 和/或 **鼠标**(See 20.3) 钩子被安装或者移除。

虽然 **Hotkey** 命令不能直接启用或禁用其它脚本中的热键，多数情况下却可以通过启用或创建同名热键来 **覆盖**(See 30.12) 它。覆盖成功与否取决于以下因素：1)将要覆盖的另一个脚本中的热键是否是一个 **钩子热键**(See 20.1.11)(非钩子热键，只要不是在 Win9x 上总是会被覆盖)；2)最近启动的脚本中的热键通常要优先于其它脚本中的热键(因此，如果要覆盖的脚本是最近启动的，覆盖总是会成功)；3)该热键被启用或者创建是否会重新激活 **键盘**(See 20.2) 或者 **鼠标**(See 20.3) 钩子(如果是，覆盖将总是成功)。

一旦某个脚本包含至少一个热键，它将持续运行，这意味着要使用 **ExitApp**(See 17.7) 而不是 **Exit** 来终结它。除非指定了 **#SingleInstance Off**，热键脚本也会自动具有 **#SingleInstance**(See 22.2) 属性。

关于 **Hotkey, IfWinXX [, WinTitle, WinText]** 要注意

"**Hotkey IfWin**" 命令将在脚本运行的时候允许上下文相关**热键**(See 4.)被创建和修改(与之不同的是，**#IfWinActive/Exist**(See 20.1.4) 指令是位置固定的并且在脚本启动同时起作用)。例如：

```
Hotkey, IfWinActive, ahk_class Notepad
```

```
Hotkey, ^!e, MyLabel ; 创建一个仅仅在记事本中工作的热键
```

使用 "**Hotkey IfWin**" 将把随后创建和修改的**热键**(See 4.)都变成上下文相关的。另外，每个 **IfWin** 子命令是互斥执行的；就是说，只有最近的一个才能产生作用。

要关闭上下文相关性(就是让随后创建的热键都能在所有窗口下运行)，指定一个忽略了 **WinTitle/Text** 这些参数的 **IfWin** 子命令即可。例如： **Hotkey ,IfWinActive**

如果一个脚本从来没用过 "**Hotkey IfWin**"，最近用过的 **#IfWin**(See 20.1.4) 指令(用过的话)将作用于 **Hotkey** 命令。

如果通过 `IfWin` 禁用一个鼠标或者键盘热键，该热键将执行它原本的功能；就是说，就算没有这样的热键，它也将发送给激活窗口。有两种情况例外：1) Windows 95/98/Me 中 `IfWin` 禁用的 热键不起作用(就连原本的功能也失效)；2)操纵杆热键，尽管 `IfWin` 有效，但它根本不会阻止其它程序检测它的按键。

变体（副本）热键 Variant (Duplicate) Hotkeys

只要具有不同的 `IfWin` 约束，特定的热键可以被多次创建。这被称作 **热键变体**。例如：

```
Hotkey, IfWinActive, ahk_class Notepad
Hotkey, ^!c, MyLabelForNotepad
Hotkey, IfWinActive, ahk_class WordPadClass
Hotkey, ^!c, MyLabelForWordPad
Hotkey, IfWinActive
Hotkey, ^!c, MyLabelForAllOtherWindows
```

如果不止一个变体符合热键的触发条件，全局变体属于例外(不受 `IfWin` 约束的那个)：它的优先级最低，就是说，其它变体都不会被触发的时候，它才触发。

创建副本热键时，像 `^!+#` 这些[修饰键符号](#)(See 4.)的顺序无关紧要。例如：`^!c` 和 `!^c` 是相同的。不过，按键的拼写方式必须始终如一。例如：`Esc` 和 `Escape` 在这种情况下就是不同的(尽管大小写没有关系)。使用了[通配符前缀\(*\)](#)(See 4.) 的热键跟没使用的完全不同；例如：`*F1` 和 `F1` 将会各自定义一套变体。

关于 `IfWin` 热键的详细内容，请参阅 [#IfWin's General Remarks](#)(See 20.1.4)。

相关命令

[Hotkey Symbols](#)(See 4.), [#IfWinActive/Exist](#)(See 20.1.4), [#MaxThreadsBuffer](#)(See 20.1.7), [#MaxThreadsPerHotkey](#)(See 20.1.8), [Suspend](#)(See 17.23), [IsLabel\(\)](#)(See 10.), [Threads](#)(See 30.19), [Thread](#)(See 22.22), [Critical](#)(See 22.7), [Gosub](#)(See 17.8), [Return](#)(See 17.19), [Menu](#)(See 22.13), [SetTimer](#)(See 17.21)

示例

```
Hotkey, ^!z, MyLabel
return

MyLabel:
MsgBox 按下了 %A_ThisHotkey%.
return
```

; 其它示例:

`Hotkey, RCtrl & RShift, AltTab ; 用 RCtrl & RShift 替换 Alt-Tab 的操作。`

`Hotkey, #c, On ; 重新启用热键 Win-C。`

`Hotkey, $+#c, Off ; 禁用热键 Shift-Win-C。`

`Hotkey, ^!a, , T5 ; 使该热键可以拥有 5 个线程。`

`Hotkey, IfWinActive, ahk_class Notepad`

`Hotkey, ^!c, MyLabelForNotepad ; 创建一个仅仅能用在记事本中的热键 Ctrl-Alt-C。`

20.1.11 ListHotkeys

显示当前脚本使用的热键，不论它们的子程序当前是否运行，也不论它们是否使用[键盘\(See 20.2\)](#)或者[鼠标\(See 20.3\)](#)钩子。

ListHotkeys

此命令相当于在主窗口选择 View->Hotkeys 菜单项。

如果一个热键被 [Hotkey\(See 20.1.10\)](#) 命令禁用，它在列表里将显示为 OFF 或者 PART ("PART" 意味着只有一些热键的[变体\(See 20.1.10\)](#)被禁用)。

如果某个热键子程序当前正在运行，那么会显示那个热键的线程总数。

最后，还会显示热键的类型，其为下列之一：

`reg`: 热键通过操作系统的 RegisterHotkey() 函数来执行。

`reg(no)`: 和上面一样，但是热键不可用(由于不被支持、被禁用或者被[挂起\(See 17.23\)](#))。

`k-hook`: 热键通过[键盘钩子\(See 20.2\)](#)来执行。

`m-hook`: 热键通过[鼠标钩子\(See 20.3\)](#)来执行。

`2-hooks`: 热键需要上面提到的两种钩子。

`joypoll`: 热键通过定期地轮询游戏操纵杆来执行。

相关命令

[#InstallKeybdHook\(See 20.2\)](#), [#InstallMouseHook\(See 20.3\)](#), [#UseHook\(See 20.1.9\)](#),
[KeyHistory\(See 20.9\)](#), [ListLines\(See 22.11\)](#), [ListVars\(See 22.12\)](#),
[#MaxThreadsPerHotkey\(See 20.1.8\)](#), [#MaxHotkeysPerInterval\(See 20.1.5\)](#)

示例

ListHotkeys

翻译: 天堂之门 menk33@163.com 2008 年 10 月 17 日

20.1.12 Pause

暂停脚本的 [current thread](#)(See 30.19)(当前线程)。

Pause [, On|Off|Toggle, OperateOnUnderlyingThread?]

参数

On Off Toggle	<p>如果留空或省略, 它默认为 Toggle。否则, 指定为下列一个单词:</p> <p>Toggle: 暂停 current thread(See 30.19) 除非它下面的线程已经暂停, 这种情况下将反暂停下面的线程。</p> <p>On: 暂停当前线程。</p> <p>Off: 如果当前线程下面的线程已经暂停, 当它恢复时将成为一种反暂停状态。反之, 命令无效。</p>
OperateOnUnderlyingThread?	<p>此参数被 "Pause Off" 忽略。对上面参数的其他两种而言, 它也被忽略除非暂停已经被开启(包括凭借 Toggle 开启的)。</p> <p>指定下面的一个数字:</p> <p>0 (或者省略): 命令暂停当前线程; 也就是, 正在运行暂定命令的线程。</p> <p>1: 命令标记当前线程下面的线程为暂定, 以便当它恢复时, 完成它运行的命令(如果有的话)并且之后进入一个暂停状态。如果当前线程下面没有线程, 脚本自己会暂停, 这将阻止 timers(See 17.21) 运行(当脚本没有线程时, 这个效果和使用了菜单项 "Pause Script" 一样)。</p>

注意

和 [Suspend](#)(See 17.23) 不同 -- 它禁用了 [hotkeys](#)(See 4.) 和 [hotstrings](#)(See 5.) -- pause 将冻结 [current thread](#)(See 30.19)。作为一个副作用, 任何当前线程下面被中断的线程也将潜伏着。

在任何线程被暂停时, [timers](#)(See 17.21) 也不会运行。相比之下, 明确地启动的线程例如 [hotkeys](#)(See 4.) 和 [menu items](#)(See 22.13) 仍能被运行; 但当它们的 [threads](#)(See 30.19) 结束时, 下面的线程仍将被暂停。换言之, 每个独立于其他的线程能被暂停。

当脚本的 [current thread](#)(See 30.19) 处于一个暂停状态时, 托盘图标的颜色从绿色转为红色。这个颜色的改变能够通过冻结图标来避免, 其通过为 Menu 命令的最后一个参数指定 1 来获得。例如:

[Menu](#)(See 22.13), Tray, Icon, C:\My Icon.ico, , 1

要禁用 [timers](#)(See 17.21) 而不暂停脚本, 使用 "[Thread, NoTimers](#)(See 22.22)"。

Pause 命令在功能上和内置菜单项 "Pause Script" 相似。

当一个脚本显示任何种类的 [menu](#)(See 22.13) (tray menu, menu bar, GUI context menu 等等) 时, 它总会停住(虽然不是正式地暂停)。

相关命令

[Suspend](#)(See 17.23), [Menu](#)(See 22.13), [ExitApp](#)(See 17.7), [Threads](#)(See 30.19),
[SetTimer](#)(See 17.21)

示例

```
Pause::Pause ; 给 "pause" 键指定暂停切换功能...
#p::Pause ; ... 或为 Win+p 或者其他一些热键指定此功能。
```

翻译: 天堂之门 menk33@163.com 2008 年 8 月 12 日

20.1.13 Reload

用一个新的脚本实例来替换当前运行的实例。

Reload

这个命令对于频繁改变的脚本很有用。给这个命令指定热键, 你可以在编辑器保存改动后轻松地重启脚本。

任何传递给原始脚本的 [command-line parameters](#)(See 8.)(命令行参数)不会被传递给新的实例。不要使用 **Reload** 来传递这样的参数。相应地, 使用 [Run](#)(See 24.5) 命令配合 [A_AhkPath](#)(See 9.) 和 [A_ScriptFullPath](#)(See 9.) 变量(以及 [A_IsCompiled](#)(See 9.) 变量, 如果脚本总是用在被编译过的形式)。同样, 包含字串 */restart* 作为第一个参数(也就是说, 在可执行程序名字的后面), 这告诉程序执行和 **Reload** 相同的动作。也可参见: [command line switches and syntax](#)(See 8.)(命令行开关和语法)。

当脚本重启时, 它在它的原来的工作目录(实际上是当它被首次载入时的那个目录)载入。也就是说, [SetWorkingDir](#)(See 16.33) 将不会改变被新的实例使用的工作目录。

如果脚本不能被重载 -- 也许是因为它有一个语法错误 -- 原来的脚本实例会继续运行。因此, **Reload** 命令后面应该跟着一旦有失败时你想要采取的任何措施(例如一个 [return](#)(See 17.19) 来退出当前子程序)。要想使原来的实例检测到失败, 仿照此例:

```
Reload

Sleep 1000 ; 如果成功, reload 将会在 Sleep 期间关闭这个实例, 因此下一行命令将从不执行。

MsgBox, 4,, 脚本不能被重载。你想打开它来编辑吗?

IfMsgBox, Yes, Edit

return
```

相关命令

[Edit](#)(See 22.9)

示例

```
^!r::Reload ;指定 Ctrl-Alt-R 作为重启脚本的热键。
```

翻译: Iwjiee 修正: 天堂之门 menk33@163.com 2008 年 8 月 1 日

20. 1. 14 Suspend

禁用或启用所有的或是选择的 [热键\(See 4.\)](#)。

Suspend [, Mode]

参数

Mode	On: 挂起所有的热键，除了那些在下面注意部分做出解释的外。 Off: 重新启用所有的热键。 Toggle (默认): 变更为它先前的相反状态 (On 或 Off)。 Permit: 除了把当前的子程序标记为免除挂起外，什么也不做。
------	---

注意

任何第一行有 **Suspend** (除了 "Suspend On") 的热键子程序将被免于挂起。换句话说，热键在 **Suspend** 为 ON 的状态下仍将可用。这就允许了通过这样的一个热键来关掉挂起。

要根据当前窗口的类型自动地禁用选择的热键或热字符串，请用 [#IfWinActive/Exist\(See 20.1.4\)](#)。

挂起一个脚本的热键不会终止脚本已在运行的 [线程\(See 30.19\)](#) (如果有的话)；请用 [Pause\(See 17.18\)](#) 命令来终止。

当脚本的热键被挂起时，它的托盘图标变为字母 S。可以通过冻结图标来避免，只要将 **Menu** 命令的最后一个参数指定为 1 即可。例如：

```
Menu(See 22.13), Tray, Icon, C:\My Icon.ico, , 1
```

如果脚本被挂起，内置变量 **A_IsSuspended** 将包含 1，否则为 0。

相关命令

[#IfWinActive/Exist\(See 20.1.4\)](#), [Pause\(See 17.18\)](#), [Menu\(See 22.13\)](#), [ExitApp\(See 17.7\)](#)

示例

```
^!s::Suspend ; 给一个热键指定挂起的开关功能。
```

翻译: 天堂之门 menk33@163.com 2008 年 9 月 8 日

20.2 #InstallKeybdHook

强制无条件地安装键盘钩子。

```
#InstallKeybdHook
```

注意

键盘钩子是为了激活不被 [RegisterHotkey](#)(操作系统内置的功能) 支持的 [热字串](#)(See 5.) 以及任何键盘 [热键](#)(See 4.) 这样的目的而监视键击的。它也支持一些其他特性例如 [Input](#)(See 20.11) 命令。

在 Windows 95/98/Me 下键盘热键不被支持，因为这些操作系统需要一个必须存在于 DLL 文件内的不同类型的钩子。

[AutoHotkey](#) 不会无条件地安装键盘和鼠标钩子因为它们总共消耗至少 500 KB 的内存。因此，键盘钩子通常仅在脚本包含下列条件之一时才会安装： 1) [热字串](#)(See 5.); 2) 一个或多个需要键盘钩子的 [热键](#)(See 4.) (大多数不需要); 3) [SetCaps/Scroll/Numlock AlwaysOn/AlwaysOff](#)(See 20.15); 4) [Input](#)(See 20.11) 命令，钩子在首次实际使用时安装。

相比之下，[#InstallKeybdHook](#) 指令将无条件地安装键盘钩子，它在允许 [KeyHistory](#)(See 20.9) 显示最近 20 次键击 (为了调试脚本目的) 或者避免需要使用 [#HotkeyModifierTimeout](#)(See 20.1.2) 时也许会很有用。

你能通过 [KeyHistory](#)(See 20.9) 命令或菜单项确定一个脚本是否在使用钩子。你能通过 [ListHotkeys](#)(See 20.1.11) 命令或菜单项确定哪个热键正在使用钩子。

这个指令也会使一个脚本 [persistent](#)(See 29.21)(持久运行)，意味着应该使用 [ExitApp](#)(See 17.7) 命令来终止脚本。

相关命令

[#InstallMouseHook](#)(See 20.3), [#UseHook](#)(See 20.1.9), [Hotkey](#)(See 20.1.10), [Input](#)(See 20.11), [#Persistent](#)(See 29.21), [KeyHistory](#)(See 20.9), [Hotstrings](#)(See 5.), [GetKeyState](#)(See 20.7), [KeyWait](#)(See 20.10)

示例

```
#InstallKeybdHook
```

翻译：天堂之门 menk33@163.com 2008 年 8 月 17 日

20.3 #InstallMouseHook

强制无条件地安装鼠标钩子。

```
#InstallMouseHook
```

注意

鼠标钩子是为了激活鼠标 [热键\(See 4.\)](#) 和 [帮助热字串\(See 5.\)](#) 为目的而监视鼠标点击的。在 Windows 95/98/Me 下它不被支持，因为这些操作系统需要一个必须存在于 DLL 文件内的不同类型的钩子。

AutoHotkey 不会无条件地安装键盘和鼠标钩子因为它们总共消耗至少 500 KB 的内存 (但如果键盘钩子已安装，那么安装鼠标钩子仅需要大约 50 KB 额外的内存；反之亦然)。因此，键盘钩子通常仅在脚本包含一个或多个鼠标 [热键\(See 4.\)](#) 时才会安装。它也会为了 [热字串\(See 5.\)](#) 而安装，但可以通过 [#Hotstring NoMouse\(See 20.1.3\)](#) 来禁用。

相比之下，[#InstallMouseHook](#) 指令会无条件地安装鼠标钩子，它在允许 [KeyHistory\(See 20.9\)](#) 来监视鼠标点击时可能会很有用。

你能通过 [KeyHistory\(See 20.9\)](#) 命令或菜单项确定一个脚本是否在使用钩子。你能通过 [ListHotkeys\(See 20.1.11\)](#) 命令或菜单项确定哪个热键正在使用钩子。

这个指令也会使一个脚本 [persistent\(See 29.21\)](#)(持久运行)，意味着应该使用 [ExitApp\(See 17.7\)](#) 命令来终止脚本。

相关命令

[#InstallKeybdHook\(See 20.2\)](#), [#UseHook\(See 20.1.9\)](#), [Hotkey\(See 20.1.10\)](#), [#Persistent\(See 29.21\)](#), [KeyHistory\(See 20.9\)](#), [GetKeyState\(See 20.7\)](#), [KeyWait\(See 20.10\)](#)

示例

```
#InstallMouseHook
```

翻译：天堂之门 menk33@163.com 2008 年 8 月 17 日

20.4 #KeyHistory

设置 [KeyHistory\(See 20.9\)](#) 窗口显示的键盘和鼠标事件的最大数量。你可以将其设为 0 来禁用 key history。

`#KeyHistory MaxEvents`

参数

MaxEvents	由 KeyHistory(See 20.9) 窗口显示的键盘和鼠标事件的最大数量。(默认 40，限定 500)。指定 0 完全禁用 key history。
-----------	--

注意

因为此设置在脚本开始运行前已经生效，所以只要指定一次 (在脚本的任意位置)。

由于每次键击或者鼠标单击由按下以及弹起事件组成，[KeyHistory\(See 20.9\)](#) 仅显示了此处指定的“完成事件”的一半。例如，如果脚本包含 “#KeyHistory 50”，将显示至多 25 次键击和鼠标鼠标点击。

相关命令

[KeyHistory](#)(See 20.9), [#NoTrayIcon](#)(See 22.1)(See 20.1.11)

示例

```
#KeyHistory 0      ; 禁用 key history。
#KeyHistory 100   ; 最多储存 100 个事件。
```

翻译：天堂之门 menk33@163.com 2008 年 10 月 7 日

20.5 BlockInput

开启或关闭用户通过鼠标和键盘与计算机交互的能力。

[BlockInput](#), [Mode](#)

参数

Mode	<p>Mode 1: 是下列的一个单词：</p> <p>On: 用户被阻止与计算机交互(鼠标和 键盘输入无效)。</p> <p>Off: 输入被重新启用。</p> <p>Mode 2 (在 Windows 9x 上无效): 此模式独立于其他两个模式进行操作。例如，<i>BlockInput On</i> 将继续阻止输入直到 <i>BlockInput Off</i> 被使用，即使下面的某个单词也在生效。</p> <p>Send: 用户的鼠标和键盘输入在 Send(See 20.12) 或 SendRaw(See 20.12) 命令正进行时将被忽略(仅对传统的 SendEvent mode(See 20.13)(发 送事件模式) 而言)。此参数阻止用户的键击去扰乱模拟的键击流程。当 <i>Send</i> 命令结束时，输入被重新启用(除非仍然被之前使用的一个 <i>BlockInput On</i> 阻止)。</p> <p>Mouse: 用户的鼠标和键盘输入在 Click(See 23.1)、MouseMove(See 23.6)、MouseClick(See 23.3) 或 MouseClickDrag(See 23.4) 命令正进行时被忽略(仅对传统的 SendEvent mode(See 20.13) 而言)。此参数阻止用户的鼠标移动和点击去扰乱模拟的鼠标事件。当鼠标命令结束时，输入被重新启用(除非仍然被之前使用的一个 <i>BlockInput On</i> 阻止)。</p> <p>SendAndMouse: 上面两种参数模式的组合。</p> <p>Default: 关掉 <i>Send</i> 和 <i>Mouse</i> 两种参数模式，但不改变当前阻止的输入状态。例如，如果 <i>BlockInput On</i> 当前生效中，使用 <i>BlockInput Default</i> 不会将它关掉。</p> <p>Mode 3 (在 Windows 9x 上无效；需要 v1.0.43.11+): 此模式独立于其他两个模式进行操作。例如，如果 <i>BlockInput On</i> 和 <i>BlockInput MouseMove</i> 同时生效中，鼠标移动将被一直阻止直到两个模式都关掉。</p> <p>MouseMove: 鼠标指针不会对用户的物理的鼠标移动做出反应而移动(DirectInput 程序可能是一个例外)。当脚本第一次使用此命令时，mouse hook(See 20.3)(鼠标钩子) 被</p>
------	--

装载(如果它还没被装载的话)。另外，脚本变成 [persistent](#)(See 29.21)(持久的)，意味着应该使用 [ExitApp](#)(See 17.7) 命令来终止它。鼠标钩子会继续处于装载状态直到下次使用了 [Suspend](#)(See 17.23) 或 [Hotkey](#)(See 20.1.10) 命令，在那时如果不被任何热键或热字串需要的话，它将被移除(见 [#Hotstring NoMouse](#)(See 20.1.3))。

MouseMoveOff: 允许用户移动鼠标指针。

注意

优先于 [BlockInput](#)，最好使用 [SendMode Input](#)(See 20.13) 或 [SendMode Play](#)(See 20.13) 以便键击和鼠标点击变得不可中断。这是因为不像 [BlockInput](#)，那些模式在发送期间不会抛弃用户输入的东西；而是将这些键击缓存起来并在之后发送。避免用 [BlockInput](#) 同样避开了像下面段落中描述的需要解决粘滞按键的情况。

如果在用户按住按键时 [BlockInput](#) 变为激活状态，可能导致这些按键"卡住"。这种情况可以通过在启用 [BlockInput](#) 前等待按键被松开来避免发生，像在这个例子中一样：

```
^!p::  
KeyWait Control ; 等待按键被松开。为每一个热键的修饰键使用一个 KeyWait 。  
KeyWait Alt  
BlockInput On  
; ... 发送键击和鼠标点击 ...  
BlockInput Off  
return
```

输入阻止即刻自动地关闭每当一个 ALT 事件发送时(之后重新启用)。

下面的表格显示了 [BlockInput](#) 的表现如何随着 Windows 的版本而改变；不过，由于一个 Windows API 的特性，在任何一个平台上按 [Ctrl+Alt+Del](#) 将重新启用输入。

操作系统	" BlockInput " 效果
Windows 95	无效。
Windows 98/Me	用户输入被阻止并且 AutoHotkey 不能模拟输入。
Windows NT 4 (without ServicePack 6)	无效。
Windows NT 4 (with ServicePack 6)	用户输入被阻止，不过 AutoHotkey 能模拟键击和鼠标点击。
Windows 2000/XP	用户输入被阻止，不过 AutoHotkey 能模拟键击和鼠标点击。

Windows 98/Me: 虽然在 [BlockInput](#) 期间脚本不能在那些操作系统上发送鼠标点击和键击，但像 [WinMove](#)(See 28.27) 这样的命令仍能起作用。[ControlSend](#) (See 20.6)也许也能工作。

某些类型的 [hook hotkeys](#)(See 20.1.9) 当 [BlockInput](#) 打开时仍能被触发。包括像 "MButton" (鼠标钩子) 和 "LWin & Space" (用明确的前缀而不是修饰符 "\$#" 的键盘钩子)这样的例子。

当脚本关闭时，输入被自动地重新启用。

相关命令

[SendMode](#)(See 20.13), [Send](#)(See 20.12), [Click](#)(See 23.1), [MouseMove](#)(See 23.6),
[MouseClick](#)(See 23.3), [MouseClickDrag](#)(See 23.4)

示例

```
if A_OSType <> WIN32_WINDOWS ; 也就是说不是 Windows 9x 。  

BlockInput, on  

Run, notepad  

WinWaitActive, 无标题 - 记事本  

Send, {F5} ; 粘贴时间和日期  

BlockInput, off
```

翻译：天堂之门 menk33@163.com 2008 年 8 月 6 日

20.6 ControlSend/ControlSendRaw

Sends simulated keystrokes to a window or control.

ControlSend [, *Control*, *Keys*, *WinTitle*, *WinText*, *ExcludeTitle*, *ExcludeText*]

ControlSendRaw: Same parameters as above.

参数

Control	<p>Can be either ClassNN (the classname and instance number of the control) or the name/text of the control, both of which can be determined via Window Spy. When using name/text, the matching behavior is determined by SetTitleMatchMode(See 28.8). If this parameter is blank or omitted, the target window's topmost control will be used. If this parameter is ahk_parent, the keystrokes will be sent directly to the control's parent window (see Automating Winamp(See 30.20) for an example).</p> <p>To operate upon a control's HWND (window handle), leave the <i>Control</i> parameter blank and specify <i>ahk_id %ControlHwnd%</i> for the <i>WinTitle</i> parameter (this also works on hidden controls even when DetectHiddenWindows(See 28.5) is Off) . The HWND of a control is typically retrieved via ControlGet Hwnd(See 28.1.4), MouseGetPos(See 23.5), or DII Call(See 18.3).</p>
Keys	<p>The sequence of keys to send (see the Send(See 20.12) command for details). To send a literal comma, escape(See 29.5) it (` ,). The rate at which characters are sent is determined by SetKeyDelay(See 20.14).</p>

	Unlike the Send (See 20.12) command, mouse clicks cannot be sent by ControlSend. Use ControlClick (See 23.2) for that.
WinTitle	The title or partial title of the target window (the matching behavior is determined by SetTitleMatchMode (See 28.8)). If this and the next 3 parameters are omitted, the Last Found Window (See 30.2) will be used. If this is the letter A and the next 3 parameters are omitted, the active window will be used. To use a window class, specify ahk_class ExactClassName (shown by Window Spy). To use a process identifier (PID) (See 24.4), specify ahk_pid %VarContainingPID%. To use a window group (See 28.2.2), specify ahk_group GroupName. To use a window's unique ID number (See 28.15), specify ahk_id %VarContainingID%. The search can be narrowed by specifying multiple criteria (See 30.2). For example: <i>My File.txt ahk_class Notepad</i>
WinText	If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if DetectHiddenText (See 28.4) is ON.
ExcludeTitle	Windows whose titles include this value will not be considered.
ExcludeText	Windows whose text include this value will not be considered.

ErrorLevel

[ErrorLevel](#)(See 30.8) is set to 1 if there was a problem or 0 otherwise.

注意

ControlSendRaw sends the keystrokes in the Keys parameter exactly as they appear rather than translating {Enter} to an ENTER keystroke, ^c to Control-C, etc.

If the *Control* parameter is omitted, this command will attempt to send directly to the target window by sending to its topmost control (which is often the correct one) or the window itself if there are no controls. This is useful if a window does not appear to have any controls at all, or just for the convenience of not having to worry about which control to send to.

By default, modifier keystrokes (Control, Alt, Shift, and Win) are sent as they normally would be by the Send command. This allows command prompt and other console windows to properly detect uppercase letters, control characters, etc. It may also improve reliability in other ways.

However, in some cases these modifier events may interfere with the active window, especially if the user is actively typing during a ControlSend or if the Alt key is being sent (since Alt activates the active window's menu bar). This can be avoided by explicitly sending modifier up and down events as in this example:

```
ControlSend, Edit1, {Alt down}f{Alt up}, Untitled - Notepad
```

The method above also allows the sending of modifier keystrokes (Control/Alt/Shift/Win) while the workstation is locked (protected by logon prompt).

[BlockInput](#)(See 20.5) should be avoided when using ControlSend against a console window such as command prompt. This is because it might prevent capitalization and modifier keys such as Control from working properly.

The value of [SetKeyDelay](#)(See 20.14) determines the speed at which keys are sent. If the target window does not receive the keystrokes reliably, try increasing the press duration via the second parameter of [SetKeyDelay](#)(See 20.14) as in these examples:

```
SetKeyDelay, 10, 10
```

```
SetKeyDelay, 0, 10
```

```
SetKeyDelay, -1, 0
```

If the target control is an Edit control (or something similar), the following are usually more reliable and faster than ControlSend:

[Control](#)(See 28.1.1), EditPaste, This text will be inserted at the caret position., ControlName, WinTitle

[ControlSetText](#)(See 28.1.10), ControlName, This text will entirely replace any current text., WinTitle

ControlSend is generally not capable of manipulating a window's menu bar. To work around this, use [WinMenuSelectItem](#)(See 28.1.14). If that is not possible due to the nature of the menu bar, you could try to discover the message that corresponds to the desired menu item by following the [SendMessage Tutorial](#)(See 30.16).

Window titles and text are case sensitive. Hidden windows are not detected unless [DetectHiddenWindows](#)(See 28.5) has been turned on.

相关命令

[SetKeyDelay](#)(See 20.14), [Escape sequences \(e.g. `%\)](#) (See 29.5), [Control](#)(See 28.1.1), [ControlGet](#)(See 28.1.4), [ControlGetText](#)(See 28.1.7), [ControlMove](#)(See 28.1.8), [ControlGetPos](#)(See 28.1.6), [ControlClick](#)(See 23.2), [ControlSetText](#)(See 28.1.10), [ControlFocus](#)(See 28.1.3), [Send](#), (See 20.12)[Automating Winamp](#)(See 30.20)

示例

```
ControlSend, Edit1, This is a line of text in the notepad window., Untitled
```

```
SetTitleMatchMode, 2
```

```
ControlSend, , abc, cmd.exe ; Send directly to a command prompt window.
```

20.7 GetKeyState

Checks if a keyboard key or mouse/joystick button is down or up. Also retrieves joystick status.

`GetKeyState, OutputVar, KeyName [, Mode]`

`KeyIsDown := GetKeyState(See 10.)("KeyName" [, "Mode"])`

参数

OutputVar	<p>The name of the variable in which to store the retrieved key state, which is either D for down or U for up (but the GetKeyState()(See 10.) function returns true (1) for down and false (0) for up). The variable will be empty (blank) if the state of the key could not be determined.</p> <p>The items below apply only to joysticks:</p> <ol style="list-style-type: none"> 1) For a joystick axis such as JoyX, <i>OutputVar</i> will be set to a floating point number between 0 and 100 to indicate the joystick's position as a percentage of that axis's range of motion. The format of the number can be changed via SetFormat(See 21.10). This test script(See 30.26) can be used to analyze your joystick(s). 2) When <i>KeyName</i> is JoyPOV, the retrieved value will be between 0 and 35900. The following approximate POV values are used by many joysticks: <ul style="list-style-type: none"> -1: no angle to report 0: forward POV 9000 (i.e. 90 degrees): right POV 27000 (i.e. 270 degrees): left POV 18000 (i.e. 180 degrees): backward POV
KeyName	<p>This can be just about any single character from the keyboard or one of the key names from the key list(See 7.), such as a mouse/joystick button (though mouse button state usually cannot be detected on Windows 95/98/Me). Examples: B, 5, LWin, RControl, Alt, Enter, Escape, LButton, MButton, Joy1.</p> <p>Alternatively, an explicit virtual key code such as vkFF may be specified. This is useful in the rare case where a key has no name. The virtual key code of such a key can be determined by following the steps at the bottom fo the key list page(See 7.).</p>
Mode	<p>This parameter is ignored when retrieving joystick status.</p> <p>If omitted, the mode will default to that which retrieves the logical state of the key. This is the state that the OS and the active window believe the key to be in, but is not necessarily the same as the physical state.</p> <p>Alternatively, one of these letters may be specified:</p> <p>P: Retrieve the physical state (i.e. whether the user is physically holding it down). Under Windows Me/98/95, the physical state (<i>Mode</i> = P) of a key is likely always the same as its logical state. Under Windows NT/2000/XP or later, the physical state of a key or mouse button will usually be the same as the logical state unless the keyboard and/or mouse hooks are installed, in which case it will</p>

accurately reflect whether or not the user is physically holding down the key or button (as long as it was pressed down while the script was running). You can determine if your script is using the hooks via the [KeyHistory](#)(See 20.9) command or menu item. You can force the hooks to be installed by adding the [#InstallKeybdHook](#)(See 20.2) and/or [#InstallMouseHook](#)(See 20.3) directives to the script.

T: Retrieve the toggle state (only valid for keys that can be toggled such as Capslock, Numlock, Scrolllock, and Insert). A retrieved value of D means the key is "on", while U means it's "off" (but the [GetKeyState\(\)](#)(See 10.) function returns true (1) for "on" and false (0) for "off").

Retrieving the toggle state of the *Insert* key might work only on Windows 2000/XP or later.

On Windows 9x, retrieving the toggle state might be less reliable. For example, immediately after the key is pressed, its new state might not be retrieved correctly until after the display of a window that's associated with the script, such as a [MsgBox](#)(See 19.11).

注意

To wait for a key or mouse/joystick button to achieve a new state, it is usually easier to use [KeyWait](#)(See 20.10) instead of a GetKeyState loop.

Systems with unusual keyboard drivers might be slow to update the state of their keys, especially the toggle-state of keys like Capslock. A script that checks the state of such a key immediately after it changed may use [Sleep](#)(See 17.22) beforehand to give the system time to update the key state.

For examples of using GetKeyState with a joystick, see the [joystick remapping page](#)(See 30.15) and the [Joystick-To-Mouse script](#)(See 30.25).

相关命令

[GetKeyState\(\)](#)(See 10.), [KeyWait](#)(See 20.10), [Key List](#)(See 7.), [Joystick remapping](#)(See 30.15), [KeyHistory](#)(See 20.9), [#InstallKeybdHook](#)(See 20.2), [#InstallMouseHook](#)(See 20.3)

示例

```
; Basic examples:

GetKeyState, state, RButton ; Right mouse button.

GetKeyState, state, Joy2 ; The second button of the first joystick.
```

```

GetKeyState, state, Shift

if state = D

    MsgBox At least one Shift key is down.

else

    MsgBox Neither Shift key is down.

GetKeyState, state, CapsLock, T ; D if CapsLock is ON or U otherwise.

state := GetKeyState("Capslock", "T") ; True if CapsLock is ON, false otherwise.

; Remapping example (this is only for illustration because it would be easier to use
; the built-in remapping feature(See 6.)):

; In the following hotkey, the mouse button is kept held down while NumpadAdd is
; down, which effectively transforms NumpadAdd into a mouse button. This method
; can also be used to repeat an action while the user is holding down a key or button:

*NumpadAdd:::

MouseClick, left,,, 1, 0, D ; Hold down the left mouse button.

Loop

{
    Sleep, 10

    GetKeyState, state, NumpadAdd, P

    if state = U ; The key has been released, so break out of the loop.

        break

    ; ... insert here any other actions you want repeated.

}

MouseClick, left,,, 1, 0, U ; Release the mouse button.

return

; Example: Make joystick button behavior depend on joystick axis position.

joy2::
```

```

GetKeyState, joyx, JoyX

if joyx > 75

    MsgBox Action #1 (button pressed while joystick was pushed to the right).

else if joyx < 25

    MsgBox Action #2 (button pressed while joystick was pushed to the left).

else

    MsgBox Action #3 (button pressed while joystick was centered horizontally).

return

; See the joystick remapping page(See 30.15) and the Joystick-To-Mouse script(See 30.25)
for other examples.

```

20.8 Key List (Keyboard, Mouse, Joystick)

LButton - 鼠标左键

RButton - 鼠标右键

MButton - 鼠标中键或滚轮

WheelDown - 这相当于将鼠标滚轮向后旋转(朝向你自己)

WheelUp - 上面这种的反向。

WheelLeft 和 WheelRight [v1.0.48+] - 这两个要求鼠标有左右滚动的能力，而且它们在 Windows Vista 以前的操作系统中没有效果。

(参见 [mouse wheel hotkeys\(See 4.\)](#) 来检测鼠标滚轮被滚动了多少。)

仅支持 Windows 2000/XP 或之后版本：

XButton1 - 仅出现在某些鼠标上的一个按键

XButton2 - 同上

注意：字母和数字按键的名称和单个字母或数字是一样的。例如：**b** 表示 "b" 键，**5** 表示 "5" 键。

Space - 空格键

Tab

Enter (或 Return)

Escape (或 Esc)

Backspace (或 BS)

Delete (或 Del)

Insert (或 Ins)

Home

End

PgUp

PgDn

Up

Down

Left

Right

ScrollLock

CapsLock

NumLock

Numlock 启用 Numlock 关闭

Numpad0 NumpadIns

Numpad1 NumpadEnd

Numpad2 NumpadDown

Numpad3 NumpadPgDn

Numpad4 NumpadLeft

Numpad5 NumpadClear

Numpad6 NumpadRight

Numpad7 NumpadHome

Numpad8 NumpadUp

Numpad9 NumpadPgUp

NumpadDot(.) NumpadDel

NumpadDiv (/) NumpadDiv (/)

NumpadMult (*) NumpadMult (*)

NumpadAdd (+) NumpadAdd (+)

NumpadSub (-) NumpadSub (-)

NumpadEnter NumpadEnter

F1 到 F24 - 在大多数键盘顶端的 12 个或更多的功能键。

AppsKey - 这是个调用右键上下文菜单的按键。 (译注：一般在右 Win 键和右 Ctrl 键中间)

LWin - 左边的 windows 标志 键

RWin - 右边的 windows 标志 键。注意：和 Control/Alt/Shift 不同，这里没有一般的/中性的 "Win" 键因为操作系统不支持。

Control (或 Ctrl)

Alt

Shift

注意: Shift::、Alt:: 和 Control:: 热键在按键松开时激发除非它们有波浪符前缀例如 ~Alt::。相比之下, 一个明确了左或右的热键例如 LAlt:: 在按下时激发。

注意: 多半, 下面的 6 个键不被 Windows 95/98/Me 支持。使用上面的代替:

LControl (或 LCtrl) - 左边的 control 键

RControl (或 RCtrl) - 右边的 control 键

LShift - 左边的 shift 键

RShift - 右边的 shift 键

LAlt - 左边的 Alt 键

RAlt - 注意: 如果你的键盘布局用 AltGr 替代了 RAlt, 你也许可以通过 <^>! 让它像[这里\(See 4.\)](#)描述的那样作为一个热键前缀来使用。此外, "LControl & RAlt::" 将让 AltGr 它自身成为一个热键。

PrintScreen

CtrlBreak

Pause

Break -- 由于此键和 Pause 是一起的, 在热键中不能使用 ^Pause 或 ^Break, 而用 ^CtrlBreak 来代替。

Help - 此键也许在大多数键盘上不存在。它通常和 F1 不一样。

Sleep - 注意 sleep 键在一些键盘上可能不是休眠功能。 (译注: 电源选项>高级 标签中可以调节。)

下面的仅存在那些有额外按键或按钮的多媒体或 Internet 键盘上:

Browser_Back

Browser_Forward

Browser_Refresh

Browser_Stop

Browser_Search

Browser_Favorites

Browser_Home

Volume_Mute

Volume_Down

Volume_Up

Media_Next

Media_Prev

Media_Stop

Media_Play_Pause

Launch_Mail

Launch_Media

Launch_App1

Launch_App2

SCnnn (这里 nnn 是一个按键的扫描代码) - 识别上面没有提到的特殊的按键。详见[特殊按键](#)。

VKn (这里 **nn** 是一个按键的十六进制虚拟按键代码) - 这种极少用的方法也阻止了某些类型的热键(See 4.)去依赖键盘钩子(See 20.2)。例如，后面的热键不使用键盘钩子，但作为一个副作用，通过按 **Home** 或 **NumpadHome** 都能触发它: ^VK24::MsgBox 你在按住 **Control** 的同时按下了 **Home** 或 **NumpadHome** 键。详见[特殊按键](#)。

Joy1 到 Joy32: 操纵杆的按钮。要帮忙确定你的操纵杆的按钮编号，请用这个[测试脚本](#)(See 30.26)。注意热键前缀符号(See 4.)例如 ^ (control) 和 + (shift) 不被支持 (虽然 [GetKeyState](#)(See 20.7) 能用来代替)。同样注意如果激活的窗口被设计成探测操纵杆按钮的按下，操纵杆的按钮按下将总会“传递”给它。

虽然下面的操纵杆控制器名称不能被用作热键，但它们能和 [GetKeyState](#)(See 20.7) 一起使用:

JoyX, JoyY 和 JoyZ: 操纵杆的 X (水平的)、Y (垂直的) 和 Z (高度/深度) 轴。

JoyR: 方向舵或操纵杆的第 4 个轴。

JoyU 和 JoyV: 操纵杆的第 5 个和第 6 个轴。

JoyPOV: point-of-view (hat) 控制器。

JoyName: 操纵杆的名称或它的驱动。

JoyButtons: 操纵杆支持的按钮总数(不会总是精确的)。

JoyAxes: 操纵杆支持的轴的总数。

JoyInfo: 提供一个包含零或下面多个字母组成的字串来显示操纵杆的功能: **Z** (有 Z 轴), **R** (有 R 轴), **U** (有 U 轴), **V** (有 V 轴), **P** (有 POV 控制器), **D** (POV 控制器有少数不连续的/不同的设置), **C** (POV 控制器是连续的/精细的)。字串示例: ZRUVPD

多个操纵杆: 如果计算机有多个操纵杆并且你想在第一个旁边使用另一个，那么在控制器名称前面包含操纵杆编号。例如，**2joy1** 是第二个操纵杆的第一个按钮。

注意: 如果你获得的脚本在识别你的操纵杆时有问题，有一个人报告说需要指定一个操纵杆的编号为除 1 以外的数字，即使只有单独一个操纵杆存在。不清楚这种情况如何发生的或者它是否正常，但将操纵杆编号在[操纵杆测试脚本](#)(See 30.26)中试验能帮助确定这是否适用于你的系统。

也可以看:

[操纵杆重映射](#)(See 30.15): 用一个操纵杆发送键击和鼠标点击的方法。

[操纵杆到鼠标脚本](#)(See 30.25): 将操纵杆作为鼠标使用。

响应手持遥控通过 [WinLIRC 客户端脚本](#)(See 30.36)发送的信号。

如果你的键盘或鼠标有按键没有在上面被列出来，你也许通过采用下面的步骤仍有可能使它成为一个热键(需要 Windows XP/2000/NT 或之后版本):

1. 确保至少有一个使用[键盘钩子](#)(See 20.2)的脚本正在运行。你只要通过打开一个脚本的主窗口并从菜单栏选择 "View->Key history" 就能知道它是否使用了键盘钩子。

2. 双击那个脚本的托盘图标来打开它的主窗口。
3. 按下你键盘上的某个“神秘按键”。
4. 选择菜单项 "View->Key history"
5. 向下滚动到页面的底部。在底部附近的某处就有你的按键的按下和弹起事件记录。注意：某些按键不生成事件记录，因此这里将看不到记录。如果是这种情况，你不能直接让那个特殊按键成为一个热键，因为你的键盘驱动或硬件在一个 AutoHotkey 访问不到的很低的层面处理它。要得到可能的解决方案，再往下看。
6. 如果你的按键可被探测到，记下在列表第二列的 3 位十六进制数值（例如 **159**）。
7. 要定义此键为一个热键，参照此例：

8.SC159:: ;用你那按键的数值替换 159。

9 MsgBox, %A_Hotkey% 被按下。

return

反向：要让其他某些键重映射为一个“神秘按键”，参照此例：

;用上面发现的数值替换 159。用按键的虚拟键值替换 FF (如果需要的话)，它能在按键历史页面的第一列找到。

#c::Send {vkFFsc159}(See 20.12)

替换的解决方案：如果你的按键或鼠标按键在按键历史页面没有被探测到，下面的某种方法也许有帮助：

1. 重新配置你的鼠标或键盘附带的软件（大多数时候能在控制面板或开始菜单进行访问）来使“神秘键”发送其他一些键击。然后这样的一个键击能在脚本里被定义为一个热键。例如，如果你设置一个神秘按键来发送 **Control+F1**，之后你能通过在脚本中用 **^F1::** 直接让它成为一个热键。
2. 试试 [DII Call: Support for Human Interface Devices](#)。你也可以在 [forum](#) (论坛) 中搜索类似 **RawInput*** 的关键字。
3. 下面是一个最后的手段并且通常应该仅在绝望时尝试。这是因为成功的机会很小并且它可能导致不想要的副作用而且难以撤销：
禁用或移除你的键盘或鼠标附带的额外的软件或者将它的驱动改变为一个更加标准的例如建立在操作系统上。这里假设的是有那么一种驱动给你特殊的键盘或鼠标，而你能不用它自定义的驱动和软件提供的特性而继续使用下去。

翻译：天堂之门 menk33@163.com 2008 年 12 月 31 日

20.9 KeyHistory

这个命令相当于选择主窗口的 "View->Key history" 菜单项。

要禁用按键记录，在脚本的任何地方指定下面这行：

#KeyHistory(See 20.4) 0

#KeyHistory(See 20.4) 也能用来改变被列出事件的最大数目。

此特性是为了帮助 [调试脚本和热键\(See 8.\)](#)。它也可以被用来检测非标准键盘按键的扫描码，按照在 [按键列表\(See 7.\)](#) 页面底部所描述的步骤(知道按键的扫描码就允许那个按键成为热键)。

滚轮事件 (WheelDown, WheelUp, WheelLeft, and WheelRight) 的虚拟按键 (VK) 是在 AutoHotkey 范围外不具有任何意义的占位符值。并且，滚轮事件的扫描码其实是滚轮转过的齿数(通常为 1)。

如果脚本没有安装 [keyboard hook \(键盘钩子\)\(See 20.2\)](#)，按键记录窗口将只列出脚本自身产生的键盘事件 (就是说没有用户的)。如果脚本没有安装 [mouse hook \(鼠标钩子\)\(See 20.3\)](#)，鼠标按钮事件将不被列出。你可以在脚本的主窗口(通过在托盘图标菜单上选择 "Open" 来访问)打开 "View->Key History" 找出你的脚本是否使用了其中的一个钩子。你可以通过添加下面两行中的一行或两行到脚本中来强制安装钩子：

```
#InstallKeybdHook(See 20.2)
#InstallMouseHook(See 20.3)
```

相关命令

[#KeyHistory\(See 20.4\)](#), [#InstallKeybdHook\(See 20.2\)](#), [#InstallMouseHook\(See 20.3\)](#),
[ListHotkeys\(See 20.1.11\)](#), [ListLines\(See 22.11\)](#), [ListVars\(See 22.12\)](#), [GetKeyState\(See 20.7\)](#),
[KeyWait\(See 20.10\)](#)

示例

KeyHistory ; 在一个窗口中显示记录。

翻译：惊幻 修正：天堂之门 menk33@163.com 2008 年 8 月 28 日

20.10 KeyWait

等待一个按键或鼠标/操纵杆按钮被松开或者按下。

KeyWait, KeyName [, Options]

参数

	这个名称可以是键盘上的几乎任意单独字符或者是 按键列表(See 7.) 里的某个按键名称，例如一个鼠标/操纵杆按钮。不支持除了操纵杆的按钮外的操纵杆属性。
KeyName	也可以指定一个显式的虚拟按键代码例如 vkFF。当一个按键没有名称并且在按下时生成不可见的字符这种罕见的情况时，这会很有用。它的虚拟按键代码可以通过参照 按键列表页面(See 7.) 底部的步骤来确定。
Options	如果此参数为空，命令将无限期地等待用户自己松开指定的按键或鼠标/操纵杆按钮。不过，如果没有安装 键盘钩子(See 20.2) 并且 KeyName 是通过例如 Send(See 20.12) 命令人为地松开的一个键盘按键，按键将被视为物理地松开了。当没有安装 鼠标钩子(See

20.3) 时，对鼠标按键来说也是一样的。

Options: 一个下列单个或多个字母组成的字符串(可任意顺序，在中间的空格是可选的):

D: 等待按键被按下。

L: 检测按键的逻辑状态，这是操作系统和激活的窗口相信按键会一直处于的状态(不一定和物理的状态一样)。**译注：**按键被检测时所处的状态会成为它永远的状态而不管之后是否物理地改变。此选项忽略操纵杆按钮。

T: 超时(例如, T3)。在判定超时并设置 [ErrorLevel\(See 30.8\)](#) 为 1 之前要等待的秒数。如果按键或按钮达到了指定的状态，命令不会等到超时期满。反而，它将马上设置 [ErrorLevel\(See 30.8\)](#) 为 0 并且脚本将继续执行。

超时的参数值可以是一个浮点数例如 2.5，但不能是一个十六进制的数值例如 0x03。

ErrorLevel

如果命令超时 [ErrorLevel\(See 30.8\)](#) 将设为 1，反之为 0。

注意

在 Windows Me/98/95 下，一个按键或鼠标按键的物理状态将一直和它的逻辑状态一致。

在 Windows NT/2k/XP 以及之后版本，一个按键或鼠标按键的物理状态通常将和逻辑状态一致，除非安装了键盘和/或鼠标钩子，这种情况下它会精确地反映逻辑状态而无论用户是否物理地按住了按键。通过 [KeyHistory\(See 20.9\)](#) 的命令或菜单项你能确定你的脚本是否在使用钩子。你能通过给脚本添加 [#InstallKeybdHook\(See 20.2\)](#) 和 [#InstallMouseHook\(See 20.3\)](#) 指令来强制安装任何一个或两个钩子。

当命令处于一个等待状态时，新的 [threads\(See 30.19\)](#)(线程)能通过 [hotkey\(See 4.\)](#), [自定义菜单项\(See 22.13\)](#) 或 [timer\(See 17.21\)](#) 来启动。

要等待两个或多个按键被松开，请连续地使用 [KeyWait](#)。例如：

KeyWait Control ; 等待 Control 和 Alt 两个都被松开。

KeyWait Alt

要等待一系列按键中的某个按键被按下，请看 [Input\(See 20.11\)](#) 命令的示例部分。

相关命令

[GetKeyState\(See 20.7\)](#), [Key List\(See 7.\)](#), [Input\(See 20.11\)](#), [KeyHistory\(See 20.9\)](#),
[#InstallKeybdHook\(See 20.2\)](#), [#InstallMouseHook\(See 20.3\)](#), [ClipWait\(See 15.1\)](#),
[WinWait\(See 28.32\)](#)

示例

; 例子 #1: 基本用法:

```
KeyWait, a ; 等待按键 A 被松开。  
KeyWait, LButton, D ; 等待鼠标左键被按下。  
KeyWait, Joy1, D T3 ; 等待首个操纵杆按钮在 3 秒内按下。  
KeyWait, LAlt, L ; 等待左边的 Alt 键被逻辑地松开。
```

; 例子 #2: 一个简单的热键:

```
~Capslock::  
KeyWait, Capslock ; 等待用户物理地松开它。  
MsgBox 你按了并且松开了 Capslock 键。  
return
```

; 例子 #3: 重映射一个按键或鼠标按钮 (这仅是为了阐述, 因为使用 内置重映射特性(See 6.) 会更加简单):

; 当 NumpadAdd 按下时鼠标左键是按住的, 这实际上把 NumpadAdd 变成了鼠标左键。

```
*NumpadAdd::  
MouseClick, left,,, 1, 0, D ; 按住鼠标左键。  
KeyWait, NumpadAdd ; 等待按键被松开。  
MouseClick, left,,, 1, 0, U ; 松开鼠标按键。  
return
```

; 例子 #4: 检测何时一个按键被两次按下 (近似于双击)。

; 当你按住 RControl 键要修饰另一个按键时, KeyWait 被用来阻止键盘的自动重复特性创造出一个不需要的两次按键。

; 它通过依赖于 #MaxThreadsPerHotkey 处于它的默认设置 1 来保持住热键的线程运行从而阻止自动重复。

; 注意: 在 SetTimer(See 17.21) 页面的底部有一个更复杂的脚本来辨别单次、两次和三次按键之间的区别。

~RControl::

```
if (A_PriorHotkey <> "~RControl" or A_TimeSincePriorHotkey > 400)
```

```
{
; 在按键之间间隔太久，所以这不是一个两次按键。

KeyWait, RControl

return

}

MsgBox 你两次按击了右边的 control 键。

return
```

翻译：天堂之门 menk33@163.com 2008 年 8 月 18 日

20.11 Input

等待用户输入一个字符串（不支持 Windows 9x，在 Windows 9x 下它什么都不处理）。

Input [, OutputVar, Options, EndKeys, MatchList]

参数

OutputVar	<p>用来保存用户输入文本的变量（默认情况下，手工的输入也被保存下来）。</p> <p>如果这个参数和其他参数被省略，那么任何在其他 thread(See 30.19) (线程)中执行的 Input 将被终止，终止后那个 Input 的 ErrorLevel(See 30.8) 会被赋值为 NewInput。而当前命令的 ErrorLevel(See 30.8) 会被设置为 0 (如果它终止了一个先前的输入) 或 1 (如果它在先前没有终止输入)。</p> <p>OutputVar 本身并不记录击键，而是记录由活动窗口的 键盘布局/键盘语言 击键产生的字符。所以，不产生字符的击键（比如 PageUp 和 Escape）并没有被记录下来。（尽管它们可以被下面的 EndKeys 参数识别出来）。</p> <p>空白字符会被逐个记录下来，比如 TAB (`t)。回车(ENTER)被记录为换行 (`n)。</p>
Options	<p><u>由下列字母组成的字符串(零个，一个或者多个，可任意顺序，在中间的空格是可选的):</u></p> <p>B: 忽略退格键。一般而言，按下退格键会删除最后一个输入的字符。注意：如果输入的文本是可见的（比如在一个编辑器中），用户在其中使用方向键或其他方式操纵了提示输入字符串的光标后，退格键依然会删除最后一个字符而不是光标前面的字符。</p> <p>C: 区分大小写。一般而言，MatchList 并不区分大小写（在 1.0.43.03 版本之前，只有字母 A-Z 被认为是有大小写的，而诸如 ü/Ü 的字母被认为没有）。</p> <p>I: 忽略所有由 AutoHotkey 脚本产生的字符，比如 SendEvent(See 20.12) 命令。然而，不论此选项是否开启，由 SendInput(See 20.12) 和 SendPlay(See 20.12) 方法产生的输入总是被忽略。</p> <p>L: 限制文本长度（比如 L5）。设置输入文本所允许的最大长度。如果文本到达了这个长度，</p>

	<p>输入会被终止, ErrorLevel(See 30.8) 会被赋值为 Max。但是如果文本和 <i>MatchList</i> 中某个词组符合的话, ErrorLevel(See 30.8) 会被赋值为 Match。如果这个选项没有开启, 那么文本默认的长度是 16383, 这也是文本的绝对最大长度。</p> <p>M: 如果修改过的从 Control-A 到 Control-Z 的击键对应于某个 ASCII 码的话, 那么它就可以被识别并被转录。试试下面这个识别 Control-C 的例子:</p> <pre>Transform, CtrlC, Chr, 3 ; Store the character for Ctrl-C in the CtrlC var. Input, OutputVar, L1 M if OutputVar = %CtrlC% MsgBox, You pressed Control-C. ExitApp</pre> <p>注意: 从 Control-A 到 Control-Z 的字符对应于 Chr(1)(See 10.) 到 Chr(26)(See 10.)。此 M 选项可能会导致一些快捷键在输入的过程中产生意料之外的效果(比如 Ctrl-LeftArrow)</p> <p>T: 设置超时 (比如 T3)。设置等待输入文本的秒数, 到时后终止 Input 并将 ErrorLevel(See 30.8) 赋值为 Timeout。如果 Input 超时, OutputVar 会被赋值为用户在超时以前输入的文本。这个参数可以被设为浮点数, 比如 2.5。</p> <p>V: 可见性。一般而言, 用户的输入是阻塞的(隐藏在操作系统之后)。使用这个选项来发送用户的击键至活动窗口。</p> <p>*: 通配符(寻找所有地方)。一般而言, 用户输入的文本必须完全匹配 <i>MatchList</i> 中的一个词组来产生一个匹配。使用这个选项来通过搜索完整长度的字符串来找到一个匹配。</p>
EndKeys	<p>按键的列表(零个, 一个或者多个), 其中任何一个按下都会终止 Input 的输入。(<i>EndKey</i> 本身不会被写入 OutputVar)。但一个 Input 通过这种方式被终止后, ErrorLevel(See 30.8) 会被赋值为 <i>EndKey</i> 接一个冒号 再接 <i>EndKey</i> 的值。例如:</p> <pre>EndKey::</pre> <p>EndKey:Escape</p> <p>这个 <i>EndKey</i> 列表使用一种和 Send(See 20.12) 命令相仿的格式。比如, 这个列表被设置为 {Enter}.{Esc} 后, 按下 ENTER, 句号 (.), 或者 ESCAPE 之后都会终止输入。要使用大括号来终止输入, 则需要将它们指定为 {}} 和/或 {}}。</p> <p>如果想要使用 Control, Alt, 或者 Shift 来作为输入的终止键, 则必须指出它是左边或者是右边的键, 比如设置为 {LControl}{RControl} 而不是 {Control}。</p> <p>尽管譬如 Control-C (^c) 之类用做修改的键不被支持, 一些需要 shift 键被按下的字符 -- 比如这些标点符号 ?!:@&{} -- 已经在 1.0.14+ 的版本中被支持了。</p> <p>明确的虚拟键也可以被指定为输入的终止键, 比如 {vkFF}。这在一些罕见的情况下会很有用, 比如一个键没有名字而且在按下的时候不产生可见的字符。这些虚拟键的键码可以通</p>

	过 key list page (See 7.) 页面下方的步骤来确定。
MatchList	<p>一个由逗号分隔的关键词组的列表，其中任何一个都可以终止一个输入（在这种情况下 ErrorLevel(See 30.8) 被赋值为 Match)。为了产生一个匹配，用户输入的文本必须完全匹配该列表中的某个词组（除非 * 选项 被选择）。另外，任何在分隔逗号旁边的空格或者跳格(tabs)都非常重要，这意味着它们也是匹配字符串中的一部分。例如，如果 <i>MatchList</i> 是 "ABC , XYZ "，那么用户在输入 ABC 后输入空格，或者在空格后输入 XYZ 都会产生一个匹配。</p> <p>两个连续的逗号在匹配词组中产生一个字面上的逗号。比如，下面的列表中，在第一个字符串的末尾会加上一个字面上的逗号： "string1,,,string2"。同样的，下面的列表中只包含一个内部有一个逗号的词组： "single,,item"。</p> <p>因为 <i>MatchList</i> 中的每一项并不是独立的参数，所以这个列表可以被全部包含在一个变量里。实际上，如果这个列表的长度超过了 16383，那么它的部分或者全部就必须存储在一个变量里，因为 16383 是任何一个脚本一行中所能存储的最大长度。例如，<i>MatchList</i> 可以由 %List1%,%List2%,%List3% 组成 -- 而其中每一个变量中保存了用来匹配字符串的词组的一个大的子列表。</p>

ErrorLevel

1 or 0	如果这个命令没有使用任何参数，那么 ErrorLevel (See 30.8) 会被赋值为 0 (如果它成功终止了一个先前的输入) 或 1 (如果没有输入需要处理)。
NewInput	这个命令被另外一个使用 Input 命令的 thread (See 30.19) (线程)所中断。
Max	输入达到了所允许的最大的字符串长度并且这个输入没有匹配到 <i>MatchList</i> 中的任何一项。
Timeout	输入超时。
Match	输入和 <i>MatchList</i> 中的某一项匹配。
EndKey:name	任意一个用来终止 Input 的 <i>EndKeys</i> 被按下。在这种情况下， ErrorLevel (See 30.8) 会被设置为 EndKey 接一个 冒号 再接 EndKey 的值(不带大括号)，比如： EndKey:Enter, EndKey:Escape, 等等。

注意

如果在使用这个命令的时候，另外一个 [thread](#)(See 30.19) (线程)中已经有一个 Input 在执行，那么那个线程中的 Input 会被终止且 [ErrorLevel](#)(See 30.8) 会被赋值为 NewInput。在这之后(如果这个命令给出了参数)，新的 Input 命令才开始执行。

在一个 Input 执行的过程中，诸如 [custom menu items](#)(See 22.13) 和 [timed subroutines](#)(See 17.21) 之类的新的 [threads](#)(See 30.19) (线程)依旧可以被创建。类似的，如果 Input 是 [visible](#) (可见的)，键盘的 [hotkeys](#)(See 4.) 也依然有效。如果 Input 不可见，那就只有 [hook hotkeys](#)(See 20.1.9) 可以被触发。

在一个脚本第一次使用这个命令时, [keyboard hook](#)(See 20.2) (键盘钩子)会被加载(如果它还没有加载)。另外, 这个脚本会变成 [persistent](#)(See 29.21) (持久的), 这意味着需要使用 [ExitApp](#)(See 17.7) 来停止这个脚本。键盘钩子会保持加载状态直到下一次 [Suspend](#)(See 17.23) 或 [Hotkey](#)(See 20.1.10) 命令的使用, 到那时它就会被移除, 除非其它的热键或热字符串仍然需要使用它。

如果您在使用多种语言或着多种键盘布局, 那么 [Input](#) 会使用当前活动窗口的键盘布局而不是脚本所使用的(无论 [Input](#) 是否 [visible](#) (可见))。但是, 在 1.0.44.03 之前的版本中使用的是脚本所使用的布局。

一般情况下, [hotstrings](#)(See 5.) 比 [Input](#) 命令更容易使用, 虽然它不是很灵活。

[Input](#) 在 Windows 9x 下什么都不做。(连 [ErrorLevel](#) 和 [OutputVar](#) 也不改变)。如果要在 Windows 9x 下检测输入, 请使用 [A_OSType](#)(See 9.)。

相关命令

[KeyWait](#)(See 20.10), [Hotstrings](#)(See 5.), [InputBox](#)(See 19.10), [#InstallKeybdHook](#)(See 20.2), [Threads](#)(See 30.19), [if var in/contains](#) [MatchList](#)(See 27.4)

示例

```
; Wait for the user to press any key. Keys that produce no visible character -- such as
; the modifier keys, function keys, and arrow keys -- are listed as end keys so that they
; will be detected too.

Input, SingleKey, L1,
{LControl}{RControl}{LAlt}{RAlt}{LShift}{RShift}{LWin}{RWin}{AppsKey}{F1}{F2}{F3}{F4}{F5}{F6}{F7}{F8}{F9}{F10}{F11}{F12}{Left}{Right}{Up}{Down}{Home}{End}{PgUp}{PgDn}{Del}{Ins}{BS}{Capslock}{Numlock}{PrintScreen}{Pause}
```

```
; This is a working hotkey example. Since the hotkey has the tilde (~)
; prefix, its own keystroke will pass through to the active window
; (except on Win9x). Thus, if you type [btw (or one of the other match
; phrases) in any editor, the script will automatically perform an
; action of your choice (such as replacing the typed text):
```

```
~[:::

Input, UserInput, V T5 L4 C, {enter}{esc}{tab}, btw,otoh,fl,ahk,ca
if ErrorLevel = Max
{
```

```
MsgBox, You entered "%UserInput%", which is the maximum length of text.  
return  
}  
  
if ErrorLevel = Timeout  
{  
    MsgBox, You entered "%UserInput%" at which time the input timed out.  
    return  
}  
  
if ErrorLevel = NewInput  
    return  
  
IfInString, ErrorLevel, EndKey:  
{  
    MsgBox, You entered "%UserInput%" and terminated the input with %ErrorLevel%.  
    return  
}  
  
; Otherwise, a match was found.  
  
SetKeyDelay, -1 ; Most editors can handle the fastest speed.  
  
if UserInput = btw  
    Send, {backspace 4}by the way  
else if UserInput = otoh  
    Send, {backspace 5}on the other hand  
else if UserInput = fl  
    Send, {backspace 3}Florida  
else if UserInput = ca  
    Send, {backspace 3}California  
else if UserInput = ahk  
    Run, www.autohotkey.com  
return
```

20.12

Send/SendRaw/SendInput/SendPlay/SendEvent

发送模拟的键击和鼠标点击到 [激活的\(See 28.12\)](#) 窗口。

[Send Keys](#)
[SendRaw Keys](#)
[SendInput Keys](#)
[SendPlay Keys](#)
[SendEvent Keys](#)

参数

Keys	一连串要发送的按键。和其他命令一起用时，在第一个参数前面的逗号是可选的。
------	--------------------------------------

Raw 模式: `SendRaw` 命令准确地按照键击本身显示的那样发送，而不是把 `{Enter}` 转换成一个 `ENTER` 键击，`^c` 转换成 `Control-C` 键击等等。要和 `SendInput`, `SendPlay` 或 `SendEvent` 命令一起使用 `raw` 模式，将 [{Raw}](#) 写在字串的首位；例如： `SendInput {Raw}abc`

普通模式: 当不用 `raw` 模式时，下面的字符被当作修饰键（这些修饰键仅影响紧跟着的下一个按键）：

!: 发送一个 `ALT` 键击。例如，`Send This is text!a` 将发送按键 "This is text" 并且之后按下 `ALT+a`。

注意: `!A` 在某些程序里会和 `!a` 产生不同的效果。这是因为 `!A` 按了 `ALT+SHIFT+A` 而 `!a` 按了 `ALT+a`。如果有疑虑，使用小写。

+: 发送一个 `SHIFT` 键击。例如，`Send +abC` 将发送文本 "AbC"，而 `Send !+a` 将按下 `ALT+SHIFT+a`。

^: 发送一个 `CONTROL` 键击。例如，`Send ^!a` 将按下 `CTRL+ALT+a`，而 `Send ^{Home}` 将发送 `CONTROL+HOME`。注意：`^A` 在某些程序里会和 `^a` 产生不同的效果。这是因为 `^A` 按了 `CONTROL+SHIFT+A` 而 `^a` 按了 `CONTROL+a`。如果有疑虑，使用小写。

#: 发送一个 `WIN` 键击，因此 `Send #e` 将按住 `Windows` 键并在之后按下字母 "e"。

SendInput 和 SendPlay [v1.0.43+]: `SendInput` 和 `SendPlay` 与 `Send` 命令使用相同的语法但是通常更快更可靠。此外，它们在发送期间缓存了任何物理的键盘或鼠标活动，这样就防止在发送时夹杂用户的键击。[SendMode\(See 20.13\)](#) 可以用来使 `Send` 命令和 `SendInput` 或 `SendPlay` 具有同样的功能。要知道每个模式更详细的内容，请看下面的 `SendInput` 和 `SendPlay`。

SendEvent [v1.0.43+]: `SendEvent` 和 1.0.43 版本之前的 `Send` 命令使用同样的方法发送键击。键击被发送时的频率由 [SetKeyDelay\(See 20.14\)](#) 来决定。默认，`Send` 和 `SendEvent` 是一样的；但它能通过 [SendMode\(See 20.13\)](#) 变得和 `SendInput` 或 `SendPlay` 一样。

Key 的名称: 下面的表格列出了能被发送的特殊按键（每个按键名称都必须用大括号围住）：

Key 的名称	生成的键击
<code>{F1}</code> - <code>{F24}</code>	功能键。例如： <code>{F12}</code> 是 <code>F12</code>

	键。
{!}	!
{#}	#
{+}	+
{^}	^
{}}	{
{}}	}
{Enter}	在主键盘上的 ENTER 键
{Escape} 或 {Esc}	ESCAPE 键
{Space}	SPACE 键(它仅需要在发送的字符串的开头或结尾出现 -- 在中间那些可能成为原义的 space)
{Tab}	TAB 键
{Backspace} 或 {BS}	Backspace 键
{Delete} 或 {Del}	Delete 键
{Insert} 或 {Ins}	Insert 键
{Up}	主键盘上的向上的箭头键
{Down}	主键盘上的向下的箭头键
{Left}	主键盘上的向左的箭头键
{Right}	主键盘上的向右的箭头键
{Home}	主键盘上的 Home 键
{End}	主键盘上的 End 键
{PgUp}	主键盘上的 Page-up 键
{PgDn}	主键盘上的 Page-down 键
{CapsLock}	CapsLock 键 (在 NT/2k/XP 上 使用 SetCapsLockState (See 20.15) 更可靠)
{ScrollLock}	ScrollLock 键 (同样请看： SetScrollLockState (See 20.15))
{NumLock}	NumLock 键 (同样请看：

	SetNumLockState (See 20.15))
{Control} 或 {Ctrl}	CONTROL 键 (技术信息：发送中性的而不是左边的虚拟按键扫描代码)
{LControl} 或 {LCtrl}	左 CONTROL 键 (技术信息：对 Win9x 来说和 CONTROL 一样，但在 NT/2k/XP 上它发送左边的虚拟按键而不是中性的按键)
{RControl} 或 {RCtrl}	右 CONTROL 键
{Control Down} 或 {Ctrl Down}	按住 CONTROL 键直到发送 {Ctrl Up}。XP/2000/NT：要按住左边或右边的键，请用 {RCtrl Down} 和 {RCtrl Up}。
{Alt}	ALT (技术信息：发送中性的而不是左边的虚拟按键扫描代码)
{LAlt}	左 ALT 键 (技术信息：对 Win9x 来说和 ALT 一样，但在 NT/2k/XP 上它发送左边的虚拟按键而不是中性的按键)
{RAlt}	右 ALT 键 (或 AltGr，取决于键盘布局)
{Alt Down}	按住 ALT 键直到发送 {Alt Up}。XP/2000/NT：要按住左边或右边的键，请用 {RAlt Down} 和 {RAlt Up}。
{Shift}	SHIFT 键 (技术信息：发送中性的而不是左边的虚拟按键扫描代码)
{LShift}	左 SHIFT 键 (技术信息：对 Win9x 来说和 SHIFT 一样，但在 NT/2k/XP 上它发送左边的虚拟按键而不是中性的按键)
{RShift}	右 SHIFT 键

{Shift Down}	按住 SHIFT 键直到发送 {Shift Up} 。XP/2000/NT: 要按住左边或右边的键, 请用 {RShift Down} 和 {RShift Up} 。
{LWin}	左 Windows 键
{RWin}	右 Windows 键
{LWin Down}	按住左 Windows 键直到发送 {LWin Up}
{RWin Down}	按住右 Windows 键直到发送 {RWin Up}
{AppsKey}	Windows App 键 (调用右键点击或上下文菜单)
{Sleep}	电脑 SLEEP 键。
{ASC nnnnn}	<p>发送一个 ALT+nnnnn 数字小键盘组合, 它能用来创建在键盘上不存在的特殊字符。要创建 ASCII 字符, 指定一个在 1 和 255 之间的数字。要创建 ANSI 字符 (多数规范语言), 指定一个在 128 和 255 之间的数字, 但在它前面用一个零开头, 例如 {Asc 0133} 。</p> <p>要创建 Unicode 字符, 指定一个在 256 和 65535 之间的数字(不用零开头)。不过, 它不是所有的应用程序都支持。因此, 为了更可靠和更简便地发送长的 Unicode 字符串, 请用 "Transform Unicode(See 21.11)" 。</p>
{vkXX} {scYYY} {vkXXscYYY}	发送一个有虚拟按键 XX 和扫描代码 YYY 的键击。例如: Send {vkFFsc159} 。如果省略 sc 或 vk 部分, 会就地发送

	最适当的值。
	XX 和 YYY 的值是十六进制的 并且能从主窗口的 View->Key history (See 20.9) 菜单项来确定。请看: 特 殊按键 (See 7.)
{Numpad0} - {Numpad9}	数字小键盘的数字键 (当 Numlock 显示是 ON 的状态 时)。例如: {Numpad5} 是数 字 5。
{NumpadDot}	数字小键盘的点 (当 Numlock 显示是 ON 的状态时)。
{NumpadEnter}	在数字小键盘上的 Enter 键
{NumpadMult}	在数字小键盘上的乘号键
{NumpadDiv}	在数字小键盘上的除号键
{NumpadAdd}	在数字小键盘上的加号键
{NumpadSub}	在数字小键盘上的减号键
{NumpadDel}	在数字小键盘上的 Delete 键 (此键和下面的数字小键盘按键 在 Numlock 是 OFF 状态时 使用)
{NumpadIns}	在数字小键盘上的 Insert 键
{NumpadClear}	在数字小键盘上的 Clear 键 (在 Numlock 是 OFF 状态时 通常是 '5' 键)。
{NumpadUp}	在数字小键盘上的向上的箭头 键
{NumpadDown}	在数字小键盘上的向下的箭头 键
{NumpadLeft}	在数字小键盘上的向左的箭头 键
{NumpadRight}	在数字小键盘上的向右的箭头 键
{NumpadHome}	在数字小键盘上的 Home 键

{NumpadEnd}	在数字小键盘上的 End 键
{NumpadPgUp}	在数字小键盘上的 Page-up 键
{NumpadPgDn}	在 数 字 小 键 盘 上 的 Page-down 键
{Browser_Back}	2000/XP/Vista+: 选择浏览器的 "后退" 按钮
{Browser_Forward}	2000/XP/Vista+: 选择浏览器的 "前进" 按钮
{Browser_Refresh}	2000/XP/Vista+: 选择浏览器的 "刷新" 按钮
{Browser_Stop}	2000/XP/Vista+: 选择浏览器的 "停止" 按钮
{Browser_Search}	2000/XP/Vista+: 选择浏览器的 "搜索" 按钮
{Browser_Favorites}	2000/XP/Vista+: 选择浏览器的 "收藏" 按钮
{Browser_Home}	2000/XP/Vista+: 启动浏览器并转到主页
{Volume_Mute}	2000/XP/Vista+: 总音量静音 / 取消静音。通常相当于 SoundSet(See 26.5) , +1, , mute
{Volume_Down}	2000/XP/Vista+: 降低总音量。通常相当于 SoundSet(See 26.5) -5
{Volume_Up}	2000/XP/Vista+: 增加总音量。通常相当于 SoundSet(See 26.5) +5
{Media_Next}	2000/XP/Vista+: 在 media player 选择下一首
{Media_Prev}	2000/XP/Vista+: 在 media player 选择前一首
{Media_Stop}	2000/XP/Vista+: 停止 media player
{Media_Play_Pause}	2000/XP/Vista+: 播放/暂停

	media player
{Launch_Mail}	2000/XP/Vista+: 启动 email 的应用程序
{Launch_Media}	2000/XP/Vista+: 启动 media player
{Launch_App1}	2000/XP/Vista+: 启动用户的程序 1
{Launch_App2}	2000/XP/Vista+: 启动用户的程序 2
{PrintScreen}	Print Screen 键
{CtrlBreak}	Ctrl+break
{Pause}	Pause 键
{Click [Options]} [v1.0.43+]	发送一个鼠标点击，使用在 Click 命令 (See 23.1) 中相同的可用选项。例如，{Click} 会在当前的鼠标指针位置点击鼠标左键一次，而 {Click 100, 200} 会在坐标 100, 200 处点击 (基于 CoordMode (See 22.6))。要移动鼠标而不是点击，在坐标后面指定 0；例如：{Click 100, 200, 0}。在鼠标点击之间的延迟由 SetMouseDelay (See 23.8) 命令决定（不是 SetKeyDelay (See 20.14)）。
{WheelDown}, {WheelUp}, {WheelLeft}, {WheelRight}, {LButton}, {RButton}, {MButton}, {XButton1}, {XButton2}	在指针当前的位置发送一个鼠标按键事件（要控制位置和其他选项，请用上面的 {Click} (See 23.1)）。在鼠标点击之间的延迟由 SetMouseDelay (See 23.8) 决定。WheelLeft/Right 需要 v1.0.48+，不过 Windows Vista 以上不需要。
{Blind}	当 {Blind} 在字符串的首位，如果 Alt/Control/Shift/Win

	<p>键在开始时是按下的位置，程序会避免松开它们。例如，热键 <code>+s::Send {Blind}abc</code> 会发送 ABC 而不是 abc 因为用户是按住了 Shift 键。</p> <p>{Blind} 也导致 SetStoreCapslockMode(See 20.16) 被忽略；也就是说，Capslock 键的状态不变。最后，{Blind} 忽略附加的控制键击被发送；这样的键击防止：1) 开始菜单出现在 LWin/RWin 键击期间； 2) 菜单栏在 Alt 键击期间被激活。</p> <p>Blind 模式当 重映射一个按键(See 6.) 时被用在其中。例如，重映射 <code>a:::b</code> 会产生：1) "b" 当你键入 "a" 时； 2) 大写 B 当你键入大写 A 时； 3) Control-B 当你键入 Control-A 时。</p> <p>{Blind} 不被 SendRaw 和 ControlSendRaw(See 20.6) 支持。此外，它不被 SendPlay 完全地支持，特别在处理修饰键(Control, Alt, Shift 和 Win)时。</p>
<p>{Raw}</p> <p>[v1.0.43+]</p>	<p>准确地按照键击本身显示的那样发送，而不是把 <code>{Enter}</code> 转换成一个 ENTER 键击，<code>^c</code> 转换成 Control-C 键击等等。不过字串 <code>{Raw}</code> 不需要出现在字符串的开头部分，一旦指定，它将持续影响字符串的剩余部分。</p>

要重复一个键击：将按键名称后跟要重复次数的数字括入大括号。例如：

`Send {DEL 4} ; 按 4 次 Delete 键。`

`Send {S 30} ; 发送大写 S 字符 30 次。`

```
Send +{TAB 4} ; 按 4 次 Shift-Tab 。
```

要按住或松开一个按键: 将按键名称后跟单词 **Down** 或 **Up** 括入大括号。例如:

```
Send {b down}{b up}
Send {TAB down}{TAB up}
Send {Up down} ; 按下向上的箭头键

Sleep 1000 ; 将它按住 1 秒。

Send {Up up} ; 松开向上的箭头键。
```

当一个按键通过上面的方法被按住，它不会开始自动重复像你物理地按住它那样（这是因为自动重复是一种驱动/硬件的特性）。不过，一个 [Loop\(See 17.12\)](#) 可被用来模拟自动重复。下面的例子发送了 20 次 tab 键击：

```
Loop 20
{
    Send {Tab down} ; 自动重复由连续的按下事件组成 (没用松开事件) 。

    Sleep 30 ; 在键击之间的毫秒数值 (或使用 SetKeyDelay(See 20.14)) 。

}
Send {Tab up} ; 松开按键。
```

单词 *DownTemp* 也可以被使用。它的效果和 *Down* 一样除了修饰键外 (Control/Shift/Alt/Win)。在那些情况下，*DownTemp* 告诉后面的发送，此键不是永远按下，并可能随时在一个键击调用它时松开。例如，*Send {Control DownTemp}* 之后跟上 *Send a* 会产生一个正常的 "a" 键击，不是一个 control-A 键击。

除了从字母 A 到 Z 外，下面的字母和符号也同样支持（不过，如果你的操作系统的代码页是某些 1252 [US & Western Europe] 之外的东西的话，这个列表可能会不一样）：

€? ,ƒ„…†‡^‰Š<Œ? Ž? ? „”•—~™š>œ? žŸ i¢£¤¥|§”©¤«¬®—°±23’μ¶·, 10»¼¼¾¼ž
ÀÁÃÃÃÃÆÇÈÉÈÈÌÌÏÐÑÒÓÔÔÖ×ØÙÛÛÝþßàáâãäåæçèéëìíïðñòóôöö÷øùûüýþþ

BlockInput 与 SendInput/SendPlay 相比: 虽然 [BlockInput\(See 20.5\)](#) 命令能用来防止用户输入的任何物理键击来扰乱模拟键击流程，但最好使用 [SendInput](#) 或 [SendPlay](#) 以便键击和鼠标点击变得不可中断。这是因为不像 BlockInput，SendInput/Play 不抛弃在发送期间用户输入的内容；取而代之的是，这样的键击被缓存起来并在之后发送。

当发送大量的键击时，一个 [continuation section\(See 8.\)](#)(连续章节) 能用来改善可读性和可维护性。

由于操作系统不允许模拟 CTRL-ALT-DELETE 组合，像 *Send ^!{Delete}* 将不产生效果。

SendInput 通常是更被喜欢用来发送键击和鼠标点击的方法因为它的优良的速度和可靠性。在大多数情况下, **SendInput** 近似于瞬间发生的, 即使是在发送长字符串时。由于 **SendInput** 如此快速, 它也更可靠, 因为将给其他一些窗口更多的机会来出乎意料地弹出并截断键击。可靠性更优是凭借用户在一个 **SendInput** 期间键入的任何东西将被延期到之后的事实来讲的。

和其他发送模式不同, 操作系统限制 **SendInput** 为大约 5000 字符 (这会随操作系统的版本和性能设置不同而改变)。字符和事件超出这个限制的将不发送。

注意: **SendInput** 忽略 **SetKeyDelay** 因为操作系统在这个模式不支持一个延迟。不过, 当 **SendInput** 在下面描述的条件下恢复为 **SendEvent** 时, 它可用 **SetKeyDelay -1, 0**(See 20.14) (除非 **SendEvent** 的按键延迟是 "-1,-1" , 这种情况使用的是 "-1,-1")。当 **SendInput** 恢复为 **SendPlay** , 它使用 **SendPlay** 的按键延迟。

如果除了那个执行 **SendInput** 的脚本, 一个脚本安装了一个 **low-level keyboard hook**(See 20.2)(底层键盘钩子), **SendInput** 自动恢复为 **SendEvent** (或者是 **SendPlay** , 如果 **SendMode InputThenPlay**(See 20.13) 是生效的话)。这被恢复是因为一个外部的钩子的存在禁用了所有 **SendInput** 的优势, 使它不如 **SendPlay** 和 **SendEvent** 两者。不过, 由于 **SendInput** 不能在程序里探测到一个底层钩子除了 AutoHotkey v1.0.43+ , 它将不会恢复到那些情况, 使它比 **SendPlay/Event** 的可靠性更低。

当 **SendInput** 通过例如 **{Click}** 的方式发送鼠标点击, 并且 **CoordMode Mouse, Relative**(See 23.8) 生效 (默认), 每次点击都会相对于在发送开始时激活的那个窗口。因此, 如果 **SendInput** 故意地激活了另一个窗口 (通过例如 **alt-tab** 的方式), 在同个命令里随后点击的坐标会是错误的, 因为它们仍然相对于那个之前的窗口而不是这个新的窗口。

Windows 95 (和 NT4 SP3 之前): **SendInput** 不被支持并且会自动地恢复为 **SendEvent** (或者是 **SendPlay** , 如果 **SendMode InputThenPlay**(See 20.13) 是生效的话)。

SendPlay 较其他模式而言的最大优势是它的 "play back" 键击能力和在多种游戏里的鼠标点击。例如, 某一特定的游戏可能仅在它们有 **SendPlay option**(See 5.) 时接受 **hotstrings**(See 5.) 。

在三个发送模式中, **SendPlay** 是最独特的, 因为它自身不模拟键击和鼠标点击。取而代之的是, 它创造一些列的事件 (消息) 直接地流向激活的窗口 (和 **ControlSend**(See 20.6) 相似, 但在一个更低的层面)。

和 **SendInput** 相似, **SendPlay** 的键击不会夹杂着用户输入的键击。因此, 如果用户碰巧在一个 **SendPlay** 期间键入某些东西, 那些键击会延期到之后。

虽然 **SendPlay** 比起 **SendInput** 来说是相当地慢, 但它通常比传统的 **SendEvent** 模式(即使当 **KeyDelay**(See 20.14) 是 -1 时) 更快。

SendPlay 不能触发调用两个 Windows 键(**LWin** 和 **RWin**) 的系统热键。例如, 它不能显示开始菜单或使用 **Win-R** 来显示运行对话框。

如果安装了 **键盘钩子**(See 20.2) , Windows 键(**LWin** 和 **RWin**) 在一个 **SendPlay** 期间会自动地被屏蔽。这防止在发送时用户意外地按到一个 Windows 键而显示开始菜单。相对而言, **LWin** 和 **RWin** 键之外的按键不需要被屏蔽, 因为操作系统自动地延迟它们直到 **SendPlay** 之后 (通过缓存)。

`SendPlay` 不使用 `SetKeyDelay` 和 `SetMouseDelay` 的标准设置。它默认根本不延迟，但可以像下面的例子展示的那样被改变：

`SetKeyDelay(See 20.14), 0, 10, Play ; 注意 0 和 -1 在 SendPlay 模式里是一样的。`

`SetMouseDelay(See 23.8), 10, Play`

`SendPlay` 不能开启或关闭 Capslock, Numlock 或 Scroll-lock 键。同样地，当 `GetKeyState(See 20.7)` 获得一个按键的状态时它不能去改变，除非键击是发送到脚本自身的一个窗口。即使这样，左/右修饰键(例如 RControl) 的任何改变能被探测到，仅通过它们的中性的相对键 (例如 Control)。同样，`SendPlay` 有其他一些在 [SendMode 页面\(See 20.13\)](#) 描述到的限制。

与 `SendInput` 和 `SendEvent` 不同，用户可以通过按 Control-Alt-Del 或 Control-Escape 来打断一个 `SendPlay`。当这发生时，剩余的键击将不被发送，但脚本会继续执行仿佛 `SendPlay` 已正常地完成了。

虽然 `SendPlay` 能发送 LWin 和 RWin 事件，但它们被直接地发送到激活的窗口而不是执行它们本来的操作系统功能。要绕弯解决它，使用 `SendEvent`。例如，`SendEvent #r` 将显示开始菜单的运行对话框。

和 `SendInput` 不同，`SendPlay` 即使在 Windows 95 和 NT4 SP3 之前也能使用。

[SendMode\(See 20.13\)](#), [SetKeyDelay\(See 20.14\)](#), [SetStoreCapslockMode\(See 20.16\)](#), [Escape sequences \(e.g. `%\) \(See 29.5\)](#), [ControlSend\(See 20.6\)](#), [BlockInput\(See 20.5\)](#), [Hotstrings\(See 5.\)](#), [WinActivate\(See 28.12\)](#)

`Send Sincerely,{enter}John Smith ; 键入一个两行的签名。`

`Send !fs ; 选择 File->Save 菜单项 (Alt+F 后跟 S)。`

`Send {End}+{Left 4} ; 跳至文本末尾然后发送四个 shift+左-箭头 键击。`

`SendInput {Raw} 一系列未加工的字符通过最快的模式(SendInput\(See 20.12\)) 发送。`

翻译：天堂之门 menk33@163.com 2008 年 8 月 21 日

20.13 SendMode

让 `Send(See 20.12)` 命令和 `SendInput` 或 `SendPlay` 具有同样的功能而不是其默认的 (`SendEvent`)。也让 Click 和 MouseMove/Click/Drag 使用了指定的方法。

`SendMode Input|Play|Event|InputThenPlay`

第一个参数是下面的某个单词：

Event: 这是所有脚本开始时默认使用的。它让 [Send](#)(See 20.12), [SendRaw](#)(See 20.12), [Click](#)(See 23.1) 和 [MouseMove](#)(See 23.6)/[Click](#)(See 23.3)/[Drag](#)(See 23.4) 使用 [SendEvent](#)(See 20.12) 方法。

Input: 让 [Send](#)(See 20.12), [SendRaw](#)(See 20.12), [Click](#)(See 23.1) 和 [MouseMove](#)(See 23.6)/[Click](#)(See 23.3)/[Drag](#)(See 23.4) 切换到 [SendInput](#)(See 20.12) 方法。已知限制：

- Windows 资源管理器忽略 [SendInput](#) 某些导航的热键例如 Alt+左箭头。要绕弯解决这问题，使用 [SendEvent !{Left}](#) 或 [SendInput {Backspace}](#)。

InputThenPlay [v1.0.43.02+]: 和上面的一样，除了当 [SendInput 不可用](#)(See 20.12) 时恢复为 Play 模式（下面的）而不是退回到 Event 模式。这也会促使 [SendInput 命令](#)(See 20.12) 它自身在 [SendInput 不可用](#) 时恢复到 Play 模式。

Play: 让 [Send](#)(See 20.12), [SendRaw](#)(See 20.12), [Click](#)(See 23.1) 和 [MouseMove](#)(See 23.6)/[Click](#)(See 23.3)/[Drag](#)(See 23.4) 切换到 [SendPlay](#)(See 20.12) 方法。(See 23.4) 已知限制：

- 不存在当前键盘布局上的字符（例如英语里的 Ô）不能被发送。要绕弯解决这问题，使用 [SendEvent](#)(See 20.12)。
- 例如在那些写字板和 Metapad 的 RichEdit(富文本编辑框) 控件（也可能其他控件）里模拟鼠标拖选可能无效。要为一个特别的拖选使用一个替换的模式，参照这个例子：[SendEvent](#)(See 20.12) {Click 6, 52, down}{Click 45, 52, up}
- 模拟鼠标滚轮的旋转仅产生在一个方向的移动（通常是向下的，但在某些应用程序里是向上的）。而且在例如微软 Word 和记事本应用程序中滚轮旋转可能无效。要为一个特别的旋转使用一个替换的模式，参照这个例子：[SendEvent](#)(See 20.12) {WheelDown 5}
- 当在自动执行部分（脚本顶部）使用 [SendMode Play](#)，将影响所有重映射的按键并可能失去它们的某些功能。详见 [SendPlay 重映射限制](#)(See 6.)。

注意

由于 [SendMode](#) 也改变了 [Click](#)(See 23.1) 和 [MouseMove](#)(See 23.6)/[Click](#)(See 23.3)/[Drag](#)(See 23.4) 的模式，当你想要为一个特别的鼠标事件使用一个不同的模式时可能很多次了。最简单的方式就是通过 [{Click}](#)(See 20.12) 来实施。例如：

`SendEvent {Click 100, 200} ; SendEvent 使用更旧更传统的方法点击。`

如果 [SendMode](#) 用在了自动执行部分（脚本顶部），它也会影响 [键盘和鼠标重映射](#)(See 6.)。尤其是如果你在重映射时使用了 [SendMode Play](#)，请看 [SendPlay 重映射限制](#)(See 6.)。

每个再次启动的 [thread](#)(See 30.19)（例如一个 [hotkey](#)(See 4.), [自定义菜单项](#)(See 22.13) 或 [定时的](#)(See 17.21) 子程序）都以此命令的默认设置重新开始。在自动执行部分（脚本顶部）使用此命令可以改变那个默认设置。

相关命令

[Send](#)(See 20.12), [SetKeyDelay](#)(See 20.14), [SetMouseDelay](#)(See 23.8), [Click](#)(See 23.1), [MouseClick](#)(See 23.3), [MouseClickDrag](#)(See 23.4), [MouseMove](#)(See 23.6)

示例

```
SendMode Input
SendMode InputThenPlay
```

翻译：天堂之门 menk33@163.com 2008 年 8 月 21 日

20.14 SetKeyDelay

设置 [Send](#)(See 20.12) 和 [ControlSend](#)(See 20.6) 命令发送的两次按键事件间的延时。

`SetKeyDelay [, Delay, PressDuration, Play]`

参数

Delay	延时时间，单位毫秒。可以是一个 expression/表达式 (See 9.)。使用 -1 表示无延时，使用 0 表示最小延时（但是，如果使用了 <i>Play</i> 参数的话，0 和 -1 都表示无延时）。留空则保持当前的 <i>延时</i> 不变。
PressDuration	<p>按键时长。有些游戏或者特殊的应用程序要求每次按键有一定的按下时间。也就是说，按下一个按键，停顿一段时间，再松开。</p> <p>使用 -1 表示无停顿，使用 0 表示最小停顿（但是，如果使用了 <i>Play</i> 参数的话，0 和 -1 都表示无停顿）。留空则保持当前的 <i>按键时长</i> 不变</p> <p>注意：<i>PressDuration</i> 参数同样会在功能键（CTRL, ALT, SHIFT, 以及 WIN）状态改变的时候产生一个停顿。</p> <p>这个参数可以是一个 expression/表达式(See 9.)。</p>
Play [v1.0.43+]	这个参数使用单词 <i>Play</i> 的话，表示这是为 SendPlay 模式 (See 20.12) 设置的延时，而不是一般的 SendEvent 模式。如果脚本中从未使用过这个参数，则 SendPlay 模式的延时永远是 -1/-1。

注意

注意：`SetKeyDelay` 的设置对 [SendInput](#)(See 20.12) 是无效的，在那个模式中永远是无延时。同样，对启用了 [SendMode Input](#)(See 20.13) 设置的 [Send](#)(See 20.12) 命令也是无效的。

脚本在执行了每个 [Send](#)(See 20.12) 或 [ControlSend](#)(See 20.6) 命令发送的按键事件之后，会有一个自动的延时（休眠）。这样做的目的是提高脚本的可靠性，因为一个窗口一般无法响应太过频繁的按键事件。

根据系统的时间精度，设置的延时可能会被四舍五入为最近的整十数字。例如在 XP 中，1 到 10 之间的延时（包括 10）都等效于 10（NT 和 2000 中也是这样）。

在 [Send/SendEvent](#) 模式中，设置延时为 0 的话相当于执行了命令 `Sleep(0)`，它会将当前脚本的剩余时间片分配给有需要的进程。如果没有进程需要，延时 0 就相当于完全没有延时。相反的，设置延时为 -1 表示永远没有延时。为了提高脚本的可靠性，一般推荐使用 0 而不是 -1。

当延时设置为 `-1` 的时候，在 [SendEvent 模式](#)(See 20.13) 中，脚本发送按键事件的速度取决于脚本进程的优先级。提升一个脚本的优先级，使用 [Process\(See 24.4\)](#), `Priority,, High`。虽然这样一般会使按键的发送速度超过 [当前窗口\(See 28.12\)](#) 的处理速度，不过系统会自动将按键存入缓存中。在 [Send\(See 20.12\)](#) 命令完成之后，缓存中的按键会继续发送到目标窗口(即使这个窗口不再处于激活状态)。这种情况一般是不要紧的，因为任何并发的对同一个窗口发送的按键事件会自动排列在缓存中的按键事件之后。

如果未进行设置，在 [SendEvent](#) 模式中默认的 `Delay` 是 `10`(在 `.aut` 脚本中是 `20`)。在 [SendPlay 模式](#)(See 20.12) 中默认的 `Delay` 是 `-1`。两个模式中默认的 `PressDuration` 都是 `-1`。

内置变量 [`A_KeyDelay`](#) 保存了当前 [Send/SendEvent](#) 模式中 `Delay` 参数的设置。没有内置变量保存 [SendPlay 模式](#)(See 20.12) 中 `Delay` 参数的设置，也没有内置变量保存 `PressDuration` 参数的设置。

每一个新运行的 [Thread/线程\(See 30.19\)](#) (例如一个 [hotkey/热键\(See 4.\)](#), [custom menu item/自定义菜单\(See 22.13\)](#), 或 [timed/定时器\(See 17.21\)](#) 事件) 会将该命令的设置重置为默认值。要更改该命令的默认值，可以将该命令放在脚本的自动执行区域（脚本的顶部）。

相关命令

[Send\(See 20.12\)](#), [ControlSend\(See 20.6\)](#), [SendMode\(See 20.13\)](#), [SetMouseDelay\(See 23.8\)](#), [SetControlDelay\(See 28.1.13\)](#), [SetWinDelay\(See 28.9\)](#), [SetBatchLines\(See 17.20\)](#), [Click\(See 23.1\)](#)

示例

```
SetKeyDelay, 0
```

20.15 SetNumScrollCapsLockState

设置 Capslock/NumLock/ScrollLock 的状态。也可以让这些键保持开启或关闭的状态。

`SetCapsLockState [, State]`

`SetNumLockState [, State]`

`SetScrollLockState [, State]`

参数

State	<p>如果省略这个参数，将会清除按键的“AlwaysOn/Off”状态（如果设置了的话）。否则，这个参数可以是下面几个词：</p> <p>On: 开启按键，同时清除按键的“AlwaysOn/Off”状态（如果设置了的话）。</p> <p>Off: 关闭按键，同时清除按键的“AlwaysOn/Off”状态（如果设置了的话）。</p> <p>AlwaysOn: 让按键保持一直启用状态（在 Windows95 中无效）。</p> <p>AlwaysOff: 让按键保持一直关闭状态（在 Windows95 中无效）</p>
--------------	--

注意

由于系统本身的限制，这些命令在 Windows Me/98/95 中不一定有用。在 Win9x 中，想切换这些按键到相反的状态，可以使用 [Send\(See 20.12\)](#) 命令，例如：`Send {Capslock}`

要保持一个键的 *AlwaysOn* 或者 *AlwaysOff* 状态，脚本会自动使用 [键盘勾子\(See 20.2\)](#)。

相关命令

[SetStoreCapslockMode\(See 20.16\)](#), [GetKeyState\(See 20.7\)](#)

示例

```
SetNumlockState, on
SetScrollLockState, AlwaysOff
```

20.16 SetStoreCapslockMode

设置在一个 [Send\(See 20.12\)](#) 命令之后是否恢复 CapsLock 的状态。

`SetStoreCapslockMode, On|Off`

参数

On Off	<p>On: 这个是脚本的默认值：如果一个 Send(See 20.12) 命令暂时改变了 CapsLock 的状态，那么命令结束之后，它会自动恢复原来的状态。</p> <p>Off: CapsLock 的状态完全不会被改变。所以，如果 CapsLock 处于开启状态的话，Send(See 20.12) 命令输出的字母都是大小写颠倒的。</p>
--------	---

注意

由于系统的限制，这个命令在 Windows Me/98/95 中可能不会有效。也就是说在 [Send\(See 20.12\)](#) 和 [ControlSend\(See 20.6\)](#) 命令中 CapsLock 不能总是保持关闭的状态。即使它被关闭了，在命令完成之后它也不会自动恢复原来的状态。

这个命令其实不怎么常用，因为默认设置在大多数情况下都非常好使。

每一个新运行的 [Thread/线程\(See 30.19\)](#)（例如一个 [hotkey/热键\(See 4.\)](#), [custom menu item/自定义菜单\(See 22.13\)](#), 或 [timed/定时器\(See 17.21\)](#) 事件）会将该命令的设置重置为默认值。要更改该命令的默认值，可以将命令放在脚本的自动执行区域（脚本的顶部）。

相关命令

[SetCaps/Num/ScrollLockState\(See 20.15\)](#), [Send\(See 20.12\)](#), [ControlSend\(See 20.6\)](#)

示例

```
SetStoreCapslockMode, off
```

21. 数学命令

21.1 EnvAdd (+=, ++)

把[变量\(See 9.\)](#)设置为它自身加上给定的值的总和 (也能从一个[日期-时间\(See 16.26\)](#)的值里加上或减去时间)。同义于: `var += value`

`EnvAdd, Var, Value [, TimeUnits]`

`Var += Value [, TimeUnits]`

`Var++`

参数

Var	要操作的 变量(See 9.) 名称。
Value	任何整数、浮点数或 表达式(See 9.) 。
TimeUnits	<p>如果提供, 此参数指示命令添加 <code>Value</code> 给 <code>Var</code>, 将 <code>Var</code> 作为一个 YYYYMMDDHH24MISS(See 16.26) 格式的日期-时间标记并且把 <code>Value</code> 作为整数或浮点数单位来添加 (要做减法, 指定一个负值)。<code>TimeUnits</code> 可以是 <code>Seconds</code>, <code>Minutes</code>, <code>Hours</code> 或 <code>Days</code> (或者仅用它们的第一个字母)。</p> <p>如果 <code>Var</code> 是一个空变量, 将用当前的时间来代替它。如果 <code>Var</code> 包含一个无效的时间标记或一个早于 1601 的年份, 又或者如果 <code>Value</code> 是非数值, <code>Var</code> 将被设为空来表明有问题。</p> <p>内置变量 <code>A_Now</code> 包含 YYYYMMDDHH24MISS(See 16.26) 格式的当前本地时间。</p> <p>要计算两个时间标记之间的时间量, 请用 EnvSub(See 21.4)。</p>

注意

此命令等于这种速记形式: `Var += Value`

变量通过使用 `Var++, Var--, ++Var` 或 `--Var` 能被加上或减去 1。

如果 `Var` 或 `Value` 为空或者不以数字开头, 为了计算的目的它将被视为 0 (除了在一个表达式内部使用时, 除了使用 `TimeUnits` 参数时)。

如果 `Var` 或者 `Value` 包含小数点, 最后的结果将是一个由 [SetFormat\(See 21.10\)](#) 设置格式的浮点数。

相关命令

[EnvSub\(See 21.4\)](#), [EnvMult\(See 21.3\)](#), [EnvDiv\(See 21.2\)](#), [SetFormat\(See 21.10\)](#),
[Expressions\(See 9.\)](#), [If var is \[not\] type\(See 21.8\)](#), [SetEnv\(See 22.19\)](#), [FileGetTime\(See 16.13\)](#)

示例

```
EnvAdd, MyCount, 2
MyCount += 2 ;等同于上面的命令

var1 = ;使它为空, 以便在下面使用当前的时间代替它。
var1 += 31, days

MsgBox, %var1% ;答案将是当前 31 天后的日期。
```

翻译: 天堂之门 menk33@163.com 2008 年 10 月 18 日

21.2 EnvDiv (/=)

将[变量\(See 9.\)](#)设置为它自身除以给定的值。同义于: `var /= value`

`EnvDiv, Var, Value`

参数

Var	要操作的 变量(See 9.) 名称。
Value	任何整数、浮点数或者 表达式(See 9.) 。

注意

此命令等同于这种速记形式: `Var /= Value`

除以零会导致脚本载入时弹出错误信息窗口(如果可能的话); 要不就使变量为空。

如果 `Var` 或 `Value` 为空或者不以数字开头, 那么出于计算的目的它将被视为 0 (除了在一个表达式内部使用时, 例如 `Var := X /= Y`)。

如果 `Var` 或 `Value` 包含小数点, 那么最后的结果将是一个浮点数, 其格式可由 [SetFormat\(See 21.10\)](#) 命令设置。否则, 结果将被截断 (例如 19 除以 10 将得出 1)。

相关命令

[EnvAdd\(See 21.1\)](#), [EnvSub\(See 21.4\)](#), [EnvMult\(See 21.3\)](#), [SetFormat\(See 21.10\)](#),
[Expressions\(See 9.\)](#), [If var is \[not\] type\(See 21.8\)](#), [SetEnv\(See 22.19\)](#), [bitwise operations \(Transform\)\(See 21.11\)](#)

示例

```
EnvDiv, MyCount, 2
MyCount /= 2 ;等同于上面的命令
```

翻译: 天堂之门 menk33@163.com 2008 年 10 月 18 日

21.3 EnvMult (*=)

将变量(See 9.)设置为它自身乘以给定的值。同义于: var *= value

EnvMult, Var, Value

参数

Var	要操作的变量(See 9.)名称。
Value	任何整数、浮点数或者表达式(See 9.)。

注意

此命令等同于这种速记形式: Var *= Value

如果 *Var* 或 *Value* 为空或者不以数字开头, 那么出于计算的目的它将被视为 0 (除了在一个表达式内部使用时, 例如 *Var := X *= Y*)。

如果 *Var* 或 *Value* 包含小数点, 那么最后的结果将是一个浮点数, 其格式可由 SetFormat(See 21.10) 命令设置。

相关命令

[EnvAdd\(See 21.1\)](#), [EnvSub\(See 21.4\)](#), [EnvDiv\(See 21.2\)](#), [SetFormat\(See 21.10\)](#),
[Expressions\(See 9.\)](#), [If var is \[not\] type\(See 21.8\)](#), [SetEnv\(See 22.19\)](#), [bitwise operations \(Transform\)\(See 21.11\)](#)

示例

```
EnvMult, MyCount, 2
MyCount *= 2 ;等同于上面的命令
```

翻译: 天堂之门 menk33@163.com 2008 年 10 月 18 日

21.4 EnvSub (-=, --)

把变量(See 9.)设置为它自身减去给定的值 (也能比较日期-时间(See 16.26)的值)。同义于: var -= value

EnvSub, Var, Value [, TimeUnits]

Var -= Value [, TimeUnits]

Var--

参数

Var	要操作的变量(See 9.)名称。
Value	任何整数、浮点数或表达式(See 9.)。(当提供了 TimeUnits 时将不支持表达式)。
TimeUnits	<p>如果提供, 此参数指示命令从 <i>Var</i> 减去 <i>Value</i> 就像它们都是 YYYYMMDDHH24MISS(See 16.26) 格式的日期-时间标记。TimeUnits 可以是 Seconds, Minutes, Hours 或 Days (或者仅用它们的第一个字母)。如果 <i>Value</i> 为空, 将用当前的时间来代替它。同样, 如果 <i>Var</i> 是一个空变量, 将用当前的时间来代替它。</p> <p>结果总会向调整成整数。例如, 如果两个时间标记之间实际的差是 1.999 天, 它将报称 1 天。如果需要更高的精度, 可指定 TimeUnits 为 Seconds 并把结果除以 60.0, 3600.0 或 86400.0。</p> <p>如果 <i>Var</i> 或 <i>Value</i> 是一个无效的时间标记或者包含一个早于 1601 的年份, <i>Var</i> 将被设为空来表明有问题。</p> <p>内置变量 A_Now 包含 YYYYMMDDHH24MISS(See 16.26) 格式的当前本地时间。</p> <p>要精确地测定在两个事件之间消逝的时间, 可用 A_TickCount 方法(See 9.), 因为它提供了毫秒级的精度。</p> <p>要从一个时间标记加上或者减去某个 seconds, minutes, hours 或 days 的数值, 也可用 EnvAdd(See 21.1) (减法是通过添加一个负数来获得的)。</p>

注意

此命令等于这种速记形式: `Var -= Value`

变量通过使用 `Var++`, `Var--`, `++Var` 或 `--Var` 能被加上或减去 1。

如果 *Var* 或 *Value* 为空或者不以数字开头, 为了计算的目的它将被视为 0 (除了在一个表达式内部使用时, 除了使用 TimeUnits 参数时)。

如果 *Var* 或者 *Value* 包含小数点, 最后的结果将是一个由 [SetFormat](#)(See 21.10) 设置格式的浮点数。

相关命令

[EnvAdd](#)(See 21.1), [EnvMult](#)(See 21.3), [EnvDiv](#)(See 21.2), [SetFormat](#)(See 21.10),
[Expressions](#)(See 9.), [If var is \[not\] type](#)(See 21.8), [SetEnv](#)(See 22.19), [FileGetTime](#)(See
16.13)

示例

```
EnvSub, MyCount, 2
MyCount -= 2 ;等同于上面的命令
```

```
var1 = 20050126
var2 = 20040126
EnvSub, var1, %var2%, days
MsgBox, %var1% ;答案会是 366 天, 因为 2004 是个闰年。
```

翻译: 天堂之门 menk33@163.com 2008 年 11 月 6 日

21.5 if (expression)

注意

包含表达式的 `if` 语句不同于[传统的 if\(See 17.10\)](#) 例如 `If FoundColor <> Blue`, 其在单词 "if" 后面的字符是左括号。尽管通常情况下会将整个表达式围在圆括号内, 不过也可以这样 `if(x > 0) and (y > 0)`。此外, 如果紧跟 "if" 后的是一个[函数调用\(See 10.\)](#)或者是一个运算符例如 "not" 或 "! ", 那么左括号将被整个省略掉。

如果 `if` 的条件语句为真(除过空字符串或数字 0 之外的其他结果), 下面的一行或者 [block\(See 17.2\)](#) (语句块) 将会被执行。否则如果有相应的 `ELSE`, 那么将执行 `ELSE` 下面的一行或者语句块。

当一个 `IF` 或 `ELSE(See 17.5)` 语句有许多行时, 那些行要用 [braces\(See 17.2\)](#) (大括号) 围起来。然而, 如果 `IF` 或 `ELSE` 语句只有一行, 那么大括号就可以不用。请看本页末尾的例子。

单个正确的大括号(OTB)类型可以和表达式 `if` 语句一起用(但不能用在[传统的 if 语句\(See 17.10\)](#))。例如:

```
if (x < y) {
    ...
}

if WinExist("无标题 - 记事本") {
    WinActivate
}

if IsDone {
    ...
} else {
    ...
}
```

不像 "else" 语句, 它的右边支持紧跟任意类型的语句。`if` 语句的右边仅支持一个 "{"。

做个相关的提示, "[if var \[not\] between LowerBound and UpperBound\(See 21.7\)](#)" 命令检查变量的值是否处在两个值之间, 而 "[if var \[not\] in value1,value2\(See 27.4\)](#)" 命令能检查变量的内容是否在一个值的列表中存在。

相关命令

[Expressions](#)(See 9.), [Assign expression \(:=\)](#)(See 21.12), [if var in/contains MatchList](#)(See 27.4),
[if var between](#)(See 21.7), [IfInString](#)(See 27.3), [Blocks](#)(See 17.2), [Else](#)(See 17.5),
[While-loop](#)(See 17.24)

示例

```
if (A_Index > 100 or Done)
    return

if (A_TickCount - StartTime > 2*MaxTime + 100)
{
    MsgBox 经过太长时间了。
    ExitApp
}

if (Color = "Blue" or Color = "White")
{
    MsgBox 颜色是允许值之一。
    ExitApp
}
else if (Color = "Silver")
{
    MsgBox 银色不是允许的颜色。
    return
}
else
{
    MsgBox 该颜色不可识别。
    ExitApp
}
```

翻译: yugi 修正: 天堂之门 menk33@163.com 2008年11月14日

21.6 If/IfEqual/IfLess/IfGreater

指定一个[变量](#)(See 9.)和一个值比较得出 TRUE 时要执行的命令。当存在多个命令时，将它们括入一个[区块](#)(See 17.2)。

IfEqual, var, value (同: if var = value)
IfNotEqual, var, value (同: if var <> value) (!= 能用来替代 <>)
IfGreater, var, value (同: if var > value)
IfGreaterOrEqual, var, value (同: if var >= value)

IfLess, var, value (同: if var < value)

IfLessOrEqual, var, value (同: if var <= value)

If var ;如果变量的内容为空或为 0, 那么它被视为 false。否则, 它为 true。

另外可见: [IfInString](#)(See 27.3)

参数

var	变量(See 9.)名称。
value	原义的字串、数字或者变量引用(例如 %var2%)。Value 可以被省略, 如果你想要将 var 与一个空字符串(空白的)相比较。

说明

如果 var 和 value 都是纯粹的数值型, 它们将被作为数字比较, 而不是作为字符串。否则, 它们将作为字符串按字母顺序来比较 (就是说, 字母的次序将决定 var 是否小于、等于或者大于 value) 。**译注: A<B**

当一个 IF 或者 ELSE(See 17.5) 拥有多行时, 那些行必须被括入大括号。例如:

```
if count <= 0
{
    WinClose Untitled - Notepad
    MsgBox There are no items present.
}
```

然而, 如果仅有一行属于 IF 或者 ELSE, 大括号就是可选的。

如果你使用命令名称类型, 那么另一个命令就只能出现在 IF 语句的同一行。换句话说, 这些是有效的:

```
IfEqual, x, 1, Sleep, 1
IfGreater, x, 1, EnvAdd, x, 2
```

但是这些是无效的:

```
if x = 1 Sleep 1
IfGreater, x, 1, x += 2
```

单个正确的大括号(OTB)类型(See 17.11)不能和这些类型的 IF 语句一起使用。它只能和表达式类型的 IF 语句一起使用。

做一个相关的提示, "if var [not] between LowerBound and UpperBound(See 21.7)" 命令检查一个变量是否在两个 values 之间, 而 "if var [not] in value1,value2(See 27.4)" 能用来检查一个变量的内容是否在 values 列表里存在。

相关命令

[IF \(expression\)\(See 17.11\)](#), [StringCaseSense\(See 27.13\)](#), [Assign expression \(:=\)\(See 21.12\)](#),
[if var in/contains MatchList\(See 27.4\)](#), [if var between\(See 21.7\)](#), [IfInString\(See 27.3\)](#),
[Blocks\(See 17.2\)](#), [Else\(See 17.5\)](#)

范例

```
if counter >= 1
    Sleep, 10

if counter >=1      ;如果一个 IF 有多行, 将那些行插入大括号:
{
    WinClose, Untitled - Notepad
    Sleep 10
}

if MyVar = %MyVar2%
    MsgBox MyVar 和 MyVar2 的内容是相同的。
else if MyVar =
{
    MsgBox, 4,, MyVar 是空的/空白的。继续?
    IfMsgBox, No
        Return
}
else if MyVar <> ,
    MsgBox value 在 MyVar 里不是一个逗号。
else
    MsgBox value 在 MyVar 里是一个逗号。

if Done
    MsgBox 变量 Done 即不为空也不为零。
```

翻译: 天堂之门 menk33@163.com 2008 年 11 月 14 日

21.7 If var [not] between Low and High

检测 [variable's\(变量\)\(See 9.\)](#) 内容是否在 2 个数字或字符串之间(包含边界)。

`if Var between LowerBound and UpperBound`

`if Var not between LowerBound and UpperBound`

参数

Var	待检测的 变量(See 9.)
LowerBound	指定的范围下界, <code>Var</code> 必须大于或等于这个数字,字符串或变量引用
UpperBound	指定的范围上界, <code>Var</code> 必须小于或等于这个数字,字符串或变量引用

注意

如果这 3 个参数全为数字, 它们就会作为数字而非字符串进行比较。否则, 作为字符串进行比较 (也就是说, 字母字典序来决定 `Var` 是否在指定的范围之内)。在这种情况下, "[StringCaseSense\(See 27.13\) On](#)" 可以用来设置是否区分大小写。

如下操作符 "**between**", "**is**", "**in**", 和 "**contains**" 不能用于 [expressions\(表达式\)\(See 9.\)](#) 中。

相关命令

[IfEqual/Greater/Less\(See 17.10\)](#), [if var in/contains MatchList\(See 27.4\)](#), [if var is type\(See 21.8\)](#), [IfInString\(See 27.3\)](#), [StringCaseSense\(See 27.13\)](#), [EnvAdd\(See 21.1\)](#), [Blocks\(See 17.2\)](#), [Else\(See 17.5\)](#)

示例

```
if var between 1 and 5
```

`MsgBox, %var% is in the range 1 to 5, inclusive.`

```
if var not between 0.0 and 1.0
```

`MsgBox %var% is not in the range 0.0 to 1.0, inclusive.`

```
if var between %VarLow% and %VarHigh%
```

`MsgBox %var% is between %VarLow% and %VarHigh%.`

```
if var between blue and red
```

```
MsgBox %var% is alphabetically between the words blue and red.
```

```
LowerLimit = 1
```

```
UpperLimit = 10
```

```
InputBox, UserInput, Enter a number between %LowerLimit% and %UpperLimit%
```

```
if UserInput not between %LowerLimit% and %UpperLimit%
```

```
    MsgBox Your input is not within the valid range.
```

21.8 If var is [not] type

检查一个[变量\(See 9.\)](#)的内容是否为数值、大写字母或其他。

```
if var is type
if var is not type
```

参数

var	变量(See 9.) 名。
type	请见下面的说明。

注意

支持的 *Types*:

integer	若 <i>var</i> 非空且包含一个没有小数点的纯数字字符串(十进制或十六进制)，则为真。允许开头与结尾有空格和 tab 。字符串可以是加号或减号开头。
float	若 <i>var</i> 非空且包含一个浮点数；也就是说一个包含小数点的纯数字字符串，则为真。允许开头与结尾有空格和 tab 。字符串可以是加号、减号或小数点开头。
number	若 <i>var</i> 包含一个整数或浮点数(两者都在上面解释过了)，则为真。
digit	若 <i>var</i> 为空或者只包含由 0 到 9 组成的数字，则为真。不允许出现后面这样的其他字符：空格、 tab 、加号、减号、小数点、十六进制数和 0x 前缀。
xdigit	十六进制数值：除了还允许包含 A 到 F(大写或小写)这些字符外，其他与 <i>digit</i> 相同。在 v1.0.44.09 及之后版本中，也容许存在 0x 前缀。
alpha	若 <i>var</i> 为空或只包含按字母排序的字符，则为真。若在字符串的任何地方有任何数字、空格、 tab 、标点符号或其他不按字母排序的字符出现，则为假。例如，如果 <i>var</i> 包含一个空格后跟一个字母，它将不被视为 <i>alpha</i> 。
upper	若 <i>var</i> 为空或只包含大写字符，则为真。若在字符串的任何地方有任何数字、空格、 tab 、标点符号或其他未大写的字符出现，则为假。
lower	若 <i>var</i> 为空或只包含小写字符，则为真。若在字符串的任何地方有任何数字、空格、 tab 、

	标点符号或其他未小写的字符出现，则为假。
alnum	除了还允许包含 0 到 9 的字符外，其他与 <i>alpha</i> 相同。
space	若 <i>var</i> 为空或只包含由下列字符组成的空白字符：空格 (%A_Space%(See 9.))、tab (%A_Tab%(See 9.)) 或 `t)、换行(`n)、回车(`r)、垂直 tab (`v) 和进纸符(`f)，则为真。
time	<p>若 <i>var</i> 包含一个有效的日期时间标记，则为真，日期时间标记可以全部是 YYYYMMDDHH24MISS(See 16.26) 格式或者仅仅开头部分是。例如，一个 4 位的数字字符串比如 2004 被视为有效。使用 StringLen(See 27.17) 可以确定是否存在额外的时间部分。</p> <p>小于 1601 的年份将被视为无效，因为操作系统通常不支持它们。被视为有效的最大的年份是 9999。</p> <p>可以使用单词 DATE 来代替 TIME，结果相同。</p>

注意：表达式(See 9.)中不支持 "between", "is", "in" 和 "contains" 运算符。

相关命令

%A_YYYY%(See 9.), [SetFormat](#)(See 21.10), [FileGetTime](#)(See 16.13), [IfEqual](#)(See 17.10), [if](#)
[var in/contains](#) [MatchList](#)(See 27.4), [if var between](#)(See 21.7), [StringLen](#)(See 27.17),
[IfInString](#)(See 27.3), [StringUpper](#)(See 27.18), [EnvAdd](#)(See 21.1), [Blocks](#)(See 17.2), [Else](#)(See
17.5)

示例

```
if var is float
    MsgBox, %var% 是一个浮点数。
else if var is integer
    MsgBox, %var% 是一个整数。
if var is time
    MsgBox, %var% 也是一个有效的日期时间。
```

翻译：wz520 修正：天堂之门 menk33@163.com 2008 年 11 月 24 日

21.9 Random

产生一个伪随机数。

Random, OutputVar [, Min, Max]

Random, , NewSeed

参数

OutputVar	存储结果的变量名。被存储的浮点数的格式由 SetFormat(See 21.10) 决定。
Min	可以产生的最小的数。可以是负数，浮点数，或者一个 expression(See 9.) (表达式)。缺省情况下，产生的最小的数是 0 。 允许的最小整数是 -2147483648，浮点数没有限制。
Max	可以产生的最大的数。可以是负数，浮点数，或者一个 expression(See 9.) (表达式)。如果省略，缺省情况下，产生的最大的数是 2147483647 (这也是允许的最大整数 -- 但是浮点数没有限制)。
NewSeed	<p>此模式通过 <i>NewSeed</i> (可以是一个 expression(See 9.) (表达式)) 对随机数产生器重新定义种子。这对随后产生的随机数都有影响。<i>NewSeed</i> 应当是介于 0 和 4294967295 (0xFFFFFFFF) 之间的整数。重新定义种子能够提高所产生随机数的质量和安全性，尤其当 <i>NewSeed</i> 是一个真正的随机数而不是像伪随机数一样质量欠佳时更是这样。通常，种子只需要重新定义一次即可。</p> <p>如果脚本中从来没有重定义过种子，种子使用一个 64 位值的低 32 位作为开始。这个 64 位的值是从 1601 年 1 月 1 日开始 100 纳秒的数目。这个低 32 位值从 0 变化至 4294967295，其中 100 毫秒累计 ~7.2 分钟。</p>

注意

此命令产生一个伪随机数，即一个模拟真正随机数的数字，但是它实际上是一个基于复杂方程式的使得决定或者猜测下一个数字极为困难的数字。

如果 *Min* 或者 *Max* 其中之一被设置为十进制浮点数，最后的结果将是由 [SetFormat\(See 21.10\)](#) 决定格式的浮点数。否则，结果将是一个整数。

结果是浮点数的局限：1) 只有其中 32 位用于产生变量；2) 它偶尔也会比指定的 *Max* 略微大一点点(这是由浮点数内在的不精确性导致的)。

相关命令

[SetFormat\(See 21.10\)](#)

示例

```
Random, rand, 1, 10
```

```
Random, rand, 0.0, 1.0
```

关于原始资料的注释

此函数使用了 Mersenne Twister 随机数产生器，MT19937，原作者：Takuji Nishimura , Matsumoto, Shawn Cokus, Matthe Bellew, Isaku Wada.

Mersenne Twister 是一个用于产生随意数的算法。它在考虑其他随机数产生器的基础上被设计出来。它的循环周期, $2^{19937}-1$, 和它的等分布顺序, 623 dimensions(维度), 都非常之大。与此同时, 这个产生器非常快, 它避免了乘法和除法, 并且从缓存和管道中得益。想了解更多, 请访问原作者的网站 www.math.keio.ac.jp/~matumoto/emt.html

(译者注: 以下是关于算法的法律声明与使用限制, 一般不允许翻译, 但是这个没说, 以下就不翻译了, 不影响大家对此命令的使用^_^)

Copyright (C) 1997 - 2002, Makoto Matsumoto and Takuji Nishimura, All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The names of its contributors may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Do NOT use for CRYPTOGRAPHY without securely hashing several returned values together, otherwise the generator state can be learned after reading 624 consecutive values.

When you use this, send an email to: matumoto@math.keio.ac.jp with an appropriate reference to your work. It would be nice to CC: rjwagner@writeme.com and Cokus@math.washington.edu when you write.

This above has been already been done for AutoHotkey, but if you use the Random command in a publicly distributed application, consider sending an e-mail to the above people to thank them.

21.10 SetFormat

设置数学运算得到的整数或浮点数的格式。

`SetFormat`, `NumberType`, `Format`

参数

NumberType	必须是 <code>INTEGER</code> /整数 或者 <code>FLOAT</code> /浮点数。
Format	<p>上一个参数 <code>NumberType</code> 使用 <code>INTEGER</code> 的话，这个参数必须是表示十六进制的 <code>HEX</code>（可以简写为 <code>H</code>），或者表示十进制的 <code>D</code>。十六进制的数字全部是以 <code>0x</code> 开头的（例如 <code>0xFF</code>）。</p> <p>上一个参数 <code>NumberType</code> 使用 <code>FLOAT</code> 的话，这个参数的格式为 <code>总宽度.小数位数</code>（例如 <code>0.6</code>）。在 v1.0.46.11 以上的版本中，可以在后面添加字母“<code>e</code>”表示使用科学计数法显示结果，例如 <code>0.6e</code> 或者 <code>0.6E</code>（使用大写字母 <code>E</code> 的话在结果中显示的就是大写字母 <code>E</code> 而不是小写字母 <code>e</code>）。注意：在 AutoHotkey 1.x 版本中，使用科学计数法显示的时候，这个参数必须包含小数点，例如 <code>1.0e1</code> 是正确的，而 <code>1e1</code> 是错误的。</p> <p><code>总宽度</code> 一般使用 <code>0</code>，表示计算结果不需要使用空格或 <code>0</code> 来填充。如果 <code>总宽度</code> 大于实际宽度，将会使用空格或 <code>0</code>（参看下面）来填充至指定宽度。注意：<code>总宽度</code> 包括小数点。</p> <p><code>小数位数</code> 表示要显示的小数部分宽度（超出部分的四舍五入）。如果留空或使用 <code>0</code>，结果中将不会显示小数点以及小数部分，也就是说，结果会以整数的方式存储，而不是浮点数。</p> <p>填充：如果 <code>总宽度</code> 大于实际宽度，结果中将会使用空格来填充左边缺少的宽度，也就是说，数字将会右对齐显示。要使数字左对齐显示，在 <code>总宽度</code> 中使用负数。要使用 <code>0</code> 来填充，在 <code>总宽度</code> 中加前缀 <code>0</code>（例如 <code>06.2</code>）。</p>

注意

如果在脚本中未使用这个命令，整数默认使用十进制，浮点数默认使用 `总宽度.小数位数 = 0.6`。每一个新运行的 [Thread/线程\(See 30.19\)](#)（例如一个 [hotkey/热键\(See 4.\)](#), [custom menu item/自定义菜单项\(See 22.13\)](#), 或 [timed/定时器\(See 17.21\)](#) 事件）会将该命令的设置重置为默认值。要更改该命令的默认值，可以将该命令放在脚本的自动执行区域（脚本的顶部）。

你可以检测一个变量是否存储了数字类型的值，使用 [if var is number/integer/float\(See 21.8\)](#)。

一个包含整数的变量与 `0` 相加，可以强制转换为当前设置的整数格式（十六进制或十进制），例如：`Var += 0`。类似的，一个包含浮点数（或整数，需要的话）的变量与 `0.0` 相加，可以强制转换为当前设置的浮点数格式，例如：`Var += 0.0`。如果需要将一个浮点数转换为一个整数，使用 [Round\(\)\(See 10.\)](#), [Ceil\(\)\(See 10.\)](#)，或者 [Floor\(\)\(See 10.\)](#) 会更加方便。

内置变量 **A_FormatInteger** 存储了当前整数格式的设置（`H` 或 `D`）。**A_FormatFloat** 存储了当前浮点数格式的设置。

要对整数使用空格或 `0` 来填充至指定宽度的话，参考下面的例子：

```
Var := "      ".Var ; 引号中包含了 10 个空格。要使用 0 来填充的话，用 0 替换空格。
```

```
StringRight, Var, Var, 10 ; 使用空格将变量 Var 中的数字填充至 10 位宽度。
```

```
Var := SubStr("          " . Var, -9) ; 上面两行代码的整合。
```

浮点格式

因为 AutoHotkey 将所有的变量当作字符串来存储，所以存储的浮点数的精度是由 小数位数 决定。也就是说，一旦浮点数存入了变量中或者变量进行了舍入操作，丧失的精度就再也不能还原了，除非再进行一次相同的运算，并设置更大的 小数位数 。

在将一个浮点数变量赋值给另外一个变量的时候，浮点数前面的空格填充会被删除掉。要避免这种情况，使用 [冒号-等号 运算符 \(:=\)](#)(See 21.12) 或者 [AutoTrim](#)(See 22.3) 。

要使当前设置的浮点数格式在数学运算中生效，例如 [addition](#)(See 21.1)，运算式中至少要有一个数字包含小数点。

一个包含整数的变量可以通过与 0.0 相加转换为浮点数，例如 Var += 0.0 。但是，如果当前的浮点数格式中没有指定 小数位数 ，使用这个来代替：

```
if Var is integer  
    Var = %Var%.0
```

十六进制格式

十六进制数字以 0x 为前缀（例如 0xA9）。它可以使用在任何一个需要数字做参数的地方。例如，[Sleep 0xFF](#) 等同于 [Sleep 255](#)，并且这与当前 [SetFormat](#) 中对整数格式的设置无关。

要将一个整数从十进制转换为十六进制，参考这个例子（也可以反过来转换）：

```
SetFormat, integer, hex  
VariableContainingAnInteger += 0  
SetFormat, integer, d
```

相关命令

[Expression assignment \(:=\)](#)(See 21.12), [EnvAdd](#)(See 21.1), [EnvSub](#)(See 21.4), [EnvMult](#)(See 21.3), [EnvDiv](#)(See 21.2), [AutoTrim](#)(See 22.3), [if var is type](#)(See 21.8)

示例

```
Var = 11.333333  
SetFormat, float, 6.2  
Var -= 1 ; Var 的值为 10.33， 并且开头有一个空格，因为总宽度是 6。  
SetFormat, float, 0.2  
Var += 1 ; Var 的值为 11.33 并且没有空格。
```

```
SetFormat, float, 06.0
Var += 0 ; Var 的值为 000011

SetFormat, integer, hex
Var += 0 ; Var 的值为 0xb

SetFormat, integer, d
```

21.11 Transform

Performs miscellaneous math functions, bitwise operations, and tasks such as ASCII/Unicode conversion.

`Transform, OutputVar, Cmd, Value1 [, Value2]`

参数

OutputVar	The name of the variable in which to store the result of <i>Cmd</i> . SetFormat (See 21.10) determines whether integers are stored as hexadecimal or decimal.
Cmd, Value1/2	See list below.

`Cmd, Value1, Value2`

The *Cmd*, *Value1* and *Value2* parameters are dependent upon each other and their usage is described below.

Unicode [, String]: Retrieves or stores Unicode text on the clipboard. Note: The entire clipboard may be saved and restored by means of [ClipboardAll](#)(See 30.6), which allows "Transform Unicode" to operate without losing the original contents of the clipboard.

There are two modes of operation as illustrated in the following examples:

```
Transform, OutputVar, Unicode ; Retrieves the clipboard's Unicode text as a UTF-8 string.
```

```
Transform, Clipboard, Unicode, %MyUTF_String% ; Places Unicode text onto the clipboard.
```

In the second example above, a literal UTF-8 string may be optionally used in place of `%MyUTF_String%`.

Use a hotkey such as the following to determine the UTF-8 string that corresponds to a given Unicode string:

```
^!u:: ; Control+Alt+U hotkey.
```

MsgBox Copy some Unicode text onto the clipboard, then return to this window and press OK to continue.

Transform, ClipUTF, Unicode

```
Clipboard = Transform, Clipboard, Unicode, %ClipUTF%`r`n
```

MsgBox The clipboard now contains the following line that you can paste into your script. When executed, this line will cause the original Unicode string you copied to be placed onto the clipboard: `n`n%Clipboard%

```
return
```

Notes: 1) Windows 95 requires the *Microsoft Layer for Unicode* to support this command (Windows 98/Me/NT4 or later have built-in support); 2) The "[Send {ASC nnnn}](#)"(See 20.12)" command is an alternate way to produce Unicode characters, but it does not work in all applications.

Asc, String: Retrieves the ASCII code (a number between 1 and 255) for the first character in *String*. If *String* is empty, *OutputVar* will also be made empty. For example: *Transform, OutputVar, Asc, %VarContainingString%*. Corresponding function: [Asc\(String\)](#)(See 10.).

Chr, Value1: Retrieves the single character corresponding to the ASCII code indicated by *Value1*. If *Value1* is not between 1 and 255 inclusive, *OutputVar* will be made blank to indicate the problem. For example: *Transform, OutputVar, Chr, 130*. Corresponding function: [Chr\(Number\)](#)(See 10.).

Deref, String: Expands variable references and [escape sequences](#)(See 29.5) contained inside other variables. Any badly formatted variable references will be omitted from the expanded result. The same is true if *OutputVar* is expanded into itself; in other words, any references to *OutputVar* inside *String*'s variables will be omitted from the expansion (note however that *String* itself can be %*OutputVar*%). In the following example, if var1 contains the string "test" and var2 contains the literal string "%var1%", *OutputVar* will be set to the string "test": *Transform, OutputVar, deref, %var2%*. Within a [function](#)(See 10.), each variable in *String* always resolves to a local variable unless there is no such variable, in which case it resolves to a global variable (or blank if none).

HTML, String: Converts *String* into its HTML equivalent by translating characters whose ASCII values are above 127 to their HTML names (e.g. £ becomes £). In addition, the four characters "&<>" are translated to "&<>. Finally, each linefeed (`n) is translated to
`n (i.e.
 followed by a linefeed).

Mod, Dividend, Divisor: Retrieves the remainder of *Dividend* divided by *Divisor*. If *Divisor* is zero, *OutputVar* will be made blank. *Dividend* and *Divisor* can both contain a decimal point. If negative, *Divisor* will be treated as positive for the calculation. In the following example, the result is 2: *Transform, OutputVar, mod, 5, 3*. Corresponding function: [Mod\(Dividend, Divisor\)](#)(See 10.).

Pow, Base, Exponent: Retrieves *Base* raised to the power of *Exponent*. Both *Base* and *Exponent* may contain a decimal point. If *Exponent* is negative, *OutputVar* will be formatted as a floating point number even if *Base* and *Exponent* are both integers. A negative *Base* combined with a fractional *Exponent* such as 1.5 is not supported; it will cause *OutputVar* to be made blank. See also: [** operator](#)(See 9.).

Exp, N: Retrieves e (which is approximately 2.71828182845905) raised to the *N*th power. *N* may be negative and may contain a decimal point. Corresponding function: [Exp\(N\)](#)(See 10.).

Sqrt, Value1: Retrieves the square root of *Value1*. If *Value1* is negative, *OutputVar* will be made blank. Corresponding function: [Sqrt\(Number\)](#)(See 10.).

Log, Value1: Retrieves the logarithm (base 10) of *Value1*. If *Value1* is negative, *OutputVar* will be made blank. Corresponding function: [Log\(Number\)](#)(See 10.).

Ln, Value1: Retrieves the natural logarithm (base e) of *Value1*. If *Value1* is negative, *OutputVar* will be made blank. Corresponding function: [Ln\(Number\)](#)(See 10.).

Round, Value1 [, N]: If *N* is omitted, *OutputVar* will be set to *Value1* rounded to the nearest integer. If *N* is positive number, *Value1* will be rounded to *N* decimal places. If *N* is negative, *Value1* will be rounded by *N* digits to the left of the decimal point. For example, -1 rounds to the ones place, -2 rounds to the tens place, and-3 rounds to the hundreds place. Note: Round does not remove trailing zeros when rounding decimal places. For example, 12.333 rounded to one decimal place would become 12.300000. This behavior can be altered by using something like [SetFormat](#)(See 21.10), *float*, 0.1 prior to the operation (in fact, [SetFormat](#)(See 21.10) might eliminate the need to use Round in the first place). Corresponding function: [Round\(Number \[, N\]\)](#)(See 10.).

Ceil, Value1: Retrieves *Value1* rounded up to the nearest integer. Corresponding function: [Ceil\(Number\)](#)(See 10.).

Floor, Value1: Retrieves *Value1* rounded down to the nearest integer. Corresponding function: [Floor\(Number\)](#)(See 10.).

Abs, Value1: Retrieves the absolute value of *Value1*, which is computed by removing the leading minus sign (dash) from *Value1* if it has one. Corresponding function: [Abs\(Number\)](#)(See 10.).

Sin, Value1: Retrieves the trigonometric sine of *Value1*. *Value1* must be expressed in radians. Corresponding function: [Sin\(Number\)](#)(See 10.).

Cos, Value1: Retrieves the trigonometric cosine of *Value1*. *Value1* must be expressed in radians. Corresponding function: [Cos\(Number\)](#)(See 10.).

Tan, Value1: Retrieves the trigonometric tangent of *Value1*. *Value1* must be expressed in radians. Corresponding function: [Tan\(Number\)](#)(See 10.).

ASin, Value1: Retrieves the arcsine (the number whose sine is *Value1*) in radians. If *Value1* is less than -1 or greater than 1, *OutputVar* will be made blank. Corresponding function: [ASin\(Number\)](#)(See 10.).

ACos, Value1: Retrieves the arccosine (the number whose cosine is *Value1*) in radians. If *Value1* is less than -1 or greater than 1, *OutputVar* will be made blank. Corresponding function: [ACos\(Number\)](#)(See 10.).

ATan, Value1: Retrieves the arctangent (the number whose tangent is *Value1*) in radians. Corresponding function: [ATan\(Number\)](#)(See 10.).

NOTE: Each of the following bitwise operations has a more concise [bitwise operator](#)(See 9.) for use in expressions.

BitNot, Value1: Stores the bit-inverted version of *Value1* in *OutputVar* (if *Value1* is floating point, it is truncated to an integer prior to the calculation). If *Value1* is between 0 and 4294967295 (0xffffffff), it will be treated as an unsigned 32-bit value. Otherwise, it is treated as a signed 64-bit value. In the following example, the result is 0xfffff0f0 (4294963440): Transform, OutputVar, BitNot, 0x0f

BitAnd, Value1, Value2: Retrieves the result of the bitwise-AND of *Value1* and *Value2* (floating point values are truncated to integers prior to the calculation). In the following example, the result is 0xff00 (65280): Transform, OutputVar, BitAnd, 0xff0f, 0xffff

BitOr, Value1, Value2: Retrieves the result of the bitwise-OR of *Value1* and *Value2* (floating point values are truncated to integers prior to the calculation). In the following example, the result is 0xf0f0 (61680): Transform, OutputVar, BitOr, 0x000f, 0x00f0

BitXor, Value1, Value2: Retrieves the result of the bitwise-EXCLUSIVE-OR of *Value1* and *Value2* (floating point values are truncated to integers prior to the calculation). In the following example, the result is 0xff00 (65280): Transform, OutputVar, BitXor, 0xf00f, 0x0f0f

BitShiftLeft, Value1, Value2: Retrieves the result of shifting *Value1* to the left by *Value2* bit positions, which is equivalent to multiplying *Value1* by "2 to the *Value2*th power" (floating point values are truncated to integers prior to the calculation). In the following example, the result is 8: Transform, OutputVar, BitShiftLeft, 1, 3

BitShiftRight, Value1, Value2: Retrieves the result of shifting *Value1* to the right by *Value2* bit positions, which is equivalent to dividing *Value1* by "2 to the *Value2*th power", truncating the remainder (floating point values are truncated to integers prior to the calculation). In the following example, the result is 2: Transform, OutputVar, BitShiftRight, 17, 3

注意

Sub-commands that accept numeric parameters can also use [expressions](#)(See 9.) for those parameters.

If either *Value1* or *Value2* is a floating point number, the following *Cmds* will retrieve a floating point number rather than an integer: Mod, Pow, Round, and Abs. The number of decimal places retrieved is determined by [SetFormat](#)(See 21.10).

To convert a radians value to degrees, multiply it by 180/pi (approximately 57.29578). To convert a degrees value to radians, multiply it by pi/180 (approximately 0.01745329252).

The value of pi (approximately 3.141592653589793) is 4 times the arctangent of 1.

相关命令

[SetFormat](#)(See 21.10), [Expressions](#)(See 9.), [EnvMult](#)(See 21.3), [EnvDiv](#)(See 21.2),
[StringLower](#)(See 27.18), [if var is type](#)(See 21.8)

示例

```
Transform, OutputVar, Asc, A ; Get the ASCII code of the letter A.
```

21.12 var :=expression

>":=

```
x := y ; 效率同: x = %y%
x := 5 ; 效率同: x = 5.
x := "literal string" ; 效率同: x = literal string.
```

单词“true”和“false”是内置常量，代表 1 和 0。它们可以增加代码的可读性，例如：

```
CaseSensitive := false
ContinueSearch := true
```

这个命令，以及所有接收 *OutputVar* 参数的命令都可以创建一个 [数组](#)(See 30.5)，只要让 *OuputVar* 包含一个对其它变量的引用就行了。例如 `Array%j% := Var/100 + 5`。查看 [数组](#)(See 30.5) 获取更多信息。

相关命令

[Expressions](#)(See 9.), [IF \(expression\)](#)(See 17.11), [Functions](#)(See 10.), [SetEnv](#)(See 22.19),
[EnvSet](#)(See 15.3), [EnvAdd](#)(See 21.1), [EnvSub](#)(See 21.4), [EnvMult](#)(See 21.3), [EnvDiv](#)(See
21.2), [If](#)(See 17.10), [Arrays](#)(See 30.5)

示例

```
Var := 3
Var := "literal string"
Var := Price * (1 - Discount/100)
```

```

Finished := not Done or A_Index > 100

if not Finished

{
    FileAppend, %NewText%`n, %TargetFile%

    return
}

else

    ExitApp

```

22. 杂项命令

22.1 #NoTrayIcon

不显示一个托盘图标。

#NoTrayIcon

在一个脚本的任何位置指定它，当脚本被运行时将防止那个脚本托盘图标的显示（即使脚本被编译进一个 EXE）。

如果你在一个有热键的脚本使用它，你也许应该捆绑一个热键到 [ExitApp\(See 17.7\)](#) 命令。否则，将没有简单的方法退出程序（除了重启计算机或者终止进程）。例如：#x::ExitApp

托盘图标能在脚本运行的任何时候通过使用命令 [menu\(See 22.13\)](#), [tray, Icon](#) 或者 [menu\(See 22.13\)](#), [tray, NoIcon](#) 来使其消失或再现。在脚本的最顶端使用 [menu\(See 22.13\)](#), [tray, NoIcon](#) 的唯一缺点是托盘图标可能在脚本一开始运行时短暂地可见。要避免这种情况，用 [#NoTrayIcon](#) 代替。

如果托盘图标当前被隐藏，内置变量 [A_IconHidden](#) 为 1，或者反之为 0。

相关命令

[Menu\(See 22.13\)](#), [ExitApp\(See 17.7\)](#)

范例

```
#NoTrayIcon
```

翻译：天堂之门 menk33@163.com 2008 年 6 月 13 日

22.2 #SingleInstance

在一个脚本已经运行时决定是否允许它再次运行。

```
#SingleInstance [force|ignore|off]
```

参数

force ignore off	<p>这个参数决定了脚本的上个实例正在运行时，启动该脚本将会发生什么动作：</p> <p>单词 FORCE 将跳过对话框，并自动地替换旧的实例，其实际上同 Reload(See 20.1.13) 命令相似。</p> <p>单词 IGNORE 将跳过对话框，并让旧的实例运行。也就是说，试图启动一个已运行的脚本将被忽略。</p> <p>单词 OFF 允许多个脚本实例同时运行。</p> <p>如果这个参数被省略，将显示一个对话框，询问是保持旧的实例还是使用新的实例来替换它。</p>
------------------	---

注意

一个包含 [hotkeys\(See 4.\)](#)、[hotstrings\(See 5.\)](#)、[#Persistent\(See 29.21\)](#)、[OnMessage\(\)\(See 18.11\)](#) 或者 [Gui\(See 19.3\)](#) 命令的脚本，默认是单实例的。使用上述方法能取消或修改这一特性。

相关命令

[Reload\(See 20.1.13\)](#), [#Persistent\(See 29.21\)](#)

示例

```
#SingleInstance force
#SingleInstance ignore
#SingleInstance off
```

翻译：坛友 lwjeee 修正：天堂之门 menk33@163.com 2008年8月3日

22.3 AutoTrim

设置在像 ["Var1 = %Var2%"\(See 22.19\)](#) 这样的表达式中，给 **Var1** 赋值的时候是否自动省略 **Var2** 中首尾的空格和 Tab。

AutoTrim, On|Off

参数

On Off	<p>On: 在像 "Var1 = %Var2%"(See 22.19) 这样的表达式中，给 Var1 赋值的时候自动省</p>
--------	--

	<p>略 Var2 首尾的空格和 Tab。这是默认设置。</p> <p>Off: 不省略首尾的空格和 Tab。但是即使在这种设置下，显式的 Tab 和空格（包括 `t）仍然会被自动省略掉。比如，在 <code>Var1 = 't%Var2%</code> 中，仍然会自动省略掉 `t。为避免这种情况，可以使用如下语句：</p> <pre>Var1 = %A_Tab%(See 9.)%Var2%%A_Space%(See 9.) ; AutoTrim 必须在 Off 的情况下才有效。</pre> <p><code>Var1 := "`t" . Var2 . " " ; AutoTrim 的设置对这种情况完全没有影响，因为它是一个 expression/表达式(See 9.)。</code></p>
--	--

注意

如果在脚本中不使用这个命令，它默认是 On。

内置变量 **A_AutoTrim** 存储了当前的设置（On 或 Off）。

内置变量 **A_Space**(See 9.) 和 **A_Tab**(See 9.) 分别存储了单个空格和单个 Tab。

AutoTrim 不影响 **expression assignments/表达式任务**(See 21.12)，例如 `Var := " string "`。也就是说，在这种情况下，首尾的空格和 Tab 总是被保留的。

每一个新运行的 **Thread/线程**(See 30.19)（例如一个 **hotkey/热键**(See 4.)，**custom menu item/自定义菜单**(See 22.13)，或 **timed/定时器**(See 17.21) 事件）会将 AutoTrim 的设置重置为默认值。要更改 AutoTrim 的默认值，可以将命令放在脚本的自动执行区域（脚本的顶部）。

相关命令

[SetEnv](#)(See 22.19)

示例

`AutoTrim, Off`

`NewVar1 = %OldVar% ; 如果 OldVar 中首尾有空格或 Tab，NewVar1 中也会保留。`

`NewVar2 = %A_Space% ; 在 AutoTrim 为 Off 的情况下可以这样分配一个空格给 NewVar2。`

`Var1 := "`t" . Var2 . " " ; AutoTrim 的设置不影响它因为它是一个 expression/表达式(See 9.)。`

22.4 BlockInput

开启或关闭用户通过鼠标和键盘与计算机交互的能力。

[BlockInput, Mode](#)

参数

Mode	<p>Mode 1: 是下列的一个单词：</p> <p>On: 用户被阻止与计算机交互(鼠标和 键盘输入无效)。</p> <p>Off: 输入被重新启用。</p> <p>Mode 2 (在 Windows 9x 上无效): 此模式独立于其他两个模式进行操作。例如，<i>BlockInput On</i> 将继续阻止输入直到 <i>BlockInput Off</i> 被使用，即使下面的某个单词也在生效。</p> <p>Send: 用户的鼠标和键盘输入在 Send(See 20.12) 或 SendRaw(See 20.12) 命令正进行时将被忽略(仅对传统的 SendEvent mode(See 20.13)(发 送事件模式) 而言)。此参数阻止用户的键击去扰乱模拟的键击流程。当 <i>Send</i> 命令结束时，输入被重新启用(除非仍然被之前使用的一个 <i>BlockInput On</i> 阻止)。</p> <p>Mouse: 用户的鼠标和键盘输入在 Click(See 23.1) 、MouseMove(See 23.6) 、MouseClick(See 23.3) 或 MouseClickDrag(See 23.4) 命令正进行时被忽略(仅对传统的 SendEvent mode(See 20.13) 而言)。此参数阻止用户的鼠标移动和点击去扰乱模拟的鼠标事件。当鼠标命令结束时，输入被重新启用(除非仍然被之前使用的一个 <i>BlockInput On</i> 阻止)。</p> <p>SendAndMouse: 上面两种参数模式的组合。</p> <p>Default: 关掉 <i>Send</i> 和 <i>Mouse</i> 两种参数模式，但不改变当前阻止的输入状态。例如，如果 <i>BlockInput On</i> 当前生效中，使用 <i>BlockInput Default</i> 不会将它关掉。</p> <p>Mode 3 (在 Windows 9x 上无效；需要 v1.0.43.11+): 此模式独立于其他两个模式进行操作。例如，如果 <i>BlockInput On</i> 和 <i>BlockInput MouseMove</i> 同时生效中，鼠标移动将被一直阻止直到两个模式都关掉。</p> <p>MouseMove: 鼠标指针不会对用户的物理的鼠标移动做出反应而移动(DirectInput 程序可能是一个例外)。当脚本第一次使用此命令时，mouse hook(See 20.3)(鼠标钩子) 被装载(如果它还没被装载的话)。另外，脚本变成 persistent(See 29.21)(持 久的)，意味着应该使用 ExitApp(See 17.7) 命令来终止它。鼠标钩子会继续处于装载状态直到下次使用了 Suspend(See 17.23) 或 Hotkey(See 20.1.10) 命令，在那时如果不被任何热键或热字串需要的话，它将被移除(见 #Hotstring NoMouse(See 20.1.3))。</p> <p>MouseMoveOff: 允许用户移动鼠标指针。</p>
------	---

注意

优先于 *BlockInput*，最好使用 [SendMode Input](#)(See 20.13) 或 [SendMode Play](#)(See 20.13) 以便键击和鼠标点击变得不可中断。这是因为不像 *BlockInput*，那些模式在发送期间不会抛弃用户输入的东西；而是将这些键击缓存起来并在之后发送。避免用 *BlockInput* 同样避开了像下面段落中描述的需要解决粘滞按键的情况。

如果在用户按住按键时 **BlockInput** 变为激活状态，可能导致这些按键"卡住"。这种情况可以通过在启用 **BlockInput** 前等待按键被松开来避免发生，像在这个例子中一样：

```
^!p::  
KeyWait Control ; 等待按键被松开。为每一个热键的修饰键使用一个 KeyWait 。  
KeyWait Alt  
BlockInput On  
; ... 发送键击和鼠标点击 ...  
BlockInput Off  
return
```

输入阻止即刻自动地关闭每当一个 **ALT** 事件发送时(之后重新启用)。

下面的表格显示了 **BlockInput** 的表现如何随着 Windows 的版本而改变；不过，由于一个 Windows API 的特性，在任何一个平台上按 **Ctrl+Alt+Del** 将重新启用输入。

操作系统	" BlockInput " 效果
Windows 95	无效。
Windows 98/Me	用户输入被阻止并且 AutoHotkey 不能模拟输入。
Windows NT 4 (without ServicePack 6)	无效。
Windows NT 4 (with ServicePack 6)	用户输入被阻止，不过 AutoHotkey 能模拟键击和鼠标点击。
Windows 2000/XP	用户输入被阻止，不过 AutoHotkey 能模拟键击和鼠标点击。

Windows 98/Me: 虽然在 **BlockInput** 期间脚本不能在那些操作系统上发送鼠标点击和键击，但像 **WinMove**(See 28.27) 这样的命令仍能起作用。**ControlSend** (See 20.6)也许也能工作。

某些类型的 **hook hotkeys**(See 20.1.9) 当 **BlockInput** 打开时仍能被触发。包括像 "MButton" (鼠标钩子) 和 "LWin & Space" (用明确的前缀而不是修饰符 "\$#" 的键盘钩子)这样的例子。

当脚本关闭时，输入被自动地重新启用。

相关命令

[SendMode](#)(See 20.13), [Send](#)(See 20.12), [Click](#)(See 23.1), [MouseMove](#)(See 23.6),
[MouseClick](#)(See 23.3), [MouseClickDrag](#)(See 23.4)

示例

```
if A_OSType <> WIN32_WINDOWS ; 也就是说不是 Windows 9x 。  
BlockInput, on  
Run, notepad  
WinWaitActive, 无标题 - 记事本
```

```
Send, {F5} ; 粘贴时间和日期
BlockInput, off
```

翻译: 天堂之门 menk33@163.com 2008 年 8 月 6 日

22.5 ClipWait

等到[剪贴板](#)(See 30.6)包含数据。

ClipWait [, SecondsToWait, 1]

参数

SecondsToWait	如果省略, 此命令会无限地等待。否则, 它将等待不超过这么多秒 (可包含小数点或者是 一个表达式(See 9.))。指定为 0 则等同于 0.5。
1	如果此参数被省略, 命令将更有选择性, 明确地等待文本或文件的出现 ("文本"包括任何 在你粘贴进记事本时会产生文本的东西)。如果此参数为 1 (可以是一个表达式(See 9.)), 命令等待任何类型的数据出现在剪贴板。

ErrorLevel

如果等待时间到期, [ErrorLevel](#)(See 30.8) 会被设置成 1。否则(例如剪贴板包含数据), [ErrorLevel](#) 设置成 0。

说明

使用此命令比一个你自己用来检查剪贴板是否为空的循环要好。这是因为剪贴板从未被此命令打开, 因此它执行地更好并且避免了对可能使用剪贴板的其他应用程序的干扰。

此命令把任何可转换成文本的东西(例如 HTML)视为文本。它也把文件视为文本, 例如那些在资源管理器窗口通过 Control-C 复制的文件。每当剪贴板变量(%clipboard%)在脚本中被使用, 这样的文件就被自动地转换为它们的文件名(以绝对路径)。详见[剪贴板](#)(See 30.6)。

最后的参数是 1 时, 任何数据出现在剪贴板上命令都将满意。它能和 [ClipboardAll](#)(See 30.6) 一起用来保存非文本数据, 例如图片。

当命令处于等待状态, 新的[线程](#)(See 30.19)能通过 [热键](#)(See 4.)、[自定义菜单项](#)(See 22.13)或者[定时器](#)(See 17.21)被运行。

相关命令

[Clipboard](#)(See 30.6), [WinWait](#)(See 28.32), [KeyWait](#)(See 20.10)

范例

```
clipboard = ;清空剪贴板
```

```

Send, ^c

ClipWait, 2

if ErrorLevel

{

    MsgBox, 尝试复制文本到剪贴板失败。

    return

}

MsgBox, clipboard = %clipboard%

return

```

翻译: 天堂之门 menk33@163.com 2008 年 11 月 6 日

22.6 CoordMode

为一些命令设置相对于激活窗口或屏幕的坐标模式。

CoordMode, ToolTip|Pixel|Mouse|Caret|Menu [, Screen|Relative]

参数

Param1	<p>ToolTip: 影响 ToolTip(See 19.15)。</p> <p>Pixel: 影响 PixelGetColor(See 22.15)、PixelSearch(See 22.16) 和 ImageSearch(See 22.10)。</p> <p>Mouse: 影响 MouseGetPos(See 23.5)、Click(See 23.1) 和 MouseMove(See 23.6)/Click(See 23.3)/Drag(See 23.4)。</p> <p>Caret: 影响内置变量 A_CaretX(See 9.) 和 A_CaretY(See 9.)。</p> <p>Menu: 当为 "Menu(See 22.13) Show" 命令指定坐标时影响它。</p>
Param2	<p>如果省略 <i>Param2</i>，默认对屏幕。</p> <p>Screen: 坐标相对于桌面(整个屏幕)。</p> <p>Relative: 坐标相对于激活的窗口。</p>

注意

如果此命令没有使用，除了那些文档上有说明的命令外，所有的命令(例如 [WinMove](#)(See 28.27) 和 [InputBox](#)(See 19.10)) 都使用相对于激活窗口的坐标。

对于此命令来说，每个最近启动的 [thread](#)(See 30.19) (例如一个 [hotkey](#)(See 4.)、[custom menu item](#)(See 22.13) 或 [timed](#)(See 17.21) 子程序) 以新的默认设置开始。这个默认设置可以通过在自动执行部分(脚本的顶部)使用此命令来改变。

相关命令

[Click](#)(See 23.1), [MouseMove](#)(See 23.6), [MouseClick](#)(See 23.3), [MouseClickDrag](#)(See 23.4), [MouseGetPos](#)(See 23.5), [PixelGetColor](#)(See 22.15), [PixelSearch](#)(See 22.16), [ToolTip](#)(See 19.15), [Menu](#)(See 22.13)

示例

```
CoordMode, ToolTip, Screen ; 将 ToolTips 放置在相对于屏幕坐标的地方:
```

```
CoordMode, ToolTip ; 和上面是一样的效果，因为 "screen" 是默认的。
```

翻译：天堂之门 menk33@163.com 2008 年 8 月 12 日

22.7 Critical

防止 [current thread](#)(See 30.19) (当前线程) 被其他线程中断。

Critical [, Off]

Critical 50 ; 请看底部的注释 [bottom of remarks](#).

如果第一个参数被省略(或者是 On)，则 [current thread](#)(See 30.19) (当前的线程) 被设置为 **Critical** (关键线程)这就意味着他不会被其他的线程所中断。如果第一个参数是 Off(或者 0，在 1.0.48+ 版本中)，则当前的线程忽略 "[Thread Interrupt](#)"(See 30.19) 的设置，立即被设置为可以中断的状态。

不同于 [high-priority](#)(See 30.19) (高优先级线程)，如果在 **Critical** 状态的线程运行中产生了另一个事件，则该事件不会被忽略。例如，当用户按下一个热键 [hotkey](#)(See 4.)，但该热键的线程正以 **Critical** 状态运行，这时用户按下的热键会被缓冲，直到当前的线程结束或者被设置为可以中断的非关键状态，这时用户按下的热键才会被一个新的线程所执行。

一个处于 **Critical** 状态的线程也会被紧急事件所中断。紧急事件包括：1) [OnExit](#)(See 17.17) 子程序；2) 任意一个监听的消息值小于 0x312 的 [OnMessage\(\)](#)(See 18.11) 函数(或是被这些消息所触发的 [callback](#)(See 18.14) 函数)；3) 任意一个被当前线程自己所间接触发的(通过 [SendMessage](#)(See 28.1.12) 或 [DII Call](#)(See 18.3) 等函数所触发)为了避免被这些函数的中断，可以临时的关闭这些函数。

当缓冲事件等待被线程执行时，使用"**Critical Off**"不能够立即中断当前的线程。相反，执行中断平均需要花费 5 毫秒。这样在中断前，至少在"**Critical Off**"下面的一行代码 99.999%可能会被执行。你可以通过利用 [Sleep -1](#)(See 17.22) 或用 [WinWait](#)(See 28.32) 等待一个不存在的窗口等方法，进行强制中断。

当一个 [MsgBox](#)(See 19.11) 或者其他对话框显示时，会中断一个 **Critical** 状态的线程，而不同于 "[Thread Interrupt](#)"(See 22.22)，线程会在对话框关闭时恢复 **Critical** 状态。

关于如何保存和还原当前 **Critical** 的设置, 请参考 [A_IsCritical\(See 9.\)](#)。因为 **Critical** 状态是一个特殊设置的线程, 当线程结束, 接下来运行的线程或者重新运行的线程 (如果有的话) 不会再以 **Critical** 状态运行。所以不需要在结束一个线程后再声明 "**Critical Off**"。

如果在自动运行片段 (也就是脚本的最上面的一段代码) 没有声明 **Critical**, 所有的线程不会以 **Critical** 状态运行 (尽管可以设置 "[Thread Interrupt](#)" (See 22.22) 来实现)。相反, 如果在自动运行片段声明了 **Critical** 且不曾关闭 **Critical** 状态, 则每个新运行的线程 (如 [hotkey](#)(See 4.), [custom menu item](#)(See 22.13) (自定义菜单项) 和 [timed](#)(See 17.21) 子程序) 都会以 **Critical** 状态运行。

"[Thread NoTimers](#)(See 22.22)" 命令和 **Critical** 相似, 除了他可以防止被 [timers](#)(See 17.21) 所中断。

在 v1.0.47 及其以后的版本中, 设置 **Critical** 状态同时会对 [current thread](#)(See 30.19) (当前线程) 产生 [SetBatchLines -1](#)(See 17.20) 效果。

在 v1.0.47 及其以后的版本中, 第一参数 指定为一个正数 (例如 **Critical 30**), 表示该线程为关键线程, 但是也改变了检查 内部消息队列的间隔时间。如果第一个参数为 **On**, 会以默认 16 毫秒间隔检查, 当参数为 **Off** 则以 5 毫秒间隔检查。增加推迟消息/事件的延时间隔, 可以给 [current thread](#)(See 30.19) (当前线程) 更多的时间来结束。这样可以减少因为“线程正在运行”而导致消息 [OnMessage\(\)](#)(See 18.11) 和界面事件 [GUI events](#)(See 19.3) 丢失的可能性。然而, 例如 [Sleep](#)(See 17.22) 和 [WinWait](#)(See 28.32) 命令会默认检查消息而忽略这个设置 (一个解决方案是 [DII Call\("Sleep", UInt, 500\)](#))。注意: 检查消息的时间间隔增加的太多有可能减少各种事件的响应, 如 [GUI](#)(See 19.3) 窗口的重绘。

相关命令

[Thread \(command\)](#), (See 22.22)[Threads](#)(See 30.19), [#MaxThreadsPerHotkey](#)(See 20.1.8), [#MaxThreadsBuffer](#)(See 20.1.7), [OnMessage\(\)](#)(See 18.11), [RegisterCallback\(\)](#)(See 18.14), [Hotkey](#)(See 20.1.10), [Menu](#)(See 22.13), [SetTimer](#)(See 17.21)

示例

```
#space::; Win+Space hotkey.
Critical
ToolTip 在这个提示消失之前不会再有新的线程运行
Sleep 3000
ToolTip ; 关闭提示
return ; 热键子程序结束, 以下的线程不会再以关键状态运行。
```

BLooM.2 Fantasy OnLine 08.07.29 翻译

22.8 DII Call

该命令可以调用一个 DLL 文件中的函数, 例如标准的 Windows API 函数。

```
Result := DII Call("[DIIFile\]Function" [, Type1, Arg1, Type2, Arg2, "Cdecl ReturnType"])
```

参数

Result	<code>DllCall</code> 返回调用函数的返回值。如果该函数没有返回值，则结果是未定义的整数类型。如果函数在调用时发生错误 error(See 18.3) ，则返回值为空（即一个空的字符串类型）。
[DllFile\]Function	<p>要调用的函数可表示为，DLL 或者 EXE 文件的名字加上反斜杠 \ 再加上函数名。例如：“MyDLL\MyFunction”（如果未指明文件的后缀，则默认其为”.dll”文件）。如果不是一个绝对路径，则假定 <code>DllFile</code>（DLL 文件）在系统目录或者在 A_WorkingDir(See 9.)（当前工作目录）。</p> <p>如果该函数在 <code>User32.dll</code>, <code>Kernel32.dll</code>, <code>ComCtl32.dll</code>, <code>Gdi32.dll</code> 这几个 DLL 文件中，则 <code>DllFile</code> 文件名可以被忽略。例如 <code>"User32\IsWindowVisible"</code> 等同于 <code>"IsWindowVisible"</code>。对于那些标准的 DLL 文件，其 API 函数的后缀“A”字母也可以被忽略。如“<code>MessageBox</code>”等同于“<code>MessageBoxA</code>”。</p> <p>对于重复调用的情况，使用 loading it beforehand(See 18.3)（预先装载 DLL 文件），可对执行效率有显著的改善。</p> <p>在 v1.0.46.08+ 版本中，这个参数也可以是指向需要调用函数的内存地址的整数。如 COM(See 18.3) 和 RegisterCallback()(See 18.14) 所提供的地址。</p>
Type1, Arg1	这样的每一对数据，表示需要传递给函数的一个参数。参数的个数没有限制。 <code>Type</code> 是用来描述传递参数的类型，具体见下表 types table(See 18.3) （参数的类型的描述）说明， <code>Arg</code> 是给函数传递的参数的具体数值。
Cdecl ReturnType	<p><code>Cdecl</code>: 该参数一般被忽略，因为大多数函数使用的是标准调用习惯而很少是符合“C”标准的调用习惯。如果在忽略这个参数后，产生了错误信息 ErrorLevel An(See 18.3)，<code>n</code> 表示你传递参数的个数，你可能需要填上 <code>Cdecl</code> 参数，声明为以 C 语言方式调用。</p> <p>如果需要在返回的类型前面增加参数 <code>Cdecl</code> 时，请用空格或 tab 将他们分开，如：“<code>Cdecl Str</code>”</p> <p><code>ReturnType</code>（返回值的类型）：如果该函数值类型是一个 32-bit 的有符号整型(<code>Int</code>)，<code>BOOL</code>，或是无返回值，<code>ReturnType</code> 可以被忽略。否则必须从下表中 types table(See 18.3) 指定一个参数类型。也支持传址传递 asterisk suffix(See 18.3)。</p>

参数及返回值的类型描述

Str	<p>表示该参数是一个如“Blue”或 <code>MyVar</code> 的字符串类型。如果函数需要修改这个字符串，或这参数是个不被保护的变量，他其中的内容会被更新。例如，下面的代码将变量 <code>MyVar</code> 里的内容改为大写：<code>DllCall("CharUpper", "str", MyVar)</code>。</p> <p>然而，如果函数需要储存一个超过当前变量容量的大型字符串，确保在调用函数前该变量的容量</p>
-----	---

	<p>足够大。这个可以通过调用函数 VarSetCapacity(MyVar, 123)(See 18.17) 来实现, 其中 123 是为变量 <i>MyVar</i> 保留的容量。</p> <p>一个 <i>str</i> 的参数不能是一个赋值的表达式 (如 <i>i+1</i>) , 如果那样, 函数将不能成功调用, 并且, 错误信息会设置为-2。</p> <p><i>str</i> 类型一般用于描述参数为 LPSTR, LPCSTR, LPTSTR, char * 或类似的类型。</p> <p>同样支持星号后缀 "str *" asterisk variable(See 18.3) (传址引用类型) , 但很少使用。他被用于参数类似于"char **" 或 "LPSTR *"的类型。</p>
Int64	表示该参数是一个 64-bit 的整型 , 其范围是 -9223372036854775808 (-0x8000000000000000) 到 9223372036854775807 (0x7FFFFFFFFFFFFF)
Int	<p>表示该参数是一个 32-bit 的整数类型 (参数大多数情况为整数类型) , 其范围是 -2147483648 (-0x80000000) 到 2147483647 (0x7FFFFFFF)。Int 有时也写作"Long"。</p> <p>Int 也用于描述 BOOL 类型的参数 (BOOL 类型的参数值只能是 1 或 0) 。</p> <p>无符号整型 unsigned(See 18.3) (UInt)也是使用频繁的类型, 可以表示如 DWORD 和 COLORREF 类型。他也被用于所有的句柄类型, 如 HWND, HBRUSH, 和 HBITMAP 类型。</p> <p>注意: 如要传递的值是 NULL 或是指针, 可传递整数 0 。</p>
Short	表示参数为 16-bit 的整型, 其范围是 -32768 (-0x8000) 到 32767 (0x7FFF)。一个无符号短整型 unsigned(See 18.3) Short (UShort) 可用于函数的参数为 WORD 类型。
Char	表示参数为 8-bit 的整型, 其范围是 -128 (-0x80) 到 127 (0x7F)。一个无符号字符 unsigned(See 18.3) character (UChar)可用于函数的参数为 BYTE 类型。
Float	表示参数为 32-bit 的浮点型, 具有 6 位精确度。
Double	表示参数为 32-bit 的浮点型, 具有 15 位精确度。
* or P (suffix)	<p>在上述类型后, 附加一个星号 (可以在之前有空格) 表示函数的参数需要传递的是变量的地 址而不是变量的数值 (当然函数原型要接受该类型的参数)。该参数的值有可能会被函数修改, 一个未被保护的变量作为一个参数传入, 其内容也会随时被更新。例 如, 下面的例子向函数 MyFunction 传递了变量 MyVar 的地址, 其变量 MyVar 可以被 MyFunction 函数随时更新以反映其变化: DllCall("MyDII\MyFunction", "int **", MyVar)</p> <p>一般来说, 星号可用于描述任何参数类型或返回类型是以 "LP" 开头的 (字符串类型 LPSTR 除外, 他应使用 "str"(See 18.3) 类型)。例如常见的 LPDWORD, 他就是指向 DWORD 类型变量的指针类型。DWORD 是一个 32-bit 的整型, 所以用"UInt *" 或 "UIntP"来表示 LPDWORD 类型。</p> <p>注意: "char *"与 "str"(See 18.3) 的类型是不一样的, 因为 "char *" 传递的是一个 8-bit 数字的地址, 但 "str"(See 18.3) 传递的是一个序列字符的地址。并且对于函数需要向变量储存数据的传址引用类型的参数, 不需要调用 VarSetCapacity(See 18.17) 来设置其容量。</p>

	<p>在上述的整数类型前加上前缀 "U" 表示无符号整型（如 UInt64, UInt, UShort 和 UChar）。严格地说，这只对返回值类型和传址引用类型有必要，因为有符号或无符号对参数的传递没有影响（Int64 除外）。</p> <p>一个 32-bit 的无符号整型（UInt）可以替代如 DWORD, HWND, 或类似的类型。一个 HWND 值（窗口句柄）和窗口的 unique ID(See 28.15) 也是一样的。</p> <p>U (prefix) 如果把一个负数类型当成一个无符号整数，则这个负整数会被转换到无符号的范围内。例如，把 -1 作为一个无符号类型 UInt，他的值会变成 0xFFFFFFFF。然而，这种情况对于 64-bit 整数（UInt64）不成立。</p> <p>对由一个函数产生的无符号 64-bit 整型不完全支持。对于大于或等于 0x8000000000000000 的数，若省略前缀 U 后，则会将函数得到的任何负数，转换成一个很大的正数。例如一个被设计其返回值为 UInt64 (64 位整型) 类型的函数，返回结果为 Int64 类型的 -1，则脚本实际返回的值将是 0xFFFFFFFFFFFFFF。</p>
--	---

注意： 当描述一个不包含空格或星号的参数类型或是返回值类型时，其引号可以忽略，如，str 与 "str" 等同，CDecl 与 "CDecl" 等同。另外，字母 P 可以替代星号，以便允许忽略其周围的引号。如：UIntP。

ErrorLevel 错误信息

ErrorLevel(See 30.8) 会赋值为以下几种情况中的一个，不论调用成功或是失败。

0: 成功。

-1 (负 1): 函数 [DllFile\]Function 的参数是一个浮点类型。需要一个字符串或是一个正数的参数。

-2: return type (返回值类型) 或一个描述参数类型的 arg types(See 18.3) 数据无效。这个错误可能是因为向字符串 (str(See 18.3)) 类型变量传递了一个赋值表达式所引起的。

-3: 指定的 DLL 文件不能够使用。若没有指定 DLL 文件的路径，则文件不存在于系统目录或是 A_WorkingDir(See 9.) (当前工作目录)。这个错误也可能是因为用户没有读取文件的权限所引起的。

-4: 无法在 DLL 文件中找到指定的函数。

N (任意正数): 表示函数被调用，但发生了异常错误，错误序号为 N (例如，0xC0000005 表示“读取错误”）。这种情况下函数会返回一个空值（空字符串），但传址引用类型的变量会被更新。例如可能是因为引用了一个无效的 NULL 指针而导致的异常错误。由于 Cdecl(See 18.3) 函数不可能产生 "An" 错误（下一段提到），也可能是因为传入的参数过少而产生了异常。

An (字母 A 后面加一个整数 n) : 表示函数被调用，但传入的参数过少。"n" 表示错误参数列表的字节数。如果 n 为正数，表示传入的参数类型太多（或是参数类型太大），也有可能是函数调用需要 CDecl(See 18.3) 声明。如果 n 是负数，表示传入的参数类型太少。这种情况可以通过改正参数类型来保证函数的正常运行。若函数返回值为空，也可能是因为产生了异常而导致的错误。

异常和 A_LastError

尽管具有内建的异常处理机制，但仍有可能因为使用 `DllCall` 而使脚本假死。这种情况会产生于，当函数不是直接产生了异常而是产生了一些不恰当的数据，如产生了一个野指针或是一个没有结束符的字符串。这可能不是因为 函数本身产生的错误，而是脚本传入了一个不恰当的参数，例如一个野指针或是一个容量不足的字符串 "`str`(See 18.3)"。当对参数类型或返回类的型描述不恰当的时，也可能使脚本假死。例如把一个普通的整数类型错误的声明为一个传址引用类型 `asterisk variable`(See 18.3) 或是 `str`(See 18.3)。

内建的变量 `A_LastError` 包含了系统函数 `GetLastError()` 返回的结果。该函数会在函数调用后被立即执行（对效率的影响不可知）。`A_LastError` 是一个介于 0 与 4294967295 之间的数（通常以十进制表示而不是十六进制）。像 `ErrorLevel`(See 30.8) 一样，`A_LastError` 也是一个独立的线程。也就是说其他的线程 `threads`(See 30.19) 无法中断他。然而 `A_LastError` 可受到 `Run/RunWait`(See 24.5) 的影响。

性能

当需要重复调用 `DLL` 时，事先明确的装载可以显著提高运行效率（标准的 `standard DLL`(See 18.3) 如 `User32` 文件可以不需要，因为他们总是挂起的）。以下是一个在每次调用 `DllCall` 时不需要重复进行 `LoadLibrary` 和 `FreeLibrary` 的例子：

```
hModule := DllCall("LoadLibrary", "str", "MyFunctions.dll") ; 避免了每次在循环中调用
DllCall() 时都进行装载文件。
Loop, C:\My Documents\*.*,,1
result := DllCall("MyFunctions\BackupFile", "str", A_LoopFileFullPath)
DllCall("FreeLibrary", "UInt", hModule) ; 为了释放内存，DLL 在使用完后进行卸载。
```

在 1.0.46.08+ 版本中，通过预先查找函数的地址甚至可以获得更快的性能。例如：

```
; 在下面的例子中，如果 DLL 还没有没加载，使用 LoadLibrary 替换 GetModuleHandle。
MulDivProc := DllCall("GetProcAddress", uint, DllCall("GetModuleHandle", str,
"kernel32"), str, "MulDiv")
Loop 500
    DllCall(MulDivProc, int, 3, int, 4, int, 3)
```

最后，向一个函数传入一个字符串变量，不会改变字符串的长度，如果将变量的地址 `by address`(See 9.) (如，`&MyVar`) 传递给函数而不是以字符串 "`str`(See 18.3)" (特别是一些长字符串) 的形式，可以提高运行效率。以下是一个将字符串改写为大写的例子：`DllCall("CharUpper", uint, &MyVar)`

结构体和数组

一个结构体是一些成员变量（空间）在内存中连续排列的集合。大多数成员变量是整型。

一些函数可以接受结构体的地址（或是一个内存块数组）的类型参数，可以调用他以二进制的形式向结构体写入数据。一般步骤如下：

- 1) 调用 `VarSetCapacity`(`MyStruct, 123, 0`)(See 18.17) 确保目标变量足够大以便储存结构体数据。将 123 替换为结构体的大小。后面的 0 是可选项；他会初始化所有的成员变量为二进制形式的 0，这样通常可以有效的避免在下一步频繁的调用 `NumPut()`。

2) 如果目标函数需要其结构体变量初始化，调用 [NumPut\(123, MyStruct, 4\)\(See 10.\)](#) 初始化其中的每个成员变量为非 0 数据。替换 123 为需要向成员变量存入的整数（或使用 &Var 形式存入一个变量的地址 [address\(See 9.\)](#)）。替换 4 为成员变量的偏移量（“偏移量”请详见第四步）。

3) 调用目标函数，以 UInt 类型的形式传递结构体 MyStruct 的地址 [address\(See 9.\)](#)。例如 [DIIcall\("MyDII\MyFunc", UInt, &MyStruct\)](#)。函数调用后会检查、改变一些传入的结构体成员变量。

4) 利用 [MyInteger := NumGet\(MyStruct, 4\)\(See 10.\)](#) 命令从结构体中获取想要的成员变量。替换 4 为目标成员变量在结构体中的偏移量。第一个成员变量的偏移量总是 0。第二个成员变量的偏移量为第一个成员变量的偏移量 0 加上第一个成员变量的大小（一般为 4 字节）。后面的成员变量的偏移量等于前一个成员变量的偏移量加上前一个成员变量的大小。大多数成员变量 -- 如 DWORD, Int 和 [other types of 32-bit integers\(See 18.3\)](#)（其他 32 位整型）其大小都是 4 字节。

参见 [Structure Examples\(See 18.3\)](#) 结构体详细用法的例子。

已知的局限性

当一个 [variable's address\(See 9.\)](#) 函数的地址（如 &MyVar）传递给一个函数时，这个函数改变了这个变量内容的长度，如果在此时使用了这个变量，可能就会产生错误。想要解决这个问题，有以下两种方法：

1) 使用 "str"(See 18.3) 类型而不是 uint/address 类型；2) 在 v1.0.44.03+ 版本，在调用 DIIcall 命令前，使用 [VarSetCapacity\(MyVar, -1\)\(See 18.17\)](#) 命令更新变量的内部容量。

任何存入二进制数据 0 的变量都会被函数隐藏掉其右边的 0；这种情况下数据不能被多数的命令或函数改变或使用。但可以被当作地址或解引用(& and *[\(See 9.\)](#))及 DIIcall 自身操作。

当一个字符串传入函数后，该函数可能会返回一个地址不同但内容相同的字符串。例如在某些程序中调用 [CharLower\(CharUpper\(MyVar\)\)](#) 可以将 MyVar 里的内容转换为小写，但当在脚本中调用 DIIcall() 完成相同的功能后，MyVar 里的内容可能仍是大写的，这是因为 CharLower 函数是对一个与 MyVar 内容相同的临时字符串变量进行操作（而不是直接对变量 MyVar 操作）。

```
MyVar = ABC
result := DIIcall("CharLower", str, DIIcall("CharUpper", str, MyVar, str), str)
```

如果想让以上的代码直接对变量 MyVar 中的内容进行修改，可以将以上两个带有下划线的 "str" 改为 UInt。这种改变会将 CharUpper 函数的返回值转换为地址，然后再以整数类型传递给函数 CharLower。

VBScript, JScript 和 组件对象模型 (COM)

VBScript 和 JScript 可以通过 [Windows Scripting for AutoHotkey](#) (AHK 中的 Window 脚本) 嵌入到 AHK 脚本中，他们是通过访问 COM 组件来完成的。

同样，可直接调用 DIIcall 命令来使用 COM 对象，参考范例：

www.autohotkey.com/wiki/index.php?title=

相关命令

[PostMessage](#)(See 28.1.12), [OnMessage\(\)](#)(See 18.11), [RegisterCallback\(\)](#)(See 18.14),
[Run](#)(See 24.5), [VarSetCapacity](#)(See 18.17), [Functions](#)(See 10.), [SysGet](#)(See 22.21), [MSDN Library](#)

示例

```
; 例子：调用 Windows API 函数 "MessageBox" 并报告用户按下了哪个按钮。
WhichButton := DllCall("MessageBox", "int", "0", "str", "Press Yes or No", "str", "Title of
box", "int", 4)
MsgBox You pressed button #%WhichButton%.
```

```
; 例子：改变桌面为指定的 Bitmap 文件 (.bmp) 文件。
DllCall("SystemParametersInfo", UInt, 0x14, UInt, 0, Str, A_WinDir . "\winnt.bmp", UInt,
2)
```

```
; 例子：调用 API 函数 "IsWindowVisible" 来判断记事本窗口是否可见。
DetectHiddenWindows On
if not DllCall("IsWindowVisible", "UInt", WinExist("无标题 - 记事本"))
; WinExist()函数返回窗口的句柄。
MessageBox 窗口不可见
```

```
; 例子：调用 API 的 wsprintf() 函数，在一个长度为 10 字节的字符串中填上 432，其他空位用 0
填充，也就是(0000000432)。
VarSetCapacity(ZeroPaddedNumber, 20) ;确保变量的容量足够大以便能够容纳新的字符串。
DllCall("wsprintf", "str", ZeroPaddedNumber, "str", "%010d", "int", 432, "Cdecl") ;需要声
明 Cdecl 的调用方法。
MessageBox %ZeroPaddedNumber%
```

```
; 例子：调用 QueryPerformanceCounter() 函数的一个示范，他提供了比 A_TickCount's(See 9.)
的 10ms 更精确的计时器。
DllCall("QueryPerformanceCounter", "Int64 *", CounterBefore)
Sleep 1000
DllCall("QueryPerformanceCounter", "Int64 *", CounterAfter)
MessageBox % "Elapsed QPC time is " . CounterAfter - CounterBefore
```

```
; 例子：这是一个用快捷键来减慢鼠标的移动速度，以便能够精确的定位。
; 按下 F1 键降低移动速度，释放回复原来的速度。
F1::
SPI_GETMOUSESPEED = 0x70
```

```

SPI_SETMOUSESPEED = 0x71
; 得到当前鼠标的速度以便稍后恢复:
DIICall("SystemParametersInfo", UInt, SPI_GETMOUSESPEED, UInt, 0, UIntP,
OrigMouseSpeed, UInt, 0)
; 现在降低鼠标的移动速度, 第三个参数为速度设定 (范围是 1-20, 10 为默认值):
DIICall("SystemParametersInfo", UInt, SPI_SETMOUSESPEED, UInt, 0, UInt, 3, UInt, 0)
KeyWait F1 ; 这句话可防止按键重复调用 DIICall.
return
F1 up::DIICall("SystemParametersInfo", UInt, 0x71, UInt, 0, UInt, OrigMouseSpeed, UInt,
0); 恢复原来的鼠标速度

```

; 例子: 当传入一个窗口的 ID 和其控件的类名或控件上的文字
; 下面的函数就可以返回该控件的 HWND (句柄)。
; 在 v1.0.43.06+ 版本中: 下面的命令能够更准确的完成该例子的功能。 (该例子省略)
[ControlGet, OutputVar, Hwnd,, ClassNN, WinTitle\(See 28.1.4\)](#)

; 例子: 监视活动窗口并显示其垂直滚动条的位置。
; 因为焦点控件随时更新, 所以需要用到 v1.0.43.06+ 本版中的 [ControlGet Hwnd\(See 28.1.4\)](#) 命令。

```

#Persistent
SetTimer, WatchScrollBar, 100
return

WatchScrollBar:
ActiveWindow := WinExist("A")
if not ActiveWindow ; No active window.
return
ControlGetFocus, FocusedControl, ahk_id %ActiveWindow%
if not FocusedControl ; No focused control.
return
; 在 ToolTip 中显示垂直或水平滚动条的位置:
ControlGet, ChildHWND, Hwnd,, %FocusedControl%, ahk_id %ActiveWindow%
ToolTip % DIICall("GetScrollPos", "UInt", ChildHWND, "Int", 1)
; 最后一个参数若为 1 表示 SB_VERT 垂直滚动条, 为 0 表示 SB_HORZ 水平滚动条。
return

```

; 例子: 这个脚本向文件中写入一些文字, 然后再把它们读入到内存中。 (需要 v1.0.34+ 版本).
; 这个方法可以实现对一个文件同时并发读写的操作。

```

FileSelectFile, FileName, S16,, Create a new file:
if FileName =

```

```

return
GENERIC_WRITE = 0x40000000 ; Open the file for writing rather than reading.
CREATE_ALWAYS = 2 ; 创建新文件 (如果该文件存在则覆盖).
hFile := DllCall("CreateFile", str, FileName, UInt, GENERIC_WRITE, UInt, 0, UInt, 0, UInt,
CREATE_ALWAYS, UInt, 0, UInt, 0)
if not hFile
{
    MsgBox 不能对文件 "%FileName%" 进行写操作
    return
}
TestString = This is a test string.`r`n ; 向新文件中写入文字, 使用 `r`n (不要用 `n ) 来表示
新的一行
DllCall("WriteFile", UInt, hFile, str, TestString, UInt, StrLen(TestString), UIntP,
BytesActuallyWritten, UInt, 0)
DllCall("CloseHandle", UInt, hFile) ; Close the file.
; 现在文件已经成功的写入, 再把它的内容读到内存中来
GENERIC_READ = 0x80000000 ; 以读取方式而不是写入方式来打开文件
OPEN_EXISTING = 3 ; 这个参数表示要打开的文件必须存在
FILE_SHARE_READ = 0x1 ; 这一行以及下一行表示, 允许其他进程在我们打开文件的同时也能够打
开该文件
FILE_SHARE_WRITE = 0x2
hFile := DllCall("CreateFile", str, FileName, UInt, GENERIC_READ, UInt,
FILE_SHARE_READ|FILE_SHARE_WRITE, UInt, 0, UInt, OPEN_EXISTING, UInt, 0, UInt, 0)
if not hFile
{
    MsgBox 不能够读取文件 "%FileName%"
    return
}
; 清空变量以便能够检查是否正确读入, 但要确保该变量的容量可以容纳将要读入的文字
BytesToRead := VarSetCapacity(TestString, StrLen(TestString))
DllCall("ReadFile", UInt, hFile, str, TestString, UInt, BytesToRead, UIntP,
BytesActuallyRead, UInt, 0)
DllCall("CloseHandle", UInt, hFile) ; Close the file.
MsgBox 以下是读取文本的内容: %TestString%

```

```

; 例子: 当按下 Win+C 时隐藏鼠标指针, 再次压下时显示。
; 这个脚本来自 www.autohotkey.com/forum/topic6107.html
OnExit, ShowCursor ; 确保当脚本退出时, 鼠标指针可见。
return
ShowCursor:
SystemCursor("On")
ExitApp
#c::SystemCursor("Toggle") ; Win+C 快捷键切换显示和隐藏

```

```

SystemCursor(OnOff=1) ; 初始化= "I"或"Init"; 隐藏 = 0 或 "Off"; 相反 = -1 或 "T" 或
"Toggle"; 显示 = 其他
{
static AndMask, XorMask, $, h_cursor
,c0,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,c13 ; 系统指针
, b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,b12,b13 ; 空白指针
, h1,h2,h3,h4,h5,h6,h7,h8,h9,h10,h11,h12,h13 ; 默认指针的句柄
if (OnOff = "Init" or OnOff = "I" or $ = "") ; 请求或第一次调用时初始化
{
$ = h ; active default cursors
VarSetCapacity( h_cursor,4444, 1 )
VarSetCapacity( AndMask, 32*4, 0xFF )
VarSetCapacity( XorMask, 32*4, 0 )
system_cursors =
32512,32513,32514,32515,32516,32642,32643,32644,32645,32646,32648,32649,326
50
StringSplit c, system_cursors, ` ,
Loop %c0%
{
h_cursor := DllCall( "LoadCursor", "uint",0, "uint",c%A_Index% )
h%A_Index% := DllCall( "CopyImage", "uint",h_cursor, "uint",2, "int",0, "int",0, "uint",0 )
b%A_Index% := DllCall("CreateCursor","uint",0, "int",0, "int",0
, "int",32, "int",32, "uint",&AndMask, "uint",&XorMask )
}
}
if (OnOff = 0 or OnOff = "Off" or $ = "h" and (OnOff < 0 or OnOff = "Toggle" or OnOff = "T"))
$ = b ; 使用空白指针
else
$ = h ; 使用储存的指针
Loop %c0%
{
h_cursor := DllCall( "CopyImage", "uint",%$%%A_Index%, "uint",2, "int",0, "int",0,
"uint",0 )
DllCall( "SetSystemCursor", "uint",h_cursor, "uint",c%A_Index% )
}
}

```

; 结构体例子：把结构体 RECT 的地址传递给 GetWindowRect() 函数。

; 完成把窗口的左，上，右，下（相对与屏幕的位置）储存到该结构体的成员中。

Run Notepad

WinWait 无标题 - 记事本 ; 也可以设置为 "last found window(See 30.2)" 使用 WinExist() 得到记事本句柄

VarSetCapacity(Rect, 16) ; 一个 RECT 结构体包含了 4 个 32-bit 的整型，(其容量为 4×4=

```
DllCall("GetWindowRect", UInt, WinExist(), UInt, &Rect) ; WinExist() 函数返回一个句柄
MsgBox % "Left " . NumGet(See 10.)(Rect, 0, true) . " Top " . NumGet(Rect, 4, true)
. " Right " . NumGet(Rect, 8, true) . " Bottom " . NumGet(Rect, 12, true)
```

```
; 结构体例子：向函数 FillRect() 传递一个 RECT 结构体，在屏幕上临时画出红色矩形
VarSetCapacity(Rect, 16, 0) ; 设置容量为 4 个 4 字节的整型即 16，并初始化为 0。
NumPut(See 10.)(A_ScreenWidth//2, Rect, 8) ; 第三个整数在结构体中为 "rect.right"
NumPut(A_ScreenHeight//2, Rect, 12) ; 第四个整数在结构体中为 s "rect.bottom".
hDC := DllCall("GetDC", UInt, 0) ; 传入 0 获得桌面的设备上下文
hBrush := DllCall("CreateSolidBrush", UInt, 0x0000FF) ; 创建一个红色画刷 (0x0000FF 是 RGB 格式的红色)
DllCall("FillRect", UInt, hDC, Str, Rect, UInt, hBrush) ; 用上面创建的画刷填充指定的矩形
DllCall("ReleaseDC", UInt, 0, UInt, hDC) ; 释放 DC
DllCall("DeleteObject", UInt, hBrush) ; 释放画刷
```

```
; 结构体例子：改变系统时间为指定的日期，请注意如果时间改为未来的日期，可能使一些软件过早的运行计划的任务。
SetSystemTime("20051008142211") ; 传入一个 timestamp(See 16.26) 参数（本地时间 Local 而非全球时间 UTC）
SetSystemTime(YYYYMMDDHHMISS)
; 设置系统时间为指定日期时间，调用必须确保传入的参数是一个有效的时间格式数据（本地时间而非全球时间）
; 返回值为非零表示调用成功，否则为失败
{
; 把时间参数从 local 转换为 UTC 以便 SetSystemTime() 函数能够调用
UTC_Delta -= %A_NowUTC%, Seconds ; 四舍五入使秒更加精确
UTC_Delta := Round(-UTC_Delta/60) ; 四舍五入使分更加精确
YYYYMMDDHHMISS += %UTC_Delta%, Minutes ; 把时间转换为 UTC 格式
VarSetCapacity(SystemTime, 16, 0) ; 这个结构体是由 8 个 UInts 组成，所以容量为 8×2=
StringLeft, Int, YYYYMMDDHHMISS, 4 ; YYYY (年)
NumPut(See 10.)(Int, SystemTime, 0, 2)
StringMid, Int, YYYYMMDDHHMISS, 5, 2 ; MM (月份, 1-12)
NumPut(Int, SystemTime, 2, 2)
StringMid, Int, YYYYMMDDHHMISS, 7, 2 ; DD (日)
NumPut(Int, SystemTime, 6, 2)
StringMid, Int, YYYYMMDDHHMISS, 9, 2 ; HH (小时 0-23)
NumPut(Int, SystemTime, 8, 2)
StringMid, Int, YYYYMMDDHHMISS, 11, 2 ; MI (分)
NumPut(Int, SystemTime, 10, 2)
StringMid, Int, YYYYMMDDHHMISS, 13, 2 ; SS (秒)
NumPut(Int, SystemTime, 12, 2)
```

```
return DllCall("SetSystemTime", UInt, &SystemTime)
}
```

更多结构体的例子：

- 1) [WinLIRC client script\(See 30.36\)](#) 的范例，介绍了如何使用 DllCall() 来完成 TCP/IP 服务器端和客户端的数据交互和网络链接。
- 2) 在 www.autohotkey.com/forum/topic17230.html 这里介绍了如何利用结构体来使用系统提供的标准对话框，如字体，颜色，图标的选取对话框。

BLoM.2 Fantasy OnLine 08.07.30 翻译

22.9 Edit

使用相关联的编辑器打开当前脚本进行编辑。

Edit

Edit 命令使用注册表中同动词"edit"相关联的编辑器(默认使用记事本)打开当前脚本，并进行编辑。但是，如果已经有一个编辑器窗口打开这个脚本(以窗口标题为准)，这个窗口会被激活，而不是打开一个新的编辑器实例。

Edit 命令相当于托盘图标菜单的"Edit This Script"命令。

当在一个已编译的脚本中执行此命令时将不会产生任何效果。

一个相关要注意的地方是，AutoHotkey 文件夹下的 Extras 文件夹内提供了一些例如 **TextPad** 编辑器的加载项。此外，通过 [这个例子\(See 30.21\)](#)，AutoHotkey 命令的区分上下文的帮助能被用在任何编辑器上。最终，通过使用一个几乎可以在任何编辑器下工作的自动完成工具像 **ISense**，您的效率将会被提高。它监视您的输入并显示菜单来完成输入工作，以及显示参数列表来提示参数的顺序。

相关命令

[Reload\(See 20.1.13\)](#)

示例

```
Edit ; 打开脚本来编辑。
```

翻译：坛友 lwjeee 修正：天堂之门 menk33@163.com 2008 年 8 月 1 日

22.10 ImageSearch

在屏幕某一区域中搜索图像。

ImageSearch, OutputVarX, OutputVarY, X1, Y1, X2, Y2, ImageFile

参数

OutputVarX/Y	变量名称，其可储存屏幕上所发现图像之左上像素之 X 坐标及 Y 坐标。(若未能搜索到图像，则变量为空)。坐标为相对于活动窗口的坐标，除非用 CoordMode(See 22.6) 进行更改。 两参数之一或两者皆可留空，其中可用事件 ErrorLevel (参见下文) 来确定是否搜索到匹配图像。
X1,Y1	欲搜索矩形范围之左上角的 X 坐标及 Y 坐标，其可为 表达式(See 9.) 。坐标是相对于活动窗口的坐标，除非用 CoordMode(See 22.6) 进行更改。
X2,Y2	欲搜索矩形范围之右下角的 X 坐标及 Y 坐标，其可为 表达式(See 9.) 。坐标是相对于活动窗口的坐标，除非用 CoordMode(See 22.6) 进行更改。
ImageFile	<p>图像的文件名，若未指定绝对路径，则假定位于 %A_WorkingDir%(See 9.) 中。所有操作系统均支持 GIF、JPG、BMP、ICO、CUR、及 ANI 图像 (BMP 图像必须为 16-bit 或更高)。其他图标资源包括以下类型的文件：EXE、DLL、CPL、SCR、及包含图标资源的其他类型。在 Windows XP 或更高版本中，可支持诸如 PNG、TIF、Exif、WMF、及 EMF 等其他图像格式。低于 XP 版本的操作系统亦可通过以下方法支持这些图像格式：即，将微软公司的免费 GDI+ DLL 拷贝至 AutoHotkey.exe 文件夹中(但当为编译的脚本(compiled script)(See 8.)时，要将 DLL 拷贝至脚本文件夹中)。如欲下载 DLL，请在 www.microsoft.com 中搜索以下词语： gdi redistributable</p> <p>Options: 亦可将零或更多个以下字符串直接置于文件名之前。每个选项用单个空格或制表符与下一选项分隔开。例如： *2 *w100 *h-1 C:\Main Logo.bmp</p> <p>*IconN: 为使用不是在文件中排第一组的图标组，在 *Icon 后紧接指定的图标组编号。举例而言，*Icon2 将会加载第二图标组的默认图标。</p> <p>*n (变化(variation)): 为 n 指定一个在 0 到 255 之间(包含 255)的数字来表示在组成每个像素颜色的红、绿和蓝的饱和度的每个方向上允许变化的 shades(色调) 的数值。举例而言，*2 将容许 2 色调变化。当该图像之颜色发生轻微变化时，或当 ImageFile 使用一种诸如 GIF 或 JPG 等不能在屏幕上精确呈现图像之格式时，此参数有用。若指定 255 色调变化，则所有颜色皆将匹配。默认值为 0 色调。</p> <p>*TransN: 此选项通过在图像中指定一种可与屏幕上任一颜色相匹配的颜色而促使更易于发现匹配图像。其通常用于寻找具有一些透明区域的 PNG、GIF、及 TIF 文件 (然而，图标不需要此选项，此乃因对其透明性是自动支持的)。对于 GIF 文件，*TransWhite 很可能会起作用。对于 PNG 及 TIF 文件，*TransBlack 可能最好。除此以外，可为 N 指定一些其他颜色名或 RGB 值(请参见 颜色表(color chart)(See 19.12) 中的说明，或以 RGB 模式使用 PixelGetColor(See 22.15))。实例：*TransBlack, *TransFFFAA, *Trans0xFFFFAA</p> <p>*wn 及 *hn: 缩放该图像的宽度及高度(此宽度及高度亦可决定自多图标 .ICO 文件中加载哪个图标)。若这两个选项均省略，则自 ICO、DLL、或 EXE 文件加载的图标会被缩放至系统默认的小图标大小，其通常为 16 X 16(你可通过指定 *w0 *h0 来强制使用实际/固有(actual/internal)大小)。非图标图像以其实际大小加载。为在保持图像纵横比的前提下，可使用 *w0 *h0 来强制使用实际大小。</p>

	提下缩小或放大图像，可将高宽两个尺度之一指定为 -1，而将另一尺度指定为正值。举例而言，指定 *w200 *h-1 将使图像的宽度为 200 像素，并对其高度进行自动设定。
--	--

ErrorLevel

若在指定区域发现了该图像，则将 [ErrorLevel\(See 30.8\)](#) 设定为 0，若未发现，则设为 1，若存在阻止命令执行搜索的问题(例如未能打开该图像文件或格式设置出错的选项)，则设为 2。

注意

ImageSearch 可用于在屏幕上查找缺少文本或其文本不易被检索到的图形目标。例如，其可用于查找图形按钮、图标、网页链接、或游戏目标的位置。一旦定位，则可通过 [Click\(See 23.1\)](#)对这些目标进行点击。

某些时候可采用如下策略：搜索图像的一小块图像剪辑而不搜索整个图像。此方法可在图像发生整体性变化而其内部某些部分保持不变的情况下提高可靠性。一种提取图像剪辑的方法是：

1. 当在活动的窗口中可以看到图像时按 **Alt+PrintScreen**。此操作可将屏幕截图置于剪贴板中。
2. 打开诸如画图板等图像处理程序。
3. 粘贴剪贴板中的内容(即屏幕截图)。
4. 选择对该图像而言唯一且不变的区域。
5. 拷贝并粘贴该区域至新图像文件。
6. 将其保存为随 **ImageSearch** 使用的小文件。

为实现匹配，屏幕上的图像必须与经由 **ImageFile** 参数及其选项所加载的图像具有相同大小。

待搜索区域必须可见；换言之，不可能搜索隐藏在另一窗口后面之窗口的区域。通过比较，通常能够探查到部分位于鼠标光标下面的的图像。例外情况是游戏光标，其在大多数情况下会遮住位于其下面的任何图像。

由于搜索自该区域的顶行开始向下移动，若存在一个以上的匹配图像，则将发现最接近顶部者。

包含透明色的图标会自动容许其颜色匹配屏幕上的任何颜色。因此，位于该图标后面之部分的颜色并不重要。

ImageSearch 可支持 8-bit 颜色的屏幕(256 色)或更高。

搜索行为可随显示适配卡的颜色深度(尤其对于 GIF 及 JPG 文件而言)发生变化。因此，若脚本将在多种颜色深度下运行，则最好在每种色深设置下皆进行测试。您可以使用变化色调选项(**shades-of-variation option (*n)**)来协助搜索行为在多种颜色深度下保持连续性。

若屏幕上的图像是半透明的，则 **ImageSearch** 可能会无法找到它。在此情况下，可尝试使用变化色调选项 (***n**) 或通过 [WinSet, Transparent, Off\(See 28.29\)](#) 使窗口临时不透明。

相关命令

[PixelSearch\(See 22.16\)](#), [PixelGetColor\(See 22.15\)](#), [CoordMode\(See 22.6\)](#), [MouseGetPos\(See 23.5\)](#)

示例

```
ImageSearch, FoundX, FoundY, 40,40, 300, 300, C:\My Images\test.bmp CoordMode  
Pixel ; 将下面坐标表示为相对于屏幕而不是相对于所激活窗口的坐标。
```

```
ImageSearch, FoundX, FoundY, 0, 0, A_ScreenWidth, A_ScreenHeight,  
*Icon3 %A_ProgramFiles%\SomeApp\SomeApp.exe if ErrorLevel = 2 MsgBox 不能实施搜  
索。else if ErrorLevel = 1 MsgBox 未在屏幕上寻得图标。else MsgBox  
在 %FoundX%x%FoundY% 处寻得图标。
```

22.11 ListLines

显示最近已运行过的各脚本行。

[ListLines](#)

注意

该命令与主窗口中"View->Lines most recently executed"菜单项的功能一致。该命令可帮助您 [调试脚本](#)(See 8.)。

做一个相关的提示，内置变量 [A_LineNumber](#)(See 9.) 和 [A_LineFile](#)(See 9.) 分别包含当前运行的行号和其所在的文件名称。

相关命令

[KeyHistory](#)(See 20.9), [ListHotkeys](#)(See 20.1.11), [ListVars](#)(See 22.12)

示例

[ListLines](#)

翻译: lastmore 修正: 天堂之门 menk33@163.com 2008 年 10 月 12 日

22.12 ListVars

列出脚本中的[变量](#)(See 9.): 它们的名称和当前的内容。

[ListVars](#)

注意

此命令等同于选择了主窗口中的 "View->Variables" 菜单项。它可以帮你[调试脚本](#)(See 8.)。

列表的每一行由以下部分组成:

- 1) 变量名。
- 2) 变量内容的长度和变量的[大小](#)(See 18.17)。例如: [50 of 63]
- 3) 变量内容的前 60 个字符。

如果该命令在一个[函数](#)(See 10.)中使用，这个函数的[局部变量](#)(See 10.)将首先列出(先于脚本的全局变量)。

已知的限制：如果一个[函数\(See 10.\)](#)(或者全局变量的列表本身)包含 10,000 个变量以上，此命令可能不会以准确的字母排列顺序来显示它们；也就是说，有些可能会在列表中丢失。

相关命令

[KeyHistory\(See 20.9\)](#), [ListHotkeys\(See 20.1.11\)](#), [ListLines\(See 22.11\)](#)

示例

ListVars

翻译：yugi 修正：天堂之门 menk33@163.com 2008 年 10 月 17 日

22.13 Menu

创建、删除、修改以及显示菜单和菜单项。更改托盘图标和它的提示。控制是否允许打开[已编译脚本\(See 8.\)](#)的主窗口。

Menu, MenuName, Cmd [, P3, P4, P5]

参数

MenuName	它可以是 TRAY 或者任何自定义菜单的名称。自定义菜单的名称在第一次和 Add 命令一起使用时就会自动地创建这个自定义菜单。例如：Menu, MyMenu, Add, Item1 一旦被创建，就可以用 Show 命令来显示自定义菜单。通过 Add 命令，它也可以附着为一个或多个其它菜单的子菜单。
Cmd, P3, P4, P5	这 4 个参数相互依赖。可用的组合如下所示。

Add [, MenuItemName, Label-or-Submenu, Pn]: 此命令有多个用途，如添加菜单项、为其更新一个新的子菜单或标签、或者将普通菜单项转换成子菜单(或者相反)。如果 MenuItemName 还不存在，它将被添加到菜单。否则，MenuItemName 将会被最新指定的 Label-or-Submenu 所更新。

要添加一个菜单分割线，只需省略所有三个参数。

当用户选择菜单项时，将以一个新的[线程\(See 30.19\)](#)运行标签子程序(类似 [Gosub\(See 17.8\)](#) 和[热键子程序\(See 4.\)](#))。如果省略了 Label-or-Submenu，那么 MenuItemName 将同时作为标签和菜单项的名称使用。

要让 MenuItemName 成为子菜单(就是一个菜单项在它被选择时会打开一个新的菜单)，那么给 Label-or-Submenu 指定一个冒号后跟一个已存在的自定义菜单 MenuName 即可。例如：

Menu, MySubmenu, add, Item1

Menu, tray, add, This Menu Item Is A Submenu, :MySubmenu

最后一个参数可以包含字母 P 后跟一个菜单的线程优先级(See 30.19)，比如 P1。如果在添加菜单项时省略此参数，优先级将是标准的默认值 0。如果在更新菜单项时省略，菜单项的优先级将不做改变。要仅仅改现有的菜单项的优先级，可以省略参数 *Label-or-Submenu*。优先级使用十进制数字(不是十六进制)。

Delete [, MenuItemName]: 从菜单删除 *MenuItemName*。标准菜单项比如 Exit (见下文)不能被单独地删除。如果 default 菜单项被删除，其效果如同使用了 NoDefault 选项。如果省略 *MenuItemName*，整个 *MenuName* 菜单将被删除，同时删除在其它菜单中使用 *MenuName* 作为子菜单的任何菜单项。

DeleteAll: 从菜单中删除所有自定义的菜单项，使菜单留空，除非它包含了 standard 菜单项(见下文)。不同于 Delete 命令删除整个菜单(如上文)，空菜单仍然存在，因此将它用作子菜单的任何其它菜单仍将保留这些子菜单。

Rename, MenuItemName [, NewName]: 将 *MenuItemName* 变更为 *NewName* (如果 *NewName* 为空，*MenuItemName* 将转换为分割线)。*NewName* 不能和任何已存在的自定义菜单项同名。菜单项当前的目标标签和子菜单不变。

Check, MenuItemName: 在菜单的 *MenuItemName* 前添加一个可见的对钩标识(如果还没有的话)。

Uncheck, MenuItemName: 从 *MenuItemName* 移除对钩标识(如果有的话)。

ToggleCheck, MenuItemName: 如果没有对钩标识则添加一个；否则，将它移除。

Enable, MenuItemName: 允许用户再次选择 *MenuItemName*，如果它先前被禁用(变灰)的话。

Disable, MenuItemName: 使 *MenuItemName* 变成灰色以表示用户不能选择它。

ToggleEnable, MenuItemName: 如果 *MenuItemName* 之前可用，则禁用它；否则，将启用它。

Default [, MenuItemName]: 将 *MenuItemName* 设为菜单的默认项并加粗它的字体(目前在 TRAY 以外的菜单中设置默认项，仅是改变外观而已)。当用户双击托盘图标时，将开启它的默认菜单项。如果没有默认项，双击将没有任何效果。如果省略 *MenuItemName*，将和下文的 NoDefault 作用相同。

NoDefault: 对于托盘菜单：将菜单改回到它标准菜单的默认项，对未编译脚本来说是 OPEN，而已编译脚本(See 8.)则没有(除非当 MainWindow 选项生效时)。如果之前使用了下面的 NoStandard 命令而导致 OPEN 菜单项不存在的话，那么将没有默认项，因此双击托盘图标也将没有任何效果。对于 TRAY 以外的菜单：任何现有的默认项将变回非加粗字体。

Standard: 在菜单底部插入标准菜单项(如果它们还不存在的话)。此命令可用于托盘菜单或者任何其它菜单。

NoStandard: 从托盘菜单移除所有标准的(非自定义的)菜单项(如果它们存在的话)。

Icon [, FileName, IconNumber, 1]: 将脚本的图标更改为 *FileName* 内图标中的一个。支持下列类型的文件：ICO, CUR, ANI, EXE, DLL, CPL, SCR 以及其它包含图标资源的类型。要使用文件内首个以外的图标组，将它的编号指定给 *IconNumber*(如果省略，那么它默认为 1)。例如，2 将会从第二个图标组加载默认图标。指定星号(*)给 *FileName* 可以将脚本恢复到它的默认图标。

最后一个参数：要冻结图标，可将末尾参数设为 1，要解冻则设为 0(或留空从而保持冻结/解冻状态不做改变)。当图标已被冻结时，[Pause\(See 17.18\)](#) 和 [Suspend\(See 17.23\)](#) 将不能改变图标。注意：要冻结当前的图标，可以像后面例子一样使用 1 或 0：Menu, Tray, Icon,,, 1

修改托盘图标也会改变 [InputBox\(See 19.10\)](#), [Progress\(See 19.12\)](#) 以及随后创建的 [GUI\(See 19.3\)](#) 窗口所显示的图标。已编译脚本([See 8.](#))即使在编译时已经指定过自定义的图标也会受其影响。注意：如果先前使用例如 [#NoTrayIcon\(See 22.1\)](#) 隐藏了托盘图标，那么改变图标是不会反隐藏它的；要这样做的话，可以使用 Menu, Tray, Icon (不带参数)。

从除了 .ICO 外的其他文件类型加载托盘图标时，可能会产生轻微的变形。特别是 16x16 的图标。要避免这种情况，将想要的托盘图标保存在 .ICO 文件里。

操作系统的 DLL 和 CPL 文件包含的一些图标可能会很有用。例如：Menu, Tray, Icon, Shell32.dll, 174 ; 省略 DLL 的路径以便它也能用在 Windows 9x 上。

内置变量 **A_IconNumber** 和 **A_IconFile** 包含了当前图标(如果是默认图标，两者都为空)的序号和名称(带完整的路径)。

Icon (不带参数)：如果不存在托盘图标则创建它。如果脚本里也存在 [#NoTrayIcon\(See 22.1\)](#) 指令的话，此命令将废除它。

NoIcon：如果存在托盘图标则移除它。如果在脚本的最顶部使用此命令，托盘图标可能在装载脚本时暂时地可见。要避免这种情况，可以使用 [#NoTrayIcon\(See 22.1\)](#) 代替。如果当前托盘图标被隐藏，内置变量 **A_IconHidden** 为 1，否则为 0。

Tip [, Text]：改变托盘图标的提示，其在鼠标悬停于图标上面时会显示。要创建一个多行的提示，在每一行中间使用换行符(`n)，例如：Line1`nLine2。将只显示 Text 的前 127 个字符。如果省略 Text，提示将被恢复为默认文本。内置变量 **A_IconTip** 包含了当前提示的文本(如果是默认文本则为空)。

Show [, X, Y]：显示 **MenuName**，允许用户通过方向键、菜单快捷键(加下划线的字母)或者鼠标来选择菜单项。可以显示包括托盘菜单在内的任何菜单，除了 [GUI\(See 19.3\)](#) 菜单栏外。如果省略 X 和 Y，菜单会在当前的鼠标光标位置显示。如果只省略了它们中的一个，那么会用鼠标光标的位置来代替它。X 和 Y 是相对于激活窗口的。事先指定 "[CoordMode\(See 22.6\)](#), Menu" 可以使它们相对于整个屏幕。

Color, ColorValue [, Single]：将菜单的背景色改成 **ColorValue**，它是 16 种 HTML 基础颜色之一或者是 6 位的 RGB 颜色值(请看[颜色表\(See 19.12\)](#))。将 **ColorValue** 留空(或指定为单词 Default)可以将菜单恢复为它的默认颜色。如果没有在下个参数中指定单词 **Single**，那么附在这个菜单上的任何子菜单也将被改变颜色。此命令在 Windows 95/NT 上无效。

Click, ClickCount：将 **ClickCount** 指定为 1 来允许单击激活托盘菜单的默认菜单项。将它指定为 2 可以返回到默认行为(双击)。例如：Menu, Tray, Click, 1

MainWindow：此命令只影响[已编译脚本\(See 8.\)](#)。它允许通过托盘图标来打开脚本的主窗口，反之则打不开。它也能启用主窗口的 View 菜单项，比如 "Lines most recently executed" 就可以查看脚本的源代码和其他信息。**MenuName** 必须是 TRAY。

NoMainWindow (默认): 此命令只影响[已编译脚本](#)(See 8.)。它使脚本恢复为它的默认行为，也就是说它会防止主窗口被打开。即使此选项生效，但在脚本运行时遇上 [ListLines](#)(See 22.11), [ListVars](#)(See 22.12), [ListHotkeys](#)(See 20.1.11) 和 [KeyHistory](#)(See 20.9) 命令时，它们仍可以显示主窗口。
MenuName 必须是 TRAY。

UseErrorLevel [, off]: 如果在脚本中从未使用过此命令，那么它默认为 OFF。每当 Menu 命令遇到错误时，OFF 设置会显示一个对话框并终止[当前线程](#)(See 30.19)。指定 *Menu, Tray, UseErrorLevel* 可以阻止对话框显示和线程终止；取代它们的是，如果遇到问题 [ErrorLevel](#)(See 30.8) 将被设为 1，否则为 0。要将此选项调回 off，可以把下个参数设为 OFF。此设置是全局性的，这意味着它会影响所有的菜单，而不仅仅是菜单 *MenuName*。

要给菜单项的某个字母加下划线，那么在那个字母前加一个 & 符号。当菜单显示时，可以按键盘上相应的按键来选择此菜单项。要显示一个原义的 & 符号，只需指定两个连续的 & 符号，例如：Save && Exit
菜单和菜单项的名称长度最多可达 260 个字符。

当引用已有的菜单或菜单项时，名称不区分大小写，但是必须包含 & 符号。例如：&Open

可以使用 *Menu, tray, add* (即省略其他所有的参数)来给菜单添加分割线。不过，当前还不能单独删除分割线。要绕弯解决这种情况，可使用 *Menu, tray, DeleteAll*，然后重新添加你的自定义菜单项。

新的菜单项总是添加到菜单的底部。对于托盘菜单：要将你的菜单项放在标准菜单项的上面(在添加完你的菜单项后)，可以运行 *Menu, tray, NoStandard* 后跟 *Menu, tray, Standard*。

不能使用任何菜单子命令来单独地操作标准菜单项，比如 "Pause Script" 和 "Suspend Hotkeys"。

如果菜单已完全变空，比如使用了 *Menu, MyMenu, DeleteAll* 命令，则它不会被显示。如果托盘菜单变空，那么右击和双击托盘图标将不起任何作用(这种情况下，通常使用 [#NoTrayIcon](#)(See 22.1) 会更好)。

如果菜单项的子程序运行时，用户再次选择了同个菜单项，那么将创建一个新的[线程](#)(See 30.19)来运行那个相同的子程序，并打断之前的线程。如果想缓存并推迟执行的话，将 [Critical](#)(See 22.7) 作为子程序的首行(不过，这么做同样会缓冲/推迟其它的线程，比如按了一个热键)。

每当通过菜单项启动一个子程序时，都将会刷新设置比如 [SendMode](#)(See 20.13) 的默认值。可以在[自动执行部分](#)(See 8.)修改这些默认值。

内置变量 [A_ThisMenuItem](#)(See 9.) 和 [A_ThisMenuItemPos](#)(See 9.) 分别包含了最近用户选择的自定义菜单项的名称和位置(如果没有则为空)。类似地，[A_ThisMenu](#) 是选择的 [A_ThisMenuItem](#) 的菜单名称。当创建的菜单内容不会一直相同时，这些变量就会很有用。在这种情况下，通常最好给这些菜单项指定相同的标签，然后把那些标签关联到上面的变量，从而确定要执行的动作。

要让一个没有热键、无 [GUI](#)(See 19.3) 的脚本持续运行，比如仅包含了自定义菜单或菜单项的脚本，可以使用 [#Persistent](#)(See 29.21)。

[GUI](#)(See 19.3), [Threads](#)(See 30.19), [Thread](#)(See 22.22), [Critical](#)(See 22.7), [#NoTrayIcon](#)(See 22.1), [Gosub](#)(See 17.8), [Return](#)(See 17.19), [SetTimer](#)(See 17.21), [#Persistent](#)(See 29.21)

;示例 #1: 该脚本在托盘图标菜单底部添加了一个新的菜单项。

```
#Persistent ;保持脚本运行, 直到用户退出它。  
Menu, tray, add ;创建一条分割线。  
Menu, tray, add, Item1, MenuHandler ;创建一个新的菜单项。  
return
```

MenuHandler:

```
MsgBox 你从 %A_ThisMenu% 菜单中选择了 %A_ThisMenuItem% 菜单项。  
return
```

;示例 #2: 该脚本创建了一个弹出菜单, 当用户按热键 Win-Z 将显示它。

```
;通过添加菜单项来创建弹出菜单。  
Menu, MyMenu, Add, Item1, MenuHandler  
Menu, MyMenu, Add, Item2, MenuHandler  
Menu, MyMenu, Add ;添加分割线。
```

;为上述菜单创建子菜单。

```
Menu, Submenu1, Add, Item1, MenuHandler  
Menu, Submenu1, Add, Item2, MenuHandler
```

;为第一个菜单创建子菜单(右箭头指示符)。当用户选择它时将显示第二个菜单。

```
Menu, MyMenu, Add, My Submenu, :Submenu1
```

Menu, MyMenu, Add ;在子菜单下添加分割线。

```
Menu, MyMenu, Add, Item3, MenuHandler ;在子菜单下添加另一个菜单项。
```

```
return ;脚本自动执行段的结尾。
```

```
MenuHandler:
```

```
MsgBox 你从 %A_Menu% 菜单中选择了 %A_MenuItem% 菜单项。
```

```
return
```

```
#z::Menu, MyMenu, Show ;即按下热键 Win-Z 显示菜单。
```

```
;示例 #3: 该脚本演示了多个菜单命令。
```

```
#Persistent  
#SingleInstance  
  
menu, tray, add ;分割线  
  
menu, tray, add, TestToggle&Check  
  
menu, tray, add, TestToggleEnable  
  
menu, tray, add, TestDefault  
  
menu, tray, add, TestStandard  
  
menu, tray, add, TestDelete  
  
menu, tray, add, TestDeleteAll  
  
menu, tray, add, TestRename  
  
menu, tray, add, Test  
  
return
```

```
|||||
```

```
TestToggle&Check:
```

```
menu, tray, ToggleCheck, TestToggle&Check
```

menu, tray, Enable, TestToggleEnable ;它也能启用下面那个测试，因为它不能撤销它自己的禁用状态。

menu, tray, add, TestDelete ;与上一行相似。

return

TestToggleEnable:

menu, tray, ToggleEnable, TestToggleEnable

return

TestDefault:

if default = TestDefault

{

 menu, tray, NoDefault

 default =

}

else

{

 menu, tray, Default, TestDefault

 default = TestDefault

}

return

TestStandard:

if standard <> n

{

 menu, tray, NoStandard

 standard = n

}

else

```
{  
    menu, tray, Standard  
    standard = y  
}  
return  
  
  
TestDelete:  
menu, tray, delete, TestDelete  
return  
  
  
TestDeleteAll:  
menu, tray, DeleteAll  
return  
  
  
TestRename:  
if NewName <> renamed  
{  
    OldName = TestRename  
    NewName = renamed  
}  
else  
{  
    OldName = renamed  
    NewName = TestRename  
}  
menu, tray, rename, %OldName%, %NewName%  
return  
  
  
Test:
```

```
MsgBox, 你在 "%A_ThisMenu%" 菜单选择了 "%A_ThisMenuItem%".
```

```
return
```

翻译: Iwjiee 修正: 天堂之门 2008 年 11 月 20 日

22.14 OutputDebug

发送字符串到调试器(如果有的话)以显示。

OutputDebug, Text

参数

Text	要发送到调试器以显示的文本。此文本可以包含换行符(`n)来启用新的一行。另外，单个较长的行可以通过 连续部分(See 8.) 分解为多个较短的部分。
------	--

注意

如果脚本运行进程中没有设置调试器，那么系统调试器会显示该字符。如果系统调试器未激活，此命令就不起什么作用了。

DebugView 是一个样本调试器，其免费并且可从 www.sysinternals.com 获取。

也可参考: [其他调试方法\(See 8.\)](#)

相关命令

[FileAppend\(See 16.4\)](#), [continuation sections\(See 8.\)](#)

示例

```
OutputDebug, %A_Now%: 由于窗口 "%TargetWindowTitle%" 不存在，进程被终止。
```

翻译: yugi 修正: 天堂之门 menk33@163.com 2008 年 10 月 20 日

22.15 PixelGetColor

Retrieves the color of the pixel at the specified x,y coordinates.

PixelGetColor, OutputVar, X, Y [, Alt|Slow|RGB]

参数

OutputVar	The name of the variable in which to store the color ID in hexadecimal blue-green-red (BGR) format. For example, the color purple is defined 0x800080 because it has an intensity of 80 for its blue and red components but an intensity of 00 for its green component.
-----------	---

X, Y	The X and Y coordinates of the pixel, which can be expressions (See 9.). Coordinates are relative to the active window unless CoordMode (See 22.6) was used to change that.
Alt Slow RGB	<p>This parameter may contain zero or more of the following words. If more than one word is present, separate each from the next with a space (e.g. <i>Alt RGB</i>).</p> <p>Alt [v1.0.43.10+]: Uses an alternate method to retrieve the color, which should be used when the normal method produces invalid or inaccurate colors for a particular type of window. This method is about 10% slower than the normal method.</p> <p>Slow [v1.0.43.10+]: Uses a more elaborate method to retrieve the color, which may work in certain full-screen applications when the other methods fail. This method is about three times slower than the normal method. Note: <i>Slow</i> takes precedence over <i>Alt</i>, so there is no need to specify <i>Alt</i> in this case.</p> <p>RGB: Retrieves the color in RGB vs. BGR format. In other words, the red and the blue components are swapped. This is useful for retrieving colors compatible with WinSet, (See 28.29)Gui(See 19.3), Progress(See 19.12), and SplashImage(See 19.12).</p>

ErrorLevel

[ErrorLevel](#)(See 30.8) is set to 1 if there was a problem or 0 otherwise.

注意

The pixel must be visible; in other words, it is not possible to retrieve the pixel color of a window hidden behind another window. By contrast, pixels beneath the mouse cursor can usually be detected. The exception to this is game cursors, which in most cases will hide any pixels beneath them.

Use Window Spy (available in tray icon menu) or the example at the bottom of this page to determine the colors currently on the screen.

Known limitations:

- A window that is [partially transparent](#)(See 28.29) or that has one of its colors marked invisible ([TransColor](#)(See 28.29)) typically yields colors for the window behind itself rather than its own.
- PixelGetColor might not produce accurate results for certain applications. If this occurs, try specifying the word *Alt* or *Slow* in the last parameter.

相关命令

[PixelSearch](#)(See 22.16), [ImageSearch](#)(See 22.10), [CoordMode](#)(See 22.6), [MouseGetPos](#)(See 23.5)

示例

```
^!z:: ; Control+Alt+Z hotkey.

MouseGetPos, MouseX, MouseY

PixelGetColor, color, %MouseX%, %MouseY%

MsgBox The color at the current cursor position is %color%.

return
```

22.16 PixelSearch

Searches a region of the screen for a pixel of the specified color.

`PixelSearch, OutputVarX, OutputVarY, X1, Y1, X2, Y2, ColorID [, Variation, Fast|RGB]`

参数

OutputVarX/Y	The names of the variables in which to store the X and Y coordinates of the first pixel that matches <i>ColorID</i> (if no match is found, the variables are made blank). Coordinates are relative to the active window unless CoordMode (See 22.6) was used to change that. Either or both of these parameters may be left blank, in which case ErrorLevel (see below) can be used to determine whether a match was found.
X1, Y1	The X and Y coordinates of the upper left corner of the rectangle to search, which can be expressions (See 9.). Coordinates are relative to the active window unless CoordMode(See 22.6) was used to change that.
X2, Y2	The X and Y coordinates of the lower right corner of the rectangle to search, which can be expressions (See 9.). Coordinates are relative to the active window unless CoordMode (See 22.6) was used to change that.
ColorID	The decimal or hexadecimal color ID to search for, in Blue-Green-Red (BGR) format, which can be an expression (See 9.). Color IDs can be determined using Window Spy (accessible from the tray menu) or via PixelGetColor (See 22.15). For example: 0x9d6346
Variation	A number between 0 and 255 (inclusive) to indicate the allowed number of shades of variation in either direction for the intensity of the red, green, and blue components of the color (can be an expression (See 9.)). This parameter is helpful if the color sought is not always exactly the same shade. If you specify 255 shades of variation, all colors will match. The default is 0 shades.

	<p>This parameter may contain the word Fast, RGB, or both (if both are present, separate them with a space; that is, <i>Fast RGB</i>).</p> <p>Fast: Uses a faster searching method that in most cases dramatically reduces the amount of CPU time used by the search. Although color depths as low as 8-bit (256-color) are supported, the fast mode performs much better in 24-bit or 32-bit color. If the screen's color depth is 16-bit or lower, the <i>Variation</i> parameter might behave slightly differently in fast mode than it does in slow mode. Finally, the fast mode searches the screen row by row (top down) instead of column by column. Therefore, it might find a different pixel than that of the slow mode if there is more than one matching pixel.</p> <p>RGB: Causes <i>ColorID</i> to be interpreted as an RGB value instead of BGR. In other words, the red and blue components are swapped.</p>
Fast RGB	

ErrorLevel

[ErrorLevel](#)(See 30.8) is set to 0 if the color was found in the specified region, 1 if it was not found, or 2 if there was a problem that prevented the command from conducting the search.

注意

The region to be searched must be visible; in other words, it is not possible to search a region of a window hidden behind another window. By contrast, pixels beneath the mouse cursor can usually be detected. The exception to this is cursors in games, which in most cases will hide any pixels beneath them.

Slow mode only: By default, the search starts at the upper-left pixel of the region and checks all pixels vertically beneath it for a match. If no match is found there, the search continues to the right, column by column, until it finds a matching pixel. The default left-to-right search order can be inverted by swapping *X1* and *X2* in the parameter list. In other words, if *X1* is greater than *X2*, the search will be conducted from right to left, starting at column *X1*. Similarly, if *Y1* is greater than *Y2*, each column of pixels to be searched starting at the bottom rather than the top. Finally, if the region to be searched is large and the search is repeated with high frequency, it may consume a lot of CPU time. To alleviate this, keep the size of the area to a minimum.

相关命令

[PixelGetColor](#)(See 22.15), [ImageSearch](#)(See 22.10), [CoordMode](#)(See 22.6),
[MouseGetPos](#)(See 23.5)

示例

```
PixelSearch, Px, Py, 200, 200, 300, 300, 0x9d6346, 3, Fast
```

```
if ErrorLevel  
    MsgBox, That color was not found in the specified region.  
  
else  
    MsgBox, A color within 3 shades of variation was found at X%Px% Y%Py%.
```

22.17 Reload

用一个新的脚本实例来替换当前运行的实例。

Reload

这个命令对于频繁改变的脚本很有用。给这个命令指定热键，你可以在编辑器保存改动后轻松地重启脚本。

任何传递给原始脚本的 [command-line parameters\(See 8.\)](#)(命令行参数)不会被传递给新的实例。不要使用 `Reload` 来传递这样的参数。相应地，使用 [Run\(See 24.5\)](#) 命令配合 [A_AhkPath\(See 9.\)](#) 和 [A_ScriptFullPath\(See 9.\)](#) 变量(以及 [A_IsCompiled\(See 9.\)](#) 变量，如果脚本总是用在被编译过的形式)。同样，包含字串 `/restart` 作为第一个参数(也就是说，在可执行程序名字的后面)，这告诉程序执行和 `Reload` 相同的动作。也可参见：[command line switches and syntax\(See 8.\)](#)(命令行开关和语法)。

当脚本重启时，它在它的原来的工作目录(实际上是当它被首次载入时的那个目录)载入。也就是说，[SetWorkingDir\(See 16.33\)](#) 将不会改变被新的实例使用的工作目录。

如果脚本不能被重载 -- 也许是因为它有一个语法错误 -- 原来的脚本实例会继续运行。因此，`Reload` 命令后面应该跟着一旦有失败时你想要采取的任何措施(例如一个 [return\(See 17.19\)](#) 来退出当前子程序)。要想使原来的实例检测到失败，仿照此例：

```
Reload  
  
Sleep 1000 ; 如果成功, reload 将会在 Sleep 期间关闭这个实例, 因此下一行命令将从不执行。  
  
MsgBox, 4,, 脚本不能被重载。你想打开它来编辑吗?  
  
IfMsgBox, Yes, Edit  
  
return
```

相关命令

[Edit\(See 22.9\)](#)

示例

```
^!r::Reload ; 指定 Ctrl-Alt-R 作为重启脚本的热键。
```

翻译：lwjeee 修正：天堂之门 menk33@163.com 2008年8月1日

22.18 SetBatchLines

决定脚本的执行速度（影响 cpu 占用）。

`SetBatchLines, 20ms`
`SetBatchLines, LineCount`

参数

20ms	(20ms 只是举一个例子。) 如果这个参数以 <code>ms</code> 结尾，它表示脚本每两次休眠之间的时 间间隔（每次休眠 10ms）。在下面的例子中，设置脚本每执行 20ms 之后休眠 10ms： <code>SetBatchLines, 20ms</code>
LineCount	两次休眠之间执行脚本的行数。这个参数最大可以到 9223372036854775807。但是， 这个参数和上面那个参数是互相冲突的；也就是说，同时只有一个参数有效。

注意

使用 `SetBatchLines -1` 让脚本无休眠地执行（换句话说，也就是让脚本全速运行）。

如果没有使用这个命令：

- AutoIt v2 (.aut) 的脚本默认使用 `SetBatchLines 1`，每执行一行脚本休眠一次。
- 其它类型的脚本（例如.ahk）默认使用 `SetBatchLines 10ms`。不过在 v1.0.16 之前的版本中，
默认使用的是 `SetBatchLines 10`。

不论是注重脚本执行速度还是注重 cpu 占用，都推荐使用带“ms”的参数。例如，在大多数的系统中，设置 10ms 的休眠间隔可以让脚本占用不超过 50% 的 cpu 资源。这样不但让脚本快速执行，也可以保留充分的 cpu 资源让其它任务使用，例如游戏或者视频捕捉、回放。

内置变量 **A_BatchLines** 保存了当前设置。

在特定的脚本中，脚本的执行速度同时还受这些命令影响：[SetWinDelay\(See 28.9\)](#)，
[SetControlDelay\(See 28.1.13\)](#)，[SendMode\(See 20.13\)](#)，[SetKeyDelay\(See 20.14\)](#)，
[SetMouseDelay\(See 23.8\)](#) 以及 [SetDefaultMouseSpeed\(See 23.7\)](#)。

每一个新运行的 [Thread/线程\(See 30.19\)](#)（例如一个 [hotkey/热键\(See 4.\)](#), [custom menu item/自
定义菜单\(See 22.13\)](#)，或 [timed/定时器\(See 17.21\)](#) 事件）会将该命令的设置重置为默认值。要更改该
命令的默认值，可以将该命令放在脚本的自动执行区域（脚本的顶部）。

相关命令

[SetWinDelay\(See 28.9\)](#), [SetControlDelay\(See 28.1.13\)](#), [SendMode\(See 20.13\)](#),
[SetKeyDelay\(See 20.14\)](#), [SetMouseDelay\(See 23.8\)](#), [SetDefaultMouseSpeed\(See 23.7\)](#),
[Critical\(See 22.7\)](#)

示例

```
SetBatchLines, 10ms
```

```
SetBatchLines, 1000
```

22.19 SetEnv (var=value)

为一个 [变量\(See 9.\)](#) 赋值。

`SetEnv, Var, Value`

`Var = Value`

参数

Var	变量(See 9.) 的名称。变量中存储了指定的 值。
Value	需要存储的值，可以是字符串或数字。如果字符串太长，可以使用 continuation section/字符串分段(See 8.) 的方法将它分为几个短小的段落，这样可以增加代码的可读性和可维护性。

注意

默认情况下，任何在 `值` 的开头和结尾处的空格和 Tab 是被忽略的，不会存入变量中。要避免这种情况，可以在赋值之前使用 [AutoTrim Off \(See 22.3\)](#) 命令。但是，Tab 在任何情况下都是被忽略的。要想保留 Tab，可以使用内置变量 [%A_Tab%](#)(See 9.)。

命令的名字“SetEnv”容易使人产生误解，保留它仅仅是为了兼容 AutoIt v2。和 AutoIt v2 不同，AutoHotkey 并不将变量存储在系统环境中，因为这样性能比较差而且系统限制这样的变量最大为 32K。使用 [EnvSet\(See 15.3\)](#) 命令来设置一个 [环境变量\(See 9.\)](#)，而不是 SetEnv。

可以设置变量为空来释放它所占用的内存。例如 `var =` 。

这个命令，以及所有接收 `OutputVar` 参数的命令都可以创建一个 [数组\(See 30.5\)](#)，只要让 `OutputVar` 包含一个对其它变量的引用就行了。例如 `array%i% = 123`。查看 [数组\(See 30.5\)](#) 获取更多信息。

相关命令

[AutoTrim\(See 22.3\)](#), [EnvSet\(See 15.3\)](#), [EnvAdd\(See 21.1\)](#), [EnvSub\(See 21.4\)](#), [EnvMult\(See 21.3\)](#), [EnvDiv\(See 21.2\)](#), [If\(See 17.10\)](#), [Arrays\(See 30.5\)](#)

示例

```
Var1 = This is a string.  
Color2 = 450  
Color3 = %Var1%  
Array%i% = %A_TICKCOUNT%
```

22.20 SetTimer

以一个指定的时间间隔来自动重复地启动子程序。

`SetTimer, Label [, Period|On|Off, Priority]`

参数

Label	欲跳转的标签或热键标签(See 4.)的名称，它使 <i>Label</i> 下面的命令被执行直到遇上 Return (See 17.19)或 Exit (See 17.6)。与几乎所有其他命令的参数一样， <i>Label</i> 可以是一个变量(See 9.)引用比如 <code>%MyLabel%</code> ，在这种情况下变量中存储的名称会被作为目标使用。
Period On Off	<p>On: 重新启用一个在它的先前 <i>period</i> (周期)中被禁用的定时器。如果不存在定时器，则创建它(默认周期为 250)。如果存在定时器但先前被设置成了 run-only-once 模式，它仍然只运行一次。</p> <p>Off: 禁用一个已存在的定时器。</p> <p>Period: 使用这个参数作为 <i>Label</i> 子程序上一次开始后必须经过的毫秒数来创建或更新一个定时器。在这段时间过后，将再次运行 <i>Label</i> (除非它仍在继续上一次运行)。定时器将被自动启用。要防止这种情况，可以随后立即再一次调用此命令，将此参数指定为 OFF。如果此参数为空并且：</p> <ol style="list-style-type: none"> 1) 定时器不存在：那么将创建一个周期为 250 的定时器。 2) 定时器已存在：那么将启用定时器，并重设它先前的 <i>period</i>，除非指定了 <i>Priority</i>。 <p>Run only once [v1.0.46.16+]: 用一个负的 <i>Period</i> 来表示定时器只能运行一次。例如，指定 -100 将在 100ms 后运行定时器，然后就像使用了 <code>SetTimer, Label, Off</code> 那样禁用定时器。</p>
Priority	这个可选参数是一个介于 -2147483648 和 2147483647 之间的整数 (或一个表达式(See 9.))，用来表示此定时器的线程优先级。如果省略将视为 0。详见 线程 (See 30.19)。要改变一个已有定时器的优先级而不在任何其他方面影响它，只要将此参数前面的那些参数都留空。

注意

定时器很有用，因为它们是异步运行的，这意味着即使当脚本正在等待一个窗口，显示一个对话框，或忙于其他任务时，定时器也将按指定的频率（间隔）运行。它们的许多应用实例包括当用户闲置时 (如 `%A_TimeIdle%`(See 9.) 反映的那样)采取某些行动或者在不需要的窗口出现时关闭它们。

尽管定时器会给人以脚本同时运行多个任务的错觉，但实际并非如此。而是定时的子程序被作为其他线程来处理：它们能中断别的线程或被另一个线程打断，比如[热键子程序](#)(See 4.)。详见[线程](#)(See 30.19)。

无论一个定时器是被创建，重新启用，还是更新为一个新的 *period*，它都不会马上运行；它的 *period* 时间必须先到期。如果你想让定时器的首次执行马上开始，可用 [Gosub](#)(See 17.8) 来执行定时器的子程序 (不过这种方式不会像定时器自身那样开启一个新线程；所以像 [SendMode](#)(See 20.13) 这样的设置就不会以它们的默认值启动了)。

如果 **SetTimer** 被用在一个已存在的定时器上且第二个参数是一个数字或单词 **ON**(或它被省略), 定时器内部的“上一次运行时间”将被重置为当前时间; 换句话说, 必须用完它的整个周期后, 定时器才能再次运行。

目前, 定时器在 **Windows XP/2000/NT** 下不能快于 **10 ms**, **Windows 9x** 下约为 **55 ms**。指定小于该值的 **Period** 通常实际的间隔结果为 **10** 或 **55** (但此策略可能在将来版本中改变, 故无需盲从)。

定时器在下列情况中可能无法按指定频率运行:

1. 其它应用程序使 **CPU** 负担过重。
2. 定时器的子程序本身花费了比它的周期更长的时间来运行, 或有太多相抵触的定时器 (调整 **SetBatchLines**(See 17.20) 可能会有所帮助)。
3. 定时器被另一个线程(See 30.19), 即另一个定时的子程序、热键子程序(See 4.)或者自定义菜单项(See 22.13)打断(这可以通过 **Critical**(See 22.7) 避免)。如果发生了这种情况且打断线程花了很长时间才结束, 那么实际上被打断的定时器在这段时间中将被禁用。不过, 其它的定时器对打断首个定时器的线程(See 30.19)进行中断, 仍可继续运行。
4. 脚本由于 **Critical**(See 22.7) 或 **Thread Interrupt/Priority**(See 22.22) 而不可中断。在此时间内, 定时器不会运行。之后, 当脚本可再次中断时, 延误的定时器会尽快运行一次然后恢复到它的正常运作。

由于定时器靠临时中断脚本的当前活动来运作, 所以它们的子程序应保持简短 (以便它们迅速完成), 无论何时都不适宜有很长的中断。

一个脚本运行期间持续生效的定时器通常应在[自动执行部分\(See 8.\)](#)创建。相比之下, 临时定时器经常会被它的子程序禁用 (请看本页底部的示例)。

每当一个定时的子程序运行时, 它的设置比如 **SendMode**(See 20.13) 都会以默认值重新开始。这些默认值可在[自动执行部分\(See 8.\)](#)修改。

虽然定时器会在脚本挂起(See 17.23)时运作, 但如果当前线程(See 30.19)的 "**Thread NoTimers**(See 22.22)" 正生效或者每当线程被暂停(See 17.18)时, 它们是不会运行的。另外, 在用户浏览脚本的某个菜单时 (如托盘图标菜单或菜单栏) 定时器也不会运作。

如果热键(See 4.)的响应时间很重要 (比如游戏中) 且脚本包含的定时器子程序执行时间会超过 **5 ms**, 那么可以使用如下命令来避免 **15 ms** 的延迟。否则如果热键正好在一个定时器线程处于它的不可中断周期时被按下, 这个延迟就会发生:

Thread(See 22.22), interrupt, 0 ;使所有线程始终可中断。

如果定时器在它的子程序正运行时被禁用, 该子程序会继续运行直到完成。

KeyHistory(See 20.9) 功能会显示存在多少个定时器, 当前启用了多少个。

定时器的周期不可大于 **4294967295** 毫秒(**49.7** 天)。

要让脚本持续运行, 比如那些只包含了定时器的脚本, 请用 **#Persistent**(See 29.21)。

相关命令

Gosub(See 17.8), **Return**(See 17.19), **Threads**(See 30.19), **Thread (command)**(See 22.22), **Critical**(See 22.7), **IsLabel()**(See 10.), **Menu**(See 22.13), **#Persistent**(See 29.21)

示例

;例 1：当不需要的窗口出现时将其关闭：

```
#Persistent  
  
SetTimer, CloseMailWarnings, 250  
  
return  
  
  
CloseMailWarnings:  
  
WinClose, Microsoft Outlook, A timeout occurred while communicating  
  
WinClose, Microsoft Outlook, A connection to the server could not be established  
  
return
```

;例 2：等待一个特定的窗口出现，然后通知用户：

```
#Persistent  
  
SetTimer, Alert1, 500  
  
return  
  
  
Alert1:  
  
IfWinNotExist, Video Conversion, Process Complete  
  
    return  
  
;否则：  
  
SetTimer, Alert1, Off ;即定时器在此处将自己关闭。  
  
SplashTextOn, , , 视频转换已完成。  
  
Sleep, 3000  
  
SplashTextOff  
  
return
```

;例 3：检测热键的单击、双击和三次按击。

;这允许热键可以按你按键的次数来执行不同的操作：

```

#C:::

if winc_presses > 0 ; SetTimer 已经启动，所以我们记录按键。

{
    winc_presses += 1

    return
}

;否则，这是新一系列按键的首次按键。将计数设为 1 并启动定时器：

winc_presses = 1

SetTimer, KeyWinC, 400 ;在 400 毫秒内等待更多的按键。

return


KeyWinC:

SetTimer, KeyWinC, off

if winc_presses = 1 ;该键已按过一次。

{
    Run, m:\ ;打开一个文件夹。
}

else if winc_presses = 2 ;该键已按过两次。

{
    Run, m:\multimedia ;打开一个不同的文件夹。
}

else if winc_presses > 2

{
    MsgBox, 检测到三次或更多次点击。
}

;不论上面哪个动作被触发，将计数复位以备下一系列的按键：

winc_presses = 0

return

```

翻译: bu45gande 修正: 天堂之门 menk33@163.com 2008 年 12 月 18 日

22.21 SysGet

Retrieves screen resolution, multi-monitor info, dimensions of system objects, and other system properties.

SysGet, OutputVar, Sub-command [, Param3]

Parameters

OutputVar	The name of the variable in which to store the result.
Sub-command	See list below.
Param3	This parameter is omitted except where noted below.

Sub-commands

MonitorCount: Retrieves the total number of monitors. Unlike SM_CMONITORS mentioned in the table below, *MonitorCount* includes all monitors, even those not being used as part of the desktop. On Windows 95/NT the count is always 1.

MonitorPrimary: Retrieves the number of the primary monitor, which will be 1 in a single-monitor system. On Windows 95/NT the primary monitor is always 1.

Monitor [, N]: Retrieves the bounding coordinates of monitor number **N** (if **N** is omitted, the primary monitor is used). The information is stored in four variables whose names all start with *OutputVar*. If **N** is too high or there is a problem retrieving the info, the variables are all made blank. For example:

SysGet, Mon2, Monitor, 2

MsgBox, Left: %Mon2Left% -- Top: %Mon2Top% -- Right: %Mon2Right% -- Bottom %Mon2Bottom%.

Within a [function](#)(See 10.), to create a set of variables that is global instead of local, [declare](#)(See 10.) *Mon2* as a global variable prior to using this command (the converse is true for [assume-global](#)(See 10.) functions).

MonitorWorkArea [, N]: Same as the above except the area is reduced to exclude the area occupied by the taskbar and other registered desktop toolbars.

MonitorName [, N]: The operating system's name for monitor number **N** (if **N** is omitted, the primary monitor is used).

(Numeric): Specify for *Sub-command* one of the numbers from the table below to retrieve the corresponding value. The following example would store the number of mouse buttons in a variable named "MouseButtonCount":

SysGet, MouseButtonCount, 43

Commonly Used

Number	Description
80	SM_CMONITORS: Number of display monitors on the desktop (not including

	"non-display pseudo-monitors"). Windows 95/NT: The retrieved value is always 0.
43	SM_CMOUSEBUTTONS: Number of buttons on mouse (0 if no mouse is installed).
16, 17	SM_CXFULLSCREEN, SM_CYFULLSCREEN: Width and height of the client area for a full-screen window on the primary display monitor, in pixels.
61, 62	SM_CXMAXIMIZED, SM_CYMAXIMIZED: Default dimensions, in pixels, of a maximized top-level window on the primary display monitor.
59, 60	SM_CXMAXTRACK, SM_CYMAXTRACK: Default maximum dimensions of a window that has a caption and sizing borders, in pixels. This metric refers to the entire desktop. The user cannot drag the window frame to a size larger than these dimensions.
28, 29	SM_CXMIN, SM_CYMIN: Minimum width and height of a window, in pixels.
57, 58	SM_CXMINIMIZED, SM_CYMINIMIZED: Dimensions of a minimized window, in pixels.
34, 35	SM_CXMINTRACK, SM_CYMINTRACK: Minimum tracking width and height of a window, in pixels. The user cannot drag the window frame to a size smaller than these dimensions. A window can override these values by processing the WM_GETMINMAXINFO message.
0, 1	SM_CXSCREEN, SM_CYSCREEN: Width and height of the screen of the primary display monitor, in pixels. These are the same as the built-in variables A_ScreenWidth (See 9.) and A_ScreenHeight (See 9.).
78, 79	SM_CXVIRTUALSCREEN, SM_CYVIRTUALSCREEN: Width and height of the virtual screen, in pixels. The virtual screen is the bounding rectangle of all display monitors. The SM_XVIRTUALSCREEN, SM_YVIRTUALSCREEN metrics are the coordinates of the top-left corner of the virtual screen. Windows NT, Windows 95: The retrieved value is always 0.
19	SM_MOUSEPRESENT: Nonzero if a mouse is installed; zero otherwise.
75	SM_MOUSEWHEELPRESENT: Nonzero if a mouse with a wheel is installed; zero otherwise. Windows 95: The retrieved value is always 0.
63	SM_NETWORK: Least significant bit is set if a network is present; otherwise, it is cleared. The other bits are reserved for future use.
8193	SM_REMOTECONTROL: This system metric is used in a Terminal Services environment. Its value is nonzero if the current session is remotely controlled; zero otherwise. Windows 2000/NT, Windows Me/98/95: The retrieved value is always 0.
4096	SM_REMOTESESSION: This system metric is used in a Terminal Services environment. If the calling process is associated with a Terminal Services client session, the return value is nonzero. If the calling process is associated with the Terminal Server console session, the return value is zero. The console session is not necessarily the physical console. Windows NT 4.0 SP3 and earlier, Windows

	Me/98/95: The retrieved value is always 0.
70	SM_SHOWSOUNDS: Nonzero if the user requires an application to present information visually in situations where it would otherwise present the information only in audible form; zero otherwise.
8192	SM_SHUTTINGDOWN: Nonzero if the current session is shutting down; zero otherwise. Windows 2000/NT, Windows Me/98/95: The retrieved value is always 0.
23	SM_SWAPBUTTON: Nonzero if the meanings of the left and right mouse buttons are swapped; zero otherwise.
76, 77	SM_XVIRTUALSCREEN, SM_YVIRTUALSCREEN: Coordinates for the left side and the top of the virtual screen. The virtual screen is the bounding rectangle of all display monitors. By contrast, the SM_CXVIRTUALSCREEN, SM_CYVIRTUALSCREEN metrics (further above) are the width and height of the virtual screen. Windows NT, Windows 95: The retrieved value is always 0.

Not Commonly Used

Number	Description
56	SM_ARRANGE: Flags specifying how the system arranged minimized windows. See MSDN for more information.
67	SM_CLEANBOOT: Specifies how the system was started: 0 Normal boot 1 Fail-safe boot 2 Fail-safe with network boot
5, 6	SM_CXBORDER, SM_CYBORDER: Width and height of a window border, in pixels. This is equivalent to the SM_CXEDGE value for windows with the 3-D look.
13, 14	SM_CXCURSOR, SM_CYCURSOR: Width and height of a cursor, in pixels. The system cannot create cursors of other sizes.
36, 37	SM_CXDOUBLECLK, SM_CYDOUBLECLK: Width and height of the rectangle around the location of a first click in a double-click sequence, in pixels. The second click must occur within this rectangle for the system to consider the two clicks a double-click. (The two clicks must also occur within a specified time.)
68, 69	SM_CXDRAG, SM_CYDRAG: Width and height of a rectangle centered on a drag point to allow for limited movement of the mouse pointer before a drag operation begins. These values are in pixels. It allows the user to click and release the mouse button easily without unintentionally starting a drag operation.
45, 46	SM_CXEDGE, SM_CYEDGE: Dimensions of a 3-D border, in pixels. These are the 3-D counterparts of SM_CXBORDER and SM_CYBORDER.

7, 8	SM_CXFIXEDFRAME, SM_CYFIXEDFRAME (synonymous with SM_CXDLGFRAME, SM_CYDLGFRAME): Thickness of the frame around the perimeter of a window that has a caption but is not sizable, in pixels. SM_CXFIXEDFRAME is the height of the horizontal border and SM_CYFIXEDFRAME is the width of the vertical border.
83, 84	SM_CXFOCUSBORDER, SM_CYFOCUSBORDER: Width (in pixels) of the left and right edges and the height of the top and bottom edges of a control's focus rectangle. Windows 2000/NT, Windows Me/98/95: The retrieved value is always 0.
21, 3	SM_CXHSCROLL, SM_CYHSCROLL: Width of the arrow bitmap on a horizontal scroll bar, in pixels; and height of a horizontal scroll bar, in pixels.
10	SM_CXHTHUMB: Width of the thumb box in a horizontal scroll bar, in pixels.
11, 12	SM_CXICON, SM_CYICON: Default width and height of an icon, in pixels.
38, 39	SM_CXICONSPACING, SM_CYICONSPACING: Dimensions of a grid cell for items in large icon view, in pixels. Each item fits into a rectangle of this size when arranged. These values are always greater than or equal to SM_CXICON and SM_CYICON.
71, 72	SM_CXMENUCHECK, SM_CYMENUCHECK: Dimensions of the default menu check-mark bitmap, in pixels.
54, 55	SM_CXMENUSIZE, SM_CYMENUSIZE: Dimensions of menu bar buttons, such as the child window close button used in the multiple document interface, in pixels.
47, 48	SM_CXMINSPACING SM_CYMINSPACING: Dimensions of a grid cell for a minimized window, in pixels. Each minimized window fits into a rectangle this size when arranged. These values are always greater than or equal to SM_CXMINIMIZED and SM_CYMINIMIZED.
30, 31	SM_CXSIZE, SM_CYSIZE: Width and height of a button in a window's caption or title bar, in pixels.
32, 33	SM_CXSIZEFRAME, SM_CYSIZEFRAME: Thickness of the sizing border around the perimeter of a window that can be resized, in pixels. SM_CXSIZEFRAME is the width of the horizontal border, and SM_CYSIZEFRAME is the height of the vertical border. Synonymous with SM_CXFRAME and SM_CYFRAME.
49, 50	SM_CXSMICON, SM_CYSMICON: Recommended dimensions of a small icon, in pixels. Small icons typically appear in window captions and in small icon view.
52, 53	SM_CXSMSIZE SM_CYSMSIZE: Dimensions of small caption buttons, in pixels.
2, 20	SM_CXVSCROLL, SM_CYVSCROLL: Width of a vertical scroll bar, in pixels; and height of the arrow bitmap on a vertical scroll bar, in pixels.
4	SM_CYCAPTION: Height of a caption area, in pixels.
18	SM_CYKANJIWINDOW: For double byte character set versions of the system, this is the height of the Kanji window at the bottom of the screen, in pixels.
15	SM_CYMENU: Height of a single-line menu bar, in pixels.

51	SM_CYSMCAPTION: Height of a small caption, in pixels.
9	SM_CYVTHUMB: Height of the thumb box in a vertical scroll bar, in pixels.
42	SM_DBCSENABLED: Nonzero if User32.dll supports DBCS; zero otherwise. Windows Me/98/95: Nonzero if the double-byte character-set (DBCS) version of User.exe is installed; zero otherwise.
22	SM_DEBUG: Nonzero if the debug version of User.exe is installed; zero otherwise.
82	SM_IMMENABLED: Nonzero if Input Method Manager/Input Method Editor features are enabled; zero otherwise. Windows NT, Windows Me/98/95: The retrieved value is always 0. SM_IMMENABLED indicates whether the system is ready to use a Unicode-based IME on a Unicode application. To ensure that a language-dependent IME works, check SM_DBCSENABLED and the system ANSI code page. Otherwise the ANSI-to-Unicode conversion may not be performed correctly, or some components like fonts or registry setting may not be present.
87	SM_MEDIACENTER: Nonzero if the current operating system is the Windows XP, Media Center Edition, zero if not.
40	SM_MENUDROPALIGNMENT: Nonzero if drop-down menus are right-aligned with the corresponding menu-bar item; zero if the menus are left-aligned.
74	SM_MIDEASTENABLED: Nonzero if the system is enabled for Hebrew and Arabic languages, zero if not.
41	SM_PENWINDOWS: Nonzero if the Microsoft Windows for Pen computing extensions are installed; zero otherwise.
44	SM_SECURE: Nonzero if security is present; zero otherwise.
81	SM_SAMEDISPLAYFORMAT: Nonzero if all the display monitors have the same color format, zero otherwise. Note that two displays can have the same bit depth, but different color formats. For example, the red, green, and blue pixels can be encoded with different numbers of bits, or those bits can be located in different places in a pixel's color value. Windows NT, Windows 95: The retrieved value is always 0.
73	SM_SLOWMACHINE: Always zero on Windows NT/2000/XP or later. Nonzero on Windows 95/98/ME only if the computer has at least one of: a 386 processor, less than 6M of RAM, or a videocard that claims to be slow.
86	SM_TABLETPC: Nonzero if the current operating system is the Windows XP Tablet PC edition, zero if not.

Remarks

The built-in variables [A_ScreenWidth](#)(See 9.) and [A_ScreenHeight](#)(See 9.) contain the dimensions of the primary monitor, in pixels.

Related

[DIICall](#)(See 18.3), [WinGet](#)(See 28.15)

Examples

```
; Example #1:

SysGet, MouseButtonCount, 43
SysGet, VirtualScreenWidth, 78
SysGet, VirtualScreenHeight, 79

; Example #2: This is a working script that displays info about each monitor:

SysGet, MonitorCount, MonitorCount
SysGet, MonitorPrimary, MonitorPrimary
MsgBox, Monitor Count: `t%MonitorCount%`nPrimary Monitor: `t%MonitorPrimary%
Loop, %MonitorCount%
{
    SysGet, MonitorName, MonitorName, %A_Index%
    SysGet, Monitor, Monitor, %A_Index%
    SysGet, MonitorWorkArea, MonitorWorkArea, %A_Index%
    MsgBox, Monitor: `t#%A_Index%`nName: `t%MonitorName%`nLeft: `t%MonitorLeft%
    (%MonitorWorkAreaLeft% work)`nTop: `t%MonitorTop% (%MonitorWorkAreaTop%
    work)`nRight: `t%MonitorRight% (%MonitorWorkAreaRight%
    work)`nBottom: `t%MonitorBottom% (%MonitorWorkAreaBottom% work)
}
```

22.22 Thread

Sets the priority or interruptibility of [threads](#)(See 30.19). It can also temporarily disable all [timers](#)(See 17.21).

```
Thread, NoTimers [, false]
Thread, Priority, n
Thread, Interrupt [, Duration, LineCount]
```

Thread, NoTimers [, false]: Prevents interruptions from any [timers](#)(See 17.21) until the [current thread](#)(See 30.19) either ends, executes "*Thread, NoTimers, false*", or is interrupted by another thread that allows timers (in which case timers can interrupt the interrupting thread until it finishes).

If "*Thread NoTimers*" is not used in the auto-execute section (top part of the script), all threads start off as interruptible by timers (though the settings of "Thread Interrupt" [below] will still apply). By contrast, if the auto-execute section turns on *NoTimers* but never turns it off, every newly launched [thread](#)(See 30.19) (such as a [hotkey](#)(See 4.), [custom menu item](#)(See 22.13), or [timer](#)(See 17.21)) starts off immune to interruptions by timers.

Regardless of the default setting, timers will always operate when the script has no threads (unless [Pause](#)(See 17.18) has been turned on).

"*Thread, NoTimers*" is equivalent to "*Thread, NoTimers, true*". In addition, since the true/false parameter is an [expression](#)(See 9.), true resolves to 1, and false to 0.

Thread, Priority, n: Specify for **n** an integer between -2147483648 and 2147483647 (or an [expression](#)(See 9.)) to indicate the current thread's new priority. This has no effect on other threads. See [Threads](#)(See 30.19) for details.

Due to its ability to buffer events, the command "[Critical](#)"(See 22.7) is generally superior to "Thread Priority".

On a related note, the OS's priority level for the entire script can be changed as in this example: [Process](#)(See 24.4), *Priority,, High*

Thread, Interrupt [, Duration, LineCount]: This command should be used sparingly because most scripts perform more consistently with settings close to the defaults.

By default, every newly launched thread is uninterruptible for a *Duration* of 15 milliseconds or a *LineCount* of 1000 script lines, whichever comes first. This gives the thread a chance to finish rather than being immediately interrupted by another thread that is waiting to launch (such as a buffered [hotkey](#)(See 4.) or a series of [timed subroutines](#)(See 17.21) that are all due to be run).

If either component is 0, each newly launched thread is immediately interruptible. If either component is -1, the thread cannot be interrupted as a result of that component. The maximum for both components is 2147483647.

The Interrupt setting is global, meaning that all subsequent threads will obey it, even if the setting is changed somewhere other than the [auto-execute section](#)(See 8.). However, [interrupted threads](#)(See 30.19) are unaffected because their period of uninterruptibility has already expired. Similarly, the [current thread](#)(See 30.19) is unaffected except if it is uninterruptible at the time the *LineCount* component is changed, in which case the new *LineCount* will be in effect for it.

If a [hotkey](#)(See 4.) is pressed or a [custom menu item](#)(See 22.13) is selected while the [current thread](#)(See 30.19) is uninterruptible, that event will be buffered. In other words, it will launch when the current thread finishes or becomes interruptible, whichever comes first. The exception to this is when the current thread becomes interruptible before it finishes, and it is of

higher [priority](#) than the buffered event; in this case the buffered event is unbuffered and discarded.

Regardless of this setting, a thread will become interruptible the moment it displays a [MsgBox](#)(See 19.11), [InputBox](#)(See 19.10), [FileSelectFile](#)(See 16.23), or [FileSelectFolder](#)(See 16.24) dialog.

Either parameter can be left blank to avoid changing it.

Remarks

Due to its greater flexibility and its ability to buffer events, the command "[Critical](#)"(See 22.7) is generally more useful than "Thread Interrupt" and "Thread Priority".

Related

[Critical](#)(See 22.7), [Threads](#)(See 30.19), [Hotkey](#)(See 20.1.10), [Menu](#)(See 22.13), [SetTimer](#)(See 17.21), [Process](#)(See 24.4)

Example

```
Thread, priority, 1 ; Make priority of current thread slightly above average.  
Thread, interrupt, 0 ; Make each newly launched thread immediately interruptible:  
Thread, interrupt, 50, 2000 ; Make each thread interruptible after 50ms or 2000 lines,  
whichever comes first.
```

22.23 Transform

Performs miscellaneous math functions, bitwise operations, and tasks such as ASCII/Unicode conversion.

`Transform, OutputVar, Cmd, Value1 [, Value2]`

参数

OutputVar	The name of the variable in which to store the result of <i>Cmd</i> . SetFormat (See 21.10) determines whether integers are stored as hexadecimal or decimal.
Cmd, Value1/2	See list below.

`Cmd, Value1, Value2`

The *Cmd*, *Value1* and *Value2* parameters are dependent upon each other and their usage is described below.

Unicode [, String]: Retrieves or stores Unicode text on the clipboard. Note: The entire clipboard may be saved and restored by means of [ClipboardAll](#)(See 30.6), which allows "Transform Unicode" to operate without losing the original contents of the clipboard.

There are two modes of operation as illustrated in the following examples:

`Transform, OutputVar, Unicode ; Retrieves the clipboard's Unicode text as a UTF-8 string.`

`Transform, Clipboard, Unicode, %MyUTF_String% ; Places Unicode text onto the clipboard.`

In the second example above, a literal UTF-8 string may be optionally used in place of `%MyUTF_String%`.

Use a hotkey such as the following to determine the UTF-8 string that corresponds to a given Unicode string:

```
^!u:: ; Control+Alt+U hotkey.
```

`MsgBox Copy some Unicode text onto the clipboard, then return to this window and press OK to continue.`

`Transform, ClipUTF, Unicode`

`Clipboard = Transform, Clipboard, Unicode, %ClipUTF%`r`n`

`MsgBox The clipboard now contains the following line that you can paste into your script. When executed, this line will cause the original Unicode string you copied to be placed onto the clipboard:`n`n%Clipboard%`

`return`

Notes: 1) Windows 95 requires the *Microsoft Layer for Unicode* to support this command (Windows 98/Me/NT4 or later have built-in support); 2) The "[Send {ASC nnnn}](#)"(See 20.12)" command is an alternate way to produce Unicode characters, but it does not work in all applications.

Asc, String: Retrieves the ASCII code (a number between 1 and 255) for the first character in *String*. If *String* is empty, *OutputVar* will also be made empty. For example: *Transform, OutputVar, Asc, %VarContainingString%*. Corresponding function: [Asc\(String\)](#)(See 10.).

Chr, Value1: Retrieves the single character corresponding to the ASCII code indicated by *Value1*. If *Value1* is not between 1 and 255 inclusive, *OutputVar* will be made blank to indicate the problem. For example: *Transform, OutputVar, Chr, 130*. Corresponding function: [Chr\(Number\)](#)(See 10.).

Deref, String: Expands variable references and [escape sequences](#)(See 29.5) contained inside other variables. Any badly formatted variable references will be omitted from the expanded result. The same is true if *OutputVar* is expanded into itself; in other words, any references to *OutputVar* inside *String*'s variables will be omitted from the expansion (note however that *String* itself can be `%OutputVar%`). In the following example, if var1 contains the string "test" and var2 contains the literal string "%var1%", *OutputVar* will be set to the string "test": *Transform, OutputVar, deref, %var2%*. Within a [function](#)(See 10.), each variable in *String*

always resolves to a local variable unless there is no such variable, in which case it resolves to a global variable (or blank if none).

HTML, String: Converts *String* into its HTML equivalent by translating characters whose ASCII values are above 127 to their HTML names (e.g. £ becomes £). In addition, the four characters "&<>" are translated to ""&<>". Finally, each linefeed (`n) is translated to
`n (i.e.
 followed by a linefeed).

Mod, Dividend, Divisor: Retrieves the remainder of *Dividend* divided by *Divisor*. If *Divisor* is zero, *OutputVar* will be made blank. *Dividend* and *Divisor* can both contain a decimal point. If negative, *Divisor* will be treated as positive for the calculation. In the following example, the result is 2: *Transform, OutputVar, mod, 5, 3*. Corresponding function: [Mod\(Dividend, Divisor\)](#)(See 10.).

Pow, Base, Exponent: Retrieves *Base* raised to the power of *Exponent*. Both *Base* and *Exponent* may contain a decimal point. If *Exponent* is negative, *OutputVar* will be formatted as a floating point number even if *Base* and *Exponent* are both integers. A negative *Base* combined with a fractional *Exponent* such as 1.5 is not supported; it will cause *OutputVar* to be made blank. See also: [** operator](#)(See 9.).

Exp, N: Retrieves e (which is approximately 2.71828182845905) raised to the *N*th power. *N* may be negative and may contain a decimal point. Corresponding function: [Exp\(N\)](#)(See 10.).

Sqrt, Value1: Retrieves the square root of *Value1*. If *Value1* is negative, *OutputVar* will be made blank. Corresponding function: [Sqrt\(Number\)](#)(See 10.).

Log, Value1: Retrieves the logarithm (base 10) of *Value1*. If *Value1* is negative, *OutputVar* will be made blank. Corresponding function: [Log\(Number\)](#)(See 10.).

Ln, Value1: Retrieves the natural logarithm (base e) of *Value1*. If *Value1* is negative, *OutputVar* will be made blank. Corresponding function: [Ln\(Number\)](#)(See 10.).

Round, Value1 [, N]: If *N* is omitted, *OutputVar* will be set to *Value1* rounded to the nearest integer. If *N* is positive number, *Value1* will be rounded to *N* decimal places. If *N* is negative, *Value1* will be rounded by *N* digits to the left of the decimal point. For example, -1 rounds to the ones place, -2 rounds to the tens place, and -3 rounds to the hundreds place. Note: Round does not remove trailing zeros when rounding decimal places. For example, 12.333 rounded to one decimal place would become 12.300000. This behavior can be altered by using something like [SetFormat](#)(See 21.10), *float, 0.1* prior to the operation (in fact, [SetFormat](#)(See 21.10) might eliminate the need to use Round in the first place). Corresponding function: [Round\(Number \[, N\]\)](#)(See 10.).

Ceil, Value1: Retrieves *Value1* rounded up to the nearest integer. Corresponding function: [Ceil\(Number\)](#)(See 10.).

Floor, Value1: Retrieves *Value1* rounded down to the nearest integer. Corresponding function: [Floor\(Number\)](#)(See 10.).

Abs, Value1: Retrieves the absolute value of *Value1*, which is computed by removing the leading minus sign (dash) from *Value1* if it has one. Corresponding function: [Abs\(Number\)](#)(See 10.).

Sin, Value1: Retrieves the trigonometric sine of *Value1*. *Value1* must be expressed in radians. Corresponding function: [Sin\(Number\)](#)(See 10.).

Cos, Value1: Retrieves the trigonometric cosine of *Value1*. *Value1* must be expressed in radians. Corresponding function: [Cos\(Number\)](#)(See 10.).

Tan, Value1: Retrieves the trigonometric tangent of *Value1*. *Value1* must be expressed in radians. Corresponding function: [Tan\(Number\)](#)(See 10.).

ASin, Value1: Retrieves the arcsine (the number whose sine is *Value1*) in radians. If *Value1* is less than -1 or greater than 1, *OutputVar* will be made blank. Corresponding function: [ASin\(Number\)](#)(See 10.).

ACos, Value1: Retrieves the arccosine (the number whose cosine is *Value1*) in radians. If *Value1* is less than -1 or greater than 1, *OutputVar* will be made blank. Corresponding function: [ACos\(Number\)](#)(See 10.).

ATan, Value1: Retrieves the arctangent (the number whose tangent is *Value1*) in radians. Corresponding function: [ATan\(Number\)](#)(See 10.).

NOTE: Each of the following bitwise operations has a more concise [bitwise operator](#)(See 9.) for use in expressions.

BitNot, Value1: Stores the bit-inverted version of *Value1* in *OutputVar* (if *Value1* is floating point, it is truncated to an integer prior to the calculation). If *Value1* is between 0 and 4294967295 (0xffffffff), it will be treated as an unsigned 32-bit value. Otherwise, it is treated as a signed 64-bit value. In the following example, the result is 0xfffff0f0 (4294963440): Transform, OutputVar, BitNot, 0xf0f

BitAnd, Value1, Value2: Retrieves the result of the bitwise-AND of *Value1* and *Value2* (floating point values are truncated to integers prior to the calculation). In the following example, the result is 0xff00 (65280): Transform, OutputVar, BitAnd, 0xff0f, 0xffff0

BitOr, Value1, Value2: Retrieves the result of the bitwise-OR of *Value1* and *Value2* (floating point values are truncated to integers prior to the calculation). In the following example, the result is 0xf0f0 (61680): Transform, OutputVar, BitOr, 0xf000, 0x00f0

BitXOr, Value1, Value2: Retrieves the result of the bitwise-EXCLUSIVE-OR of *Value1* and *Value2* (floating point values are truncated to integers prior to the calculation). In the following example, the result is 0xff00 (65280): Transform, OutputVar, BitXOr, 0xf00f, 0x0f0f

BitShiftLeft, Value1, Value2: Retrieves the result of shifting *Value1* to the left by *Value2* bit positions, which is equivalent to multiplying *Value1* by "2 to the *Value2*th power" (floating point values are truncated to integers prior to the calculation). In the following example, the result is 8: Transform, OutputVar, BitShiftLeft, 1, 3

BitShiftRight, Value1, Value2: Retrieves the result of shifting *Value1* to the right by *Value2* bit positions, which is equivalent to dividing *Value1* by "2 to the *Value2*th power", truncating the remainder (floating point values are truncated to integers prior to the calculation). In the following example, the result is 2: Transform, OutputVar, BitShiftRight, 17, 3

注意

Sub-commands that accept numeric parameters can also use [expressions](#)(See 9.) for those parameters.

If either *Value1* or *Value2* is a floating point number, the following *Cmds* will retrieve a floating point number rather than an integer: Mod, Pow, Round, and Abs. The number of decimal places retrieved is determined by [SetFormat](#)(See 21.10).

To convert a radians value to degrees, multiply it by 180/pi (approximately 57.29578). To convert a degrees value to radians, multiply it by pi/180 (approximately 0.01745329252).

The value of pi (approximately 3.141592653589793) is 4 times the arctangent of 1.

相关命令

[SetFormat](#)(See 21.10), [Expressions](#)(See 9.), [EnvMult](#)(See 21.3), [EnvDiv](#)(See 21.2),
[StringLower](#)(See 27.18), [if var is type](#)(See 21.8)

示例

```
Transform, OutputVar, Asc, A ; Get the ASCII code of the letter A.
```

22.24 URLDownloadToFile

从 Internet(国际互联网) 下载一个文件。

UrlDownloadToFile, URL, Filename

参数

URL	要下载的文件的 URL(统一资源定位符)。例如, <code>http://someorg.org</code> 可能取得那个组织的欢迎页面。
Filename	<p>下载到一个文件: 指定要在本地创建的文件名称, 如果绝对路径未指定将假设在 %A_WorkingDir%(See 9.)。任何已经存在的文件将被新文件覆盖。</p> <p>下载到一个变量: 在 www.autohotkey.com/forum/topic10466.html 有一个可调用的函数, 你可以复制并粘贴到你的脚本中。</p>

ErrorLevel

如果遇到一个问题 [ErrorLevel\(See 30.8\)](#) 被设为 1 , 否则是 0 。

注意

即使远程文件不存在下载也可能显示成功。这是因为许多网络服务器发送了一个错误页面代替了缺失的文件。此错误页面会代替 *Filename* 被储存。

必须安装 Internet Explorer 3 或更高的版本来使此功能运作。防火墙或者多个网卡的存在可能导致此功能失败。而且，一些网站可能阻止这样的下载。

缓存:

- 在 v1.0.44.07+ , URL 直接从远程服务器获取(也就是，决不从 Internet Explorer 的缓存取得)。要允许缓存，在 URL 前加 *0 后跟一个空格；例如：*0 <http://someorg.org> 。星号后面的零可以由任何有效的 dwFlags 数值代替；有关详情，请在 www.microsoft.com 以 [InternetOpenUrl](#) 搜索。
- 在比 1.0.44.07 更早的版本中，每当可能时就从缓存获取文件。要避免这种情况，在 URL 结尾指定一个有问题的字串。例如：<http://www.someorg.org/doc.html?fakeParam=> 。注意：如果你频繁地下载同个文件，问题字串应该使其有变化。

代理服务器: 如果已经在 Microsoft Internet Explorer 的设置里设定了一个代理服务器，[UrlDownloadToFile](#) 会使用它来访问 Internet 。

相关命令

[FileRead\(See 16.19\)](#), [FileCopy\(See 16.5\)](#)

示例

```
UrlDownloadToFile, http://www.autohotkey.com/download/CurrentVersion.txt,
C:\AutoHotkey Latest Version.txt
```

```
UrlDownloadToFile, http://someorg.org/archive.zip, C:\SomeOrg's Archive.zip
```

翻译：天堂之门 menk33@163.com 2008 年 8 月 12 日

22.25 VarSetCapacity

Enlarges a variable's holding capacity or frees its memory. Normally, this is necessary only for unusual circumstances such as [DllCall\(See 18.3\)](#).

GrantedCapacity := VarSetCapacity(UnquotedVarName [, RequestedCapacity, FillByte])

参数

GrantedCapacity	The length of string that Var can now hold, which will be greater or equal to <i>RequestedCapacity</i> . If <i>VarName</i> is not a valid variable name (such as a
-----------------	--

	literal string or number), 0 is returned. If the system has insufficient memory to make the change (very rare), an error dialog will be displayed and the current thread (See 30.19) will exit.
UnquotedVarName	The name of the variable (<i>not in quotes</i>). For example: <code>VarSetCapacity(MyVar, 1000)</code> . This can also be a dynamic variable such as <code>Array%i%</code> or a function's ByRef parameter (See 10.).
RequestedCapacity	<p>If omitted, the variable's current capacity will be returned and its contents will not be altered. Otherwise, anything currently in the variable is lost (the variable becomes blank).</p> <p>Specify for <i>RequestedCapacity</i> the length of string that the variable should be able to hold after the adjustment. This length does not include the internal zero terminator. For example, specifying 1 would allow the variable to hold up to one character in addition to its internal terminator. Note: the variable will auto-expand if the script assigns it a larger value later.</p> <p>Since this function is often called simply to ensure the variable has a certain minimum capacity, for performance reasons, it shrinks the variable only when <i>RequestedCapacity</i> is 0. In other words, if the variable's capacity is already greater than <i>RequestedCapacity</i>, it will not be reduced (but the variable will still made blank for consistency).</p> <p>Therefore, to explicitly shrink a variable, first free its memory with <code>VarSetCapacity(Var, 0)</code> and then use <code>VarSetCapacity(Var, NewCapacity)</code> -- or simply let it auto-expand from zero as needed.</p> <p>For performance reasons, freeing a variable whose previous capacity was between 1 and 63 might have no effect because its memory is of a permanent type. In this case, the current capacity will be returned rather than 0.</p> <p>For performance reasons, the memory of a variable whose capacity is under 4096 is not freed by storing an empty string in it (e.g. <code>Var := ""</code>). However, <code>VarSetCapacity(Var, 0)</code> does free it.</p> <p>In v1.0.44.03+, specify -1 for <i>RequestedCapacity</i> to update the variable's internally-stored length to the length of its current contents. This is useful in cases where the variable has been altered indirectly, such as by passing its address(See 9.) via DIIICall()(See 18.3). In this mode, <code>VarSetCapacity()</code> returns the length rather than the capacity.</p>
FillByte	This parameter is normally omitted, in which case the memory of the target variable is not initialized (instead, the variable is simply made blank as described above). Otherwise, specify a number between 0 and 255.

Each byte in the target variable's memory area (its current capacity) is set to that number. Zero is by far the most common value, which is useful in cases where the variable will hold raw binary data such as a [DII Call structure](#)(See 18.3).

注意

In addition to its uses described at [DII Call](#)(See 18.3), this function can also be used to enhance performance when building a string by means of gradual concatenation. This is because multiple automatic resizings can be avoided when you have some idea of what the string's final length will be. In such a case, *RequestedCapacity* need not be accurate: if the capacity is too small, performance is still improved and the variable will begin auto-expanding when the capacity has been exhausted. If the capacity is too large, some of the memory is wasted, but only temporarily because all the memory can be freed after the operation by means of [VarSetCapacity\(Var, 0\)](#) or [Var := ""](#).

[#MaxMem](#)(See 29.15) restricts only the automatic expansion that a variable does on its own. It does not affect [VarSetCapacity](#)(See 18.17).

相关命令

[DII Call](#)(See 18.3), [#MaxMem](#)(See 29.15)

示例

```
VarSetCapacity(MyVar, 10240000) ; ~10 MB

Loop
{
    ...
    MyVar = %MyVar% %StringToConcatenate%
    ...
}
```

23. 鼠标控制

23.1 Click

在指定的坐标处模拟鼠标点击，同样也可以模拟按住鼠标、滚动滚轮、鼠标移动。

这里有一些一般用法的例子（所有逗号都是可省略的）：

Click (无参数)	在鼠标当前位置点击鼠标左键。
Click 44, 55	在坐标 44, 55 处点击鼠标左键（以 coordmode/坐标模式(See 22.6) 的设置为基础）。
Click right 44, 55	和上面一样，不过是点击鼠标右键
Click 2	在鼠标当前位置点击鼠标左键两次（同双击）
Click down	按住鼠标左键不放
Click up right	松开鼠标右键
Click %x% %y%	因为 Click 命令不支持 expressions/表达式(See 9.) ，变量必须用两个百分号包围。

Click 之后可以跟 0 个或多个参数，各个参数之间用至少一个空格、**tab**、“\”或逗号分隔。各个参数之间的顺序不限，除了 *ClickCount*，它必须出现在坐标的后面（如果使用了坐标的话）。

X, Y: 鼠标需要点击或移动到目标位置的横坐标和纵坐标。坐标默认是相对于当前的窗口，除非设置了不同的 [CoordMode/坐标模式\(See 22.6\)](#)。如果省略该参数，则默认为当前鼠标位置。

Button Name: **Left**/左键（默认），**Right**/右键，**Middle**/中键（或取这些单词的首字母），某些鼠标有第三个或第四个键(**X1** 和 **X2**)的话，必须要 Windows2000 以上的版本才支持。注意：和 [MouseClick/鼠标点击\(See 23.3\)](#) 不一样，这里的 **Left**/左键（默认）和 **Right**/右键在所有的系统设置中都是一致的，无论你是否在控制面板中将鼠标左右键进行了对调。

Mouse Wheel（这个参数在 Windows2000 之前的系统上是无效的）：使用 **WheelUp** 或者 **WU** 向上滚动滚轮（远离你），使用 **WheelDown** 或 **WD** 向下滚动滚轮（面向你）。在 1.0.48+ 的版本中，**WheelLeft**(或 **WL**) 或者 **WheelRight**(或 **WR**) 也可以被使用（但是它们在 Windows Vista 以前的操作系统中是无效的）。对于后续的 *ClickCount* 参数，它指定了滚动滚轮的次数。不过有些程序不接受 *ClickCount* 大于 1 的情况，对于这些程序，可以使用一个如下的 [Loop/循环\(See 17.12\)](#) 来解决：

Loop 5

Click WheelUp

ClickCount: 鼠标点击次数（例如：*Click 2 ... Click 100, 200, 2*）。如果省略，默认点击一次。如果使用了坐标参数，*ClickCount* 必须出现在坐标参数之后。使用 0 表示只移动鼠标到指定坐标而不点击（例如：*Click 100, 200, 0*）。

Down 或 **Up**: 这些参数一般是省略的，因为一次鼠标点击过程本身包含了鼠标按下和鼠标松开两个事件。如果需要的话，使用 **Down**（或者字母 **D**）表示点击之后按住鼠标不放，之后再使用 **Up**（或者字母 **U**）来释放鼠标。

Relative: 使用 **Rel** 或者 **Relative**，表示以当前鼠标坐标为坐标原点，也就是说，鼠标会根据设置的坐标参数从当前位置开始往右移 X 像素（如果设置了负值则是往左移），以及往下移 Y 像素（如果设置了负值则是往上移）。

注意

`Click` 一般比 [MouseClick/鼠标点击\(See 23.3\)](#) 更推荐使用，因为它不受控制面板中对鼠标左右键设置的影响。

`Click` 的指令发送模式由 [SendMode/发送模式\(See 20.13\)](#) 来控制。如果需要进行特殊的点击，请使用特殊的 `Send` 命令，可以参看例子：[SendEvent {Click, 100, 200}\(See 20.12\)](#)

如果需要模拟 `Shift +` 点击，或者 `Ctrl +` 点击，使用 [Send {Click}\(See 20.12\)](#) 方法是最简单的。例如：

```
Send +{Click 100, 200} ; Shift + 左键点击
Send ^{Click 100, 200, Right} ; Ctrl + 右键点击
```

和 [Send/发送\(See 20.12\)](#) 不同，`Click` 不会自动释放功能键（`Ctrl`、`Alt`、`Shift`、`Win`）。例如，如果 `Ctrl` 当前被设置为按下状态，直接使用 `Click` 会产生 `Ctrl +` 点击的效果，而 `Send {Click}` 会产生普通点击的效果。

在某些游戏中，[SendPlay 模式\(See 20.13\)](#) 比其它的模式具有更好的兼容性。在某些程序或者游戏中，如果鼠标移动速度过快，它们可能无法识别鼠标的移动，这时候可以使用 [SetDefaultMouseSpeed/设置鼠标默认移动速度\(See 23.7\)](#) 来减慢鼠标的移动速度（不过只能在 [SendEvent 模式\(See 20.13\)](#) 中有效）。

[BlockInput/阻止输入\(See 20.5\)](#) 命令可以防止用户在脚本运行过程中因移动或点击鼠标而造成脚本错误。不过在 [SendInput\(See 20.13\)](#) 和 [SendPlay\(See 20.13\)](#) 模式中一般不需要使用这个命令，因为它们本身就自动推迟了用户对鼠标的操作。

可以使用 [SetMouseDelay/设置鼠标延时\(See 23.8\)](#) 来设置鼠标每次点击的时候按下和松开的时间间隔（在 [SendInput 模式\(See 20.13\)](#) 和模拟滚轮滚动的时候无效）。

相关命令

`Send {Click}(See 20.12)`, [SendMode\(See 20.13\)](#), [CoordMode\(See 22.6\)](#),
[SetDefaultMouseSpeed\(See 23.7\)](#), [SetMouseDelay\(See 23.8\)](#), [MouseClick\(See 23.3\)](#),
[MouseClickDrag\(See 23.4\)](#), [MouseMove\(See 23.6\)](#), [ControlClick\(See 23.2\)](#), [BlockInput\(See 20.5\)](#)

示例

```
Click ; 在鼠标当前位置点击鼠标左键。
```

```
Click 100, 200 ; 在指定的坐标处点击鼠标左键。
```

```
Click 100, 200, 0 ; 移动鼠标而不点击。
```

```
Click 100, 200 right ; 点击鼠标右键。
```

```
Click 2 ; 双击。
```

```
Click down ; 点击鼠标左键并且按住不放。
```

```
Click up right ; 松开鼠标右键。
```

23.2 ControlClick

向指定控件发送一个鼠标点击或鼠标滚轮事件。

`ControlClick [, Control-or-Pos, WinTitle, WinText, WhichButton, ClickCount, Options, ExcludeTitle, ExcludeText]`

参数

	<p>如果此 参数(parameter)为空，目标窗口的最顶端控件将被点击(如果窗口没有控件则点击窗口本身)。如果有参数，则分为以下两种模式。</p> <p>模式 1 (坐标): 被指定的 X 和 Y 坐标相对于目标窗口的左上角。X 坐标必须位于 Y 坐标的前面，并且他们之间至少要有一个空格或者制表符把他们隔开。例如: <code>X55 Y33</code>。如果有一个控件位于这个指定的坐标处，则将发送鼠标事件到该坐标处(控件将被点击)。如果此处没有控件，则目标窗口自身会被发送鼠标事件(也许并不会产生效果，取决于这个窗口)。在这种情况下, <i>Options</i> 参数(parameter)里的 X 和 Y 选项将被忽略。</p> <p>模式 2 (控件的 ClassNN 或 文本): 可以是控件的名称/文本或 ClassNN (控件的 classname(类名)和序号)，它们都可以通过 Window Spy 来确定。当使用名称/文本时，matching behavior (匹配模式)通过 SetTitleMatchMode(See 28.8) 决定。</p> <p>默认情况下，模式 2 优先于模式 1。打比方，有这样的一个控件，它的文本或 ClassNN 是这样的格式: "<code>Xnnn Ynnn</code>"，这种情况下模式 2 将会生效。如果想要使用模式 1(坐标)并且避免这种情况的发生，请在 <i>Options</i> 中指定 Pos 一词。例如: <code>ControlClick, x255 y152, WinTitle,,, Pos</code></p> <p>如果要通过控件的 HWND (句柄)来操作，请在此参数(parameter)上留空并在 <i>WinTitle</i> 上指定 <code>ahk_id %控件句柄%</code> (这样仍可以作用于隐藏的控件即使 DetectHiddenWindows(See 28.5) 设定为 Off)。控件的 HWND 可以通过以下方式来获取: ControlGet Hwnd(See 28.1.4), MouseGetPos(See 23.5), 或者 DIICall(See 18.3)。</p>
WinTitle	目标窗口的标题或标题中的部分文字 (匹配模式由 SetTitleMatchMode (See 28.8) 决定)。如果这个参数和另外 3 个参数(WinText, ExcludeTitle, ExcludeText)都被忽略，默认目标是 上一次匹配窗口 (See 30.2)。如果这个参数使用字母 A，同时省略其它 3 个窗口参数，则以当前激活的窗口作为目标。要用窗口的 class 名进行匹配，使用 <code>ahk_class</code> 精确 class 名 (Window Spy 中可以显示 class 名)。要用窗口的 进程标识符 (PID) (See 24.4)进行匹配，使用 <code>ahk_pid %PID%</code> 变量%。要用 窗口组 (See 28.2.2)，使用 <code>ahk_group</code> 组名 (此时 <i>WinText</i> , <i>ExcludeTitle</i> , 以及 <i>ExcludeText</i> 三个变量要省略)。要用窗口的 唯一 ID (See 28.15) 进行匹配，使用 <code>ahk_id %ID%</code> 变量%。要减小检测范围，使用 多重条件 (See 30.2) ，例如: <code>My File.txt ahk_class Notepad</code>
WinText	如果使用这个参数(parameter)，则它应该是目标窗口中某个文本元素的子字符串 (在 Window Spy 中会显示出窗口中的文本元素)。隐藏文本只有在 DetectHiddenText (See 28.4) 设置为 ON 的时候才能检测到。

WhichButton	<p>要发送的鼠标事件: LEFT(左键), RIGHT(右键), MIDDLE(中键) (或者取这些单词的首字母)。如果忽略或留空, 将点击左键。</p> <p>Windows 2000/XP 及以上: X1 (X 按键 1, 第 4 个鼠标按键) 和 X2 (X 按键 2, 第 5 个鼠标按键) 是被支持的。</p> <p>Windows NT/2000/XP 及以上: WheelUp (or WU) 和 WheelDown (or WD) 是被支持的。在这种情况下, <i>ClickCount</i> 是滚轮滚动的次数。</p> <p>Windows Vista 及以上 [v1.0.48+]: WheelLeft (or WL) 和 WheelRight (or WR) 是被支持的 (它们在旧的操作系统中没有效果)。在这种情况下, <i>ClickCount</i> 是滚轮滚动的次数。</p>
ClickCount	<p>发送鼠标点击事件的次数, 可以使用 expression(表达式)(See 9.)。如果忽略或者留空, 默认次数为 1。</p>
Options	<p>一个选项字母的序列, 包含零个或更多的选项字母。例如: d x50 y25</p> <p>NA [v1.0.45+]: 可能提高可靠性。参考下面的 reliability (可靠性)。</p> <p>D: 按住鼠标按键并不放开(产生一个按下的事件)。如果 D 和 U 两个选项都不存在的话, 则将发送一个完整的点击(按下再放开)。</p> <p>U: 放开鼠标按键(产生一个放开的事件)。如果 D 已经存在则不能同时使用这个选项 (反之亦然)。</p> <p>Pos: 在 <i>Options</i> 的任何位置指定 Pos 一词, 以无条件地使用 X/Y 坐标模式, 就像在上面 <i>Control-or-Pos</i> 参数里描述的那样。</p> <p>Xn: n 指定要点击的 X 坐标, 相对于控件的左上角。如果没有指定, 则点击会被发送到控件的横向中点。</p> <p>Yn: n 指定要点击的 Y 坐标, 相对于控件的左上角。如果没有指定, 则点击会被发送到控件的纵向中点。</p> <p>请对 X 和 Y 选项使用十进制(而非十六进制)数字。</p>
ExcludeTitle	标题含有此参数值的窗口将不被考虑。
ExcludeText	文本含有此参数值的窗口将不被考虑。

ErrorLevel

如果出现问题, [ErrorLevel\(See 30.8\)](#) 值将被设置为 1, 否则为 0。

reliability (可靠性)

提高可靠性 -- 特别是当用户正在使用鼠标的时候进行 `ControlClick` -- 请尝试一下的两点或其一：

1) 在 `ControlClick` 之前使用 [SetControlDelay -1](#)(See 28.1.13)。这样可以避免在点击时鼠标按键被按住，以降低用户对鼠标操作时对脚本运行造成的干扰。

2) 在任何地方指定 `NA` 在第 6 个参数 (*Options*)，如下所示：

```
SetControlDelay -1
ControlClick, Toolbar321, WinTitle,,, NA
```

"`NA`" 可以避免激活目标窗口，也可以避免它的输入处理和脚本的相混淆，而这可以免除物理鼠标移动带来的干扰（通常是在目标窗口没有激活时）。然而，这个方法可能对所有类型的窗口和控件都无效。

注意

不是所有的程序都会响应一个 `ClickCount` 大于 1 的鼠标滚轮事件。对于这些程序，请使用一个循环来发送一个级数大于 1 的滚轮事件，就像这个使其滚动 5 级的例子：

```
Loop, 5
ControlClick, Control, WinTitle, WinText, WheelUp
```

窗口的标题和包含的文本是区分大小写的。隐藏的窗口是不会被检测到的，除非 [DetectHiddenWindows](#)(See 28.5) 被打开(`DetectHiddenWindows On`)。

相关命令

[SetControlDelay](#)(See 28.1.13), [Control](#)(See 28.1.1), [ControlGet](#)(See 28.1.4),
[ControlGetText](#)(See 28.1.7), [ControlMove](#)(See 28.1.8), [ControlGetPos](#)(See 28.1.6),
[ControlFocus](#)(See 28.1.3), [ControlSetText](#)(See 28.1.10), [ControlSend](#)(See 20.6), [Click](#)(See
23.1)

示例

`ControlClick, OK, Some Window Title ;` 点击 OK 按钮。

`ControlClick, x55 y77, WinTitle ;`点击一个设置的坐标。注意 X 和 Y 之间的空隙(至少一个空格或者制表符，没有逗号)。

`; 下面的方法可能会提高可靠性并且减少其他不好的效果：`

`SetControlDelay -1`

`ControlClick, Toolbar321, WinTitle,,, NA x192 y10 ;` 在 `NA` 模式下点击相对于一个指定控件的坐标。

翻译：游否 校对：hsudatalks 2009 年 2 月 27 日

23.3 MouseClick

点击或按住一个鼠标按键，或者滚动鼠标滚轮。注意：一般来说，[Click/点击 命令\(See 23.1\)](#) 更加灵活和易用。

`MouseClick [, WhichButton , X, Y, ClickCount, Speed, D|U, R]`

参数

WhichButton	<p>需要点击的按钮: <code>Left</code>/左键(默认), <code>Right</code>/右键, <code>Middle</code>/中键(或取这些单词的首字母), 某些鼠标有第三个或第四个键(<code>X1</code> 和 <code>X2</code>)的话, 必须要 Windows2000 以上的版本才支持。例如: <code>MouseClick, X1</code>。如果省略该参数则默认为左键。</p> <p><u>滚动鼠标滚轮</u>: 在 Windows NT/2000/XP 或之后 d 版本中, 使用 <code>WheelUp</code> 或者 <code>WU</code> 向上滚动滚轮(远离你), 使用 <code>WheelDown</code> 或 <code>WD</code> 向下滚动滚轮(面向你)。在 1.0.48+ 版本中, 分别使用 <code>WheelLeft</code> (或 <code>WL</code>) 或 <code>WheelRight</code> (或 <code>WR</code>) 来向左或向右滚动滚轮(但是在 Windows Vista 之前的操作系统中没有效果)。<code>ClickCount</code> 参数表示滚动滚轮的次数。</p> <p>如果要自动适应用户在控制面板里面对鼠标左右键的交换, 使用 Click/点击 命令(See 23.1) 来替换。</p>
X, Y	鼠标点击位置的横坐标和纵坐标, 可以使用 expressions/表达式(See 9.) 。坐标默认是相对于当前的窗口, 除非设置了不同的 CoordMode/坐标模式(See 22.6) 。如果省略该参数, 则默认为当前鼠标位置。
ClickCount	鼠标点击次数, 可以使用 expressions/表达式(See 9.) 。如果省略该参数, 则默认点击 1 次。
Speed	<p>鼠标移动的速度, 从 0(最快)到 100(最慢), 可以是 expression/表达式(See 9.)。注意: 设置为 0 的时候, 鼠标会立即移动到目标位置。如果省略, 默认使用 SetDefaultMouseSpeed/设置鼠标默认移动速度(See 23.7) 设置的速度, 如果没有设置, 则使用 2。</p> <p><code>Speed</code> 参数在 SendInput/SendPlay 模式(See 20.13) 中是无效的, 在这两种模式中会直接移动鼠标到目标位置(即使 SetMouseDelay/设置鼠标延时(See 23.8) 中有一种模式适用于 <code>SendPlay</code> 模式)。如果需要显示鼠标移动轨迹——比如在向观众演示的时候——可以使用 SendEvent {Click 100, 200}(See 20.12) 或者 SendMode Event/发送模式事件(See 20.13)。(也可同时设置 BlockInput/阻止输入(See 20.5))。</p>
D U	<p>如果省略该参数, 每一次点击都会由一次“down/按下”事件和一次“up/释放”事件组成。单独使用的话:</p> <p><code>D</code> = 按住鼠标按键不放(也就是说产生了一次“down/按下”事件)。</p> <p><code>U</code> = 释放鼠标按键(也就是说产生了一次“up/释放”事件)。</p>
R	如果这个参数使用了字母 <code>R</code> 的话, 坐标轴参数 <code>X</code> 和 <code>Y</code> 会被当作当前鼠标位置的偏移来处理。也就是说, 鼠标会根据设置的坐标参数从当前位置开始往右移 <code>X</code> 像素(如果设置了负值则是往左移), 以及往下移 <code>Y</code> 像素(如果设置了负值则是往上移)。

注意

这个命令使用的发送模式由 [SendMode/发送模式\(See 20.13\)](#) 来设置。

更推荐使用 [Click/点击 命令\(See 23.1\)](#)，因为：

1. 它自动适应用户在控制面板中对鼠标左右键的交换。
2. 它一般更容易使用。

如果需要模拟 Shift + 点击，或者 Ctrl + 点击，在点击命令之前和之后使用 [Send\(See 20.12\)](#) 命令。

例如：

```
; 范例 #1:
Send, {Control down}
MouseClick, left, 55, 233
Send, {Control up}

; 范例 #2:
Send, {Shift down}
MouseClick, left, 55, 233
Send, {Shift up}
```

在某些游戏中，[SendPlay 模式\(See 20.13\)](#) 比其它的模式具有更好的兼容性。在某些程序或者游戏中，如果鼠标移动速度过快，它们可能无法识别鼠标的移动，这时候可以使用 [SetDefaultMouseSpeed/设置鼠标默认移动速度\(See 23.7\)](#) 来减慢鼠标的移动速度(不过只能在 [SendEvnnet 模式\(See 20.13\)](#) 中有效)。

有些程序不接受 ClickCount 大于 1 的情况，对于这些程序，可以使用一个如下的 [Loop/循环\(See 17.12\)](#) 来解决：

```
Loop 5
    Click WheelUp
```

[BlockInput/阻止输入\(See 20.5\)](#) 命令可以防止用户在脚本运行过程中因移动或点击鼠标而造成脚本错误。不过在 [SendInput\(See 20.13\)](#) 和 [SendPlay\(See 20.13\)](#) 模式中一般不需要使用这个命令，因为它们本身就自动推迟了用户对鼠标的操作。

在每次鼠标按下和鼠标释放之后都会有一个自动的延时(除了 [SendInput 模式\(See 20.13\)](#) 和滚动鼠标滚轮)，使用 [SetMouseDelay/设置鼠标延时\(See 23.8\)](#) 来改变延时的长度。

相关命令

[CoordMode](#)(See 22.6), [SendMode](#)(See 20.13), [SetDefaultMouseSpeed](#)(See 23.7),
[SetMouseDelay](#)(See 23.8), [Click](#)(See 23.1), [MouseClickDrag](#)(See 23.4), [MouseGetPos](#)(See
 23.5), [MouseMove](#)(See 23.6), [ControlClick](#)(See 23.2), [BlockInput](#)(See 20.5)

示例

; 在鼠标当前位置双击:

```
MouseClick, left
```

```
MouseClick, left
```

; 和上面一样:

```
MouseClick, left, , , 2
```

; 移动到指定的坐标位置然后点击右键一次:

```
MouseClick, right, 200, 300
```

; 这里用 2 个热键模拟了滚动鼠标滚轮:

```
#up::MouseClick, WheelUp, , , 2 ; 滚动 2 格
```

```
#down::MouseClick, WheelDown, , , 2
```

23.4 MouseClickDrag

点击并按住指定的鼠标按键，移动鼠标到目标位置，释放鼠标按键。

`MouseClickDrag, WhichButton, X1, Y1, X2, Y2 [, Speed, R]`

参数

WhichButton	<p>需要点击的按钮: <code>Left</code>/左键(默认), <code>Right</code>/右键, <code>Middle</code>/中键(或取这些单词的首字母),某些鼠标有第三个或第四个键的话, 必须要 Windows2000 以上的版本才支持: 使用 <code>X1</code>代表第四个键, <code>X2</code> 代表第五个键。例如: MouseClickDrag, X1, ...</p> <p>如果要自动适应用户在控制面板里面对鼠标左右键的交换, 使用 Click/点击 命令(See 23.1) 来替换。</p>
X1, Y1	<p>鼠标开始拖动位置的横坐标和纵坐标, 可以使用 expressions/表达式(See 9.) (鼠标会在开始拖动前移动到该坐标)。坐标默认是相对于当前的窗口, 除非设置了不同的 CoordMode/坐标模式(See 22.6)。如果省略该参数, 则默认为当前鼠标位置。</p>

X2, Y2	鼠标需要拖动到的目标位置的横坐标和纵坐标，可以使用 expressions/表达式 (See 9.)。坐标默认是相对于当前的窗口，除非设置了不同的 CoordMode/坐标模式 (See 22.6)。
Speed	鼠标移动的速度，从 0 (最快) 到 100 (最慢)，可以是 expression/表达式 (See 9.)。注意：设置为 0 的时候，鼠标会立即移动到目标位置。如果省略，默认使用 SetDefaultMouseSpeed/设置鼠标默认移动速度 (See 23.7) 设置的速度，如果没有设置，则使用 2。 <i>Speed</i> 参数在 SendInput/SendPlay 模式 (See 20.13) 中是无效的，在这两种模式中会直接移动鼠标到目标位置（即使 SetMouseDelay/设置鼠标延时 (See 23.8) 中有一种模式适用于 SendPlay 模式 ）。如果需要显示鼠标移动轨迹——比如在向观众演示的时候——可以使用 SendEvent {Click 100, 200} (See 20.12) 或者 SendMode Event/发送模式事件 (See 20.13)。（也可同时设置 BlockInput/阻止输入 (See 20.5)）。
R	如果这个参数使用了字母 R 的话，坐标轴参数 X1 和 Y1 会被当作当前鼠标位置的偏移来处理。也就是说，鼠标会根据设置的坐标参数从当前位置开始往右移 X1 像素（如果设置了负值则是往左移），以及往下移 Y1 像素（如果设置了负值则是往上移）。 相似的，坐标轴参数 X2 和 Y2 会被当作坐标轴参数 X1 和 Y1 的偏移来处理。例如，下面的例子鼠标会首先从开始位置向右和向下移动 5 像素，然后再从这个位置向右和向下拖动 10 像素： <code>MouseClickDrag, Left, 5, 5, 10, 10, , R</code>

注意

这个命令使用的发送模式由 [SendMode/发送模式](#)(See 20.13) 来设置。

拖动鼠标这个操作同样可以由多条 [Send/发送](#) 命令来完成，这样比较灵活，因为可以指定发送的模式。例如：

```
SendEvent {Click 6, 52, down}{Click 45, 52, up}
```

使用 [Send/发送](#) 命令的另外一个好处是，和 [MouseClickDrag](#) 不同，它可以自动适应用户在控制面板中对鼠标左右键的交换。

在某些游戏中，[SendPlay 模式](#)(See 20.13) 比其它的模式具有更好的兼容性。但是，[SendPlay](#) 模式下的鼠标拖动，在 [RichEdit/富文本](#) 控件中（可能还有其它的），例如 [WordPad](#) 和 [Metapad](#)，可能不能正常工作。

在某些程序或者游戏中，如果鼠标移动速度过快，它们可能无法识别鼠标的移动，这时候可以使用 *speed* 参数或 [SetDefaultMouseSpeed/设置鼠标默认移动速度](#)(See 23.7) 来减慢鼠标的移动速度（不过只能在 [SendEvent 模式](#)(See 20.13) 中有效）。

[BlockInput/阻止输入](#)(See 20.5) 命令可以防止用户在脚本运行过程中因移动或点击鼠标而造成脚本错误。不过在 [SendInput](#)(See 20.13) 和 [SendPlay](#)(See 20.13) 模式中一般不需要使用这个命令，因为它们本身就自动推迟了用户对鼠标的操作。

在每次鼠标按下和鼠标释放之后都会有一个自动的延时（除了 [SendInput 模式](#)(See 20.13)）。这个延时同样会发生在每次鼠标移动之后。使用 [SetMouseDelay/设置鼠标延时](#)(See 23.8) 来改变延时的长度。

相关命令

[CoordMode\(See 22.6\)](#), [SendMode\(See 20.13\)](#), [SetDefaultMouseSpeed\(See 23.7\)](#),
[SetMouseDelay\(See 23.8\)](#), [Click\(See 23.1\)](#), [MouseClick\(See 23.3\)](#), [MouseGetPos\(See 23.5\)](#),
[MouseMove\(See 23.6\)](#), [BlockInput\(See 20.5\)](#)

示例

```
MouseClickDrag, left, 0, 200, 600, 400
```

; 下面的例子打开了画图并且画了一个小房子:

```
Run, mspaint.exe
WinWaitActive, ahk_class MSPaintApp,, 2
if ErrorLevel
    return
MouseClickDrag, L, 150, 250, 150, 150
MouseClickDrag, L, 150, 150, 200, 100
MouseClickDrag, L, 200, 100, 250, 150
MouseClickDrag, L, 250, 150, 150, 150
MouseClickDrag, L, 150, 150, 250, 250
MouseClickDrag, L, 250, 250, 250, 150
MouseClickDrag, L, 250, 150, 150, 250
MouseClickDrag, L, 150, 250, 250, 250
```

23.5 MouseGetPos

返回鼠标的当前位置，以及鼠标当前悬停的窗口和控件（可选）。

`MouseGetPos, [OutputVarX, OutputVarY, OutputVarWin, OutputVarControl, 1|2|3]`

参数

OutputVarX/Y	这两个变量存储了当前鼠标的横坐标和纵坐标。坐标默认相对于当前窗口，需要的话可以使用 CoordMode/坐标模式(See 22.6) 来更改屏幕的坐标相对位置。
OutputVarWin	这个变量存储了当前鼠标悬停的窗口的 unique ID number/唯一 ID(See 28.15) ，

	<p>如果无法检测到窗口，则该变量为空。</p> <p>即使窗口未处于激活状态，仍能获取它的信息。不过隐藏窗口无法被侦测到。</p>
OutputVarControl	<p>这个变量存储了当前鼠标悬停的控件的名称（ClassNN）。如果无法检测到控件，则该变量为空。</p> <p>控件的名称和使用 Window Spy (v1.0.14 版以上) 所获取的控件名称是一样的（可能和老版本的 Window Spy 不一样）。不过和 Window Spy 不同的是，使用 MouseGetPos 检测控件不需要该控件所在的窗口处于激活状态。</p>
1 2 3	<p>如果省略了，默认值是 0。否则，可以按如下所示的功能指定一个值：</p> <p>1: 使用一种更简单的方式来获取 OutputVarControl。这种方式在多文档界面(MDI)的程序（例如 SysEdit 或 TextPad）中可以准确的获取激活或置顶的子窗口信息。但是，它对于其它的情况，例如获取 GroupBox/控件组 中的控件，就不那么准确了。</p> <p>2 [v1.0.43.06+]: 在 OutputVarControl 中存储 control's HWND/窗口句柄(See 28.1.4) 而不是 ClassNN。</p> <p>3 [v1.0.43.06+]: 同时包含 1 和 2 的功能。</p>

注意

如果不需，任何参数都是可以省略的。

相关命令

[CoordMode](#)(See 22.6), [WinGet](#)(See 28.15), [SetDefaultMouseSpeed](#)(See 23.7), [Click](#)(See 23.1)

示例

```
MouseGetPos, xpos, ypos
Msgbox, The cursor is at X%xpos% Y%ypos%.

; 这个例子允许你通过移动鼠标来查看
; 鼠标悬停的窗口的信息
#Persistent

SetTimer, WatchCursor, 100

return
```

```

WatchCursor:

MouseGetPos, , , id, control

WinGetTitle, title, ahk_id %id%
WinGetClass, class, ahk_id %id%
ToolTip, ahk_id %id%`nahk_class %class%`n%title%`nControl: %control%

return

```

23.6 MouseMove

移动鼠标。

`MouseMove, X, Y [, Speed, R]`

参数

X, Y	鼠标需要移动到的目标位置的横坐标和纵坐标，可以使用 expressions/表达式 (See 9.)。坐标默认相对于当前激活窗口，可以通过设置 CoordMode/坐标模式 (See 22.6) 来修改。
Speed	鼠标移动的速度，从 0 (最快) 到 100 (最慢)，可以是 expression/表达式 (See 9.)。注意：设置为 0 的时候，鼠标会立即移动到目标位置。如果省略，默认使用 SetDefaultMouseSpeed/设置鼠标默认移动速度 (See 23.7) 设置的速度，如果没有设置，则使用 2。 <code>Speed</code> 参数在 SendInput/SendPlay 模式 (See 20.13) 中是无效的，在这两种模式中会直接移动鼠标到目标位置（即使 SetMouseDelay/设置鼠标延时 (See 23.8) 中有一种模式适用于 SendPlay 模式 ）。如果需要显示鼠标移动轨迹——比如在向观众演示的时候——可以使用 SendEvent {Click 100, 200} (See 20.12) 或者 SendMode Event/发送模式事件 (See 20.13)。（也可同时设置 BlockInput/阻止输入 (See 20.5)）。
R	如果使用了这个参数使用了字母 R 的话，坐标轴参数 X 和 Y 会被当作当前鼠标位置的偏移来处理。也就是说，鼠标会根据设置的坐标参数从当前位置开始往右移 X 像素（如果设置了负值则是往左移），以及往下移 Y 像素（如果设置了负值则是往上移）。

注意

这个命令使用的发送模式由 [SendMode/发送模式](#)(See 20.13) 来设置。

在某些游戏中，[SendPlay 模式](#)(See 20.13) 比其它的模式具有更好的兼容性。在某些程序或者游戏中，如果鼠标移动速度过快，它们可能无法识别鼠标的移动，这时候可以使用 [SetDefaultMouseSpeed/设置鼠标默认移动速度](#)(See 23.7) 来减慢鼠标的移动速度（不过只能在 [SendEvent 模式](#)(See 20.13) 中有效）。

[BlockInput/阻止输入](#)(See 20.5) 命令可以防止用户在脚本运行过程中因移动或点击鼠标而造成脚本错误。不过在 [SendInput/SendPlay 模式](#)(See 20.13) 中一般不需要使用这个命令，因为它们本身就自动推迟了用户对鼠标的操作。

每个鼠标移动事件后都会有一个自动的延时（除了 [SendInput 模式](#)(See 20.13)），使用 [SetMouseDelay/设置鼠标延时](#)(See 23.8) 来改变延时的长度。

这是另外一种移动鼠标的方式，它在一些多显示器环境下会运行得更好：

`DllCall("SetCursorPos", int, 100, int, 400)` ; 第一个数值是横坐标，第二个数值是纵坐标（相对于屏幕）。

顺便说一下，这里有一个暂时隐藏鼠标的例子 [hide-cursor example](#)(See 18.3)。

相关命令

[CoordMode](#)(See 22.6), [SendMode](#)(See 20.13), [SetDefaultMouseSpeed](#)(See 23.7),
[SetMouseDelay](#)(See 23.8), [Click](#)(See 23.1), [MouseClick](#)(See 23.3), [MouseClickDrag](#)(See 23.4),
[MouseGetPos](#)(See 23.5), [BlockInput](#)(See 20.5)

示例

```
; 移动鼠标到一个新位置
MouseMove, 200, 100

; 缓慢的移动鼠标（速度 50），从当前位置往右 20 像素，往下 30 像素
MouseMove, 20, 30, 50, r
```

23.7 SetDefaultMouseSpeed

设置鼠标默认速度，如果下列命令没有指定鼠标速度的话，则使用默认速度：[Click](#)(See 23.1) 以及 [MouseMove](#)(See 23.6)/[MouseClick](#)(See 23.3)/[MouseClickDrag](#)(See 23.4)。

`SetDefaultMouseSpeed, Speed`

参数

Speed	鼠标移动的速度，从 0（最快）到 100（最慢），可以是 expression/表达式 (See 9.)。 注意：设置为 0 的时候，鼠标会立即移动到目标位置。
-------	---

注意

`SetDefaultMouseSpeed` 在 [SendInput/SendPlay 模式](#)(See 20.13) 中是无效的，在这两种模式中会直接移动鼠标到目标位置（不过 [SetMouseDelay 命令](#)(See 23.8) 中有一个参数可以设置 [SendPlay](#) 模式中的延时）。如果需要显示鼠标移动轨迹——比如在向观众演示的时候——可以使用 [SendEvent {Click}](#)

[100, 200\)](#)(See 20.12) 或者 [SendMode 事件](#)(See 20.13)。 (也可同时设置 [BlockInput/阻止输入](#)(See 20.5))。

如果没有使用这个命令， 默认鼠标速度是 2。内置变量 **A_DefaultMouseSpeed** 保存了当前设置。

[MouseClick](#)(See 23.3) , [MouseMove](#)(See 23.6) , 以及 [MouseClickDrag](#)(See 23.4) 都有一个参数来设定自己的鼠标速度。

不论 *Speed* 参数是否大于 0 , [SetMouseDelay](#)(See 23.8) 命令设置的延时都会对鼠标速度产生影响，鼠标每移动一点就会延时一次。

每一个新运行的 [Thread/线程](#)(See 30.19) (例如一个 [hotkey/热键](#)(See 4.) , [custom menu item/自定义菜单](#)(See 22.13), 或 [timed/定时器](#)(See 17.21) 事件) 会将该命令的设置重置为默认值。要更改该命令的默认值，可以将该命令放在脚本的自动执行区域（脚本的顶部）。

相关命令

[SetMouseDelay](#)(See 23.8), [SendMode](#)(See 20.13), [Click](#)(See 23.1), [MouseClick](#)(See 23.3), [MouseMove](#)(See 23.6), [MouseClickDrag](#)(See 23.4), [SetWinDelay](#)(See 28.9), [SetControlDelay](#)(See 28.1.13), [SetKeyDelay](#)(See 20.14), [SetKeyDelay](#)(See 23.8)

示例

```
SetDefaultMouseSpeed, 0 ; 立即移动鼠标。
```

23.8 SetMouseDelay

设置两次鼠标操作类命令间的延时。

`SetMouseDelay, Delay [, Play]`

参数

Delay	延时时间，单位毫秒。可以是一个 expression/表达式 (See 9.)。使用 -1 表示无延时，使用 0 表示最小延时（但是，如果使用了 <i>Play</i> 参数的话，0 和 -1 都表示无延时）。如果没有设置，在 SendEvent 模式中默认是 10，在 SendPlay 模式 (See 20.12) 中默认是 -1。
Play [v1.0.43+]	这个参数使用单词 <i>Play</i> 的话，表示这是为 SendPlay 模式 (See 20.12) 设置的延时，而不是一般的 Send/SendEvent 模式。如果脚本中从未使用过这个参数，则 SendPlay 模式的延时永远是 -1。

注意

脚本在执行了每个鼠标操作类命令之后，会有一个自动的延时（休眠）。这些命令包括：[Click](#)(See 23.1) 以及 [MouseMove](#)(See 23.6)/[MouseClick](#)(See 23.3)/[MouseClickDrag](#)(See 23.4) (except for [SendInput mode](#)(See 20.13))。这样做的目的是提高脚本的可靠性，因为一个窗口一般无法响应太过频繁的鼠标事件。

根据系统的时间精度，设置的延时可能会被四舍五入为最近的整十数字。例如在 XP 中，1 到 10 之间的延时（包括 10）都等效于 10（NT 和 2000 中也是这样）。

设置延时为 0 的话相当于执行了命令 `Sleep(0)`，它会将当前脚本的剩余时间片分配给有需要的进程。如果没有进程需要，延时 0 就相当于完全没有延时。

内置变量 **A_MouseDelay** 保存了当前对 `Send/SendEvent` 模式的设置（没有内置变量保存 `SendPlay` 模式(See 20.12) 的设置）。

每一个新运行的 `Thread/线程(See 30.19)`（例如一个 `hotkey/热键(See 4.)`, `custom menu item/自定义菜单(See 22.13)`, 或 `timed/定时器(See 17.21)` 事件）会将该命令的设置重置为默认值。要更改该命令的默认值，可以将该命令放在脚本的自动执行区域（脚本的顶部）。

相关命令

`SetDefaultMouseSpeed(See 23.7)`, `Click(See 23.1)`, `MouseMove(See 23.6)`, `MouseClick(See 23.3)`, `MouseClickDrag(See 23.4)`, `SendMode(See 20.13)`, `SetKeyDelay(See 20.14)`, `SetControlDelay(See 28.1.13)`, `SetWinDelay(See 28.9)`, `SetBatchLines(See 17.20)`

示例

```
SetMouseDelay, 0
```

24. 进程管理

24.1 Exit

退出 `current thread(See 30.19)`(当前线程) 或(如果脚本非 `persistent(See 29.21)` 且未包含热键) 整个脚本。

`Exit [, ExitCode]`

参数

ExitCode	一个 <code>integer(整数)</code> (也就是负数、正数、零或者 <code>expression(See 9.)</code> (表达式)) 在脚本退出时被返回给它的调用者。此退出代码可被任何调用脚本的程序使用，例如另一个脚本(通过 <code>RunWait</code>) 或一个 <code>batch (.bat) file</code> (批处理文件)。如果省略， <code>ExitCode</code> 默认为 0。0 通常被用来表示成功。注意：Windows 95 可能限制 <code>ExitCode</code> 的大小。
----------	--

注意

如果脚本没有热键，也不是 `persistent(See 29.21)`，并且没有要求 `Num/Scroll/CapsLock` 键保持一直开启或一直关闭，那么当遇到 `Exit` 时它会立即终止(除非它有一个 `OnExit(See 17.17)` 子程序)。

否则，`Exit` 命令只终止 `current thread(See 30.19)`(当前线程)。换句话说，通过 `menu(See 22.13)`、`timer(See 17.21)` 或 `热键(See 4.)` 子程序来直接或间接地调用 `stack of subroutines` (子程序堆栈)，将

好像各自直接地遇上一个 [Return\(See 17.19\)](#) 那样全部被返回。如果直接在这样的子程序内部使用 `Exit --` 而不是在子程序内间接地被脚本调用 `-- Exit` 等同于 [Return\(See 17.19\)](#)。

使用 [ExitApp\(See 17.7\)](#) 来完全终止一个 [persistent\(See 29.21\)](#) 或包含热键的脚本。

相关命令

[ExitApp\(See 17.7\)](#), [OnExit\(See 17.17\)](#), [Functions\(See 10.\)](#), [Gosub\(See 17.8\)](#), [Return\(See 17.19\)](#), [Threads\(See 30.19\)](#), [#Persistent\(See 29.21\)](#)

示例

```
#z::  
Gosub, Sub2  
MsgBox, 由于 EXIT, 这个消息框将不会弹出。  
return  
  
Sub2:  
Exit ; 终止子程序, 也终止了来调用的热键子程序。
```

翻译: Xiangee 修正: 天堂之门 menk33@163.com 2008 年 8 月 8 日

24.2 ExitApp

无条件地终止脚本。

`ExitApp [, ExitCode]`

参数

ExitCode	当脚本退出时返回一个整数(相当于负数、正数或者是零)给它的调用程序。此代码能被任何生成脚本的程序运用, 例如另一个脚本(通过 <code>RunWait</code>)或者一个批处理(<code>.bat</code>)文件。如果省略, <code>ExitCode</code> 默认为零。零通常用来表示成功。注意: Windows 95 可能限制了 <code>ExitCode</code> 的大小。
----------	--

说明

脚本将立即被终止, 除非它有一个 [OnExit\(See 17.17\)](#) 子程序。这相当于从脚本的托盘菜单或主菜单选择 "Exit"。

当脚本没有包含热键, 不是[持续的\(See 29.21\)](#), 并且没有要求 `Num/Scroll/Capslock` 键保持一直打开或者一直关闭时, [Exit\(See 17.6\)](#) 和 `ExitApp` 作用相同。

如果脚本有一个 [OnExit\(See 17.17\)](#) 子程序, 它将响应 `ExitApp` 而运行。

相关命令

[Exit](#)(See 17.6), [OnExit](#)(See 17.17), [#Persistent](#)(See 29.21)

范例

```
#x::ExitApp ;指定一个热键来终止此脚本。
```

翻译：天堂之门 menk33@163.com 2008年5月17日

24.3 OnExit

Specifies a [subroutine](#)(See 17.8) to run automatically when the script exits.

`OnExit [, Label]`

参数

Label	If omitted, the script is returned to its normal exit behavior. Otherwise, specify the name of the label whose contents will be executed (as a new thread (See 30.19)) when the script exits by any means.
-------	--

IMPORTANT: Since the specified subroutine is called instead of terminating the script, that subroutine must use the [ExitApp](#)(See 17.7) command if termination is desired. Alternatively (as in the Examples section below), the OnExit subroutine might display a [MsgBox](#)(See 19.11) to prompt the user for confirmation -- and only if the user presses YES would the script execute ExitApp to close itself.

注意

The OnExit subroutine is called when the script exits by any means (except when it is killed by something like "End Task"). It is also called whenever the [#SingleInstance](#)(See 22.2) and [Reload](#)(See 20.1.13) commands ask a previous instance to terminate.

A script can detect and optionally abort a system shutdown or logoff via [OnMessage\(0x11, "WM_QUERYENDSESSION"\)](#)(See 18.11).

The OnExit [thread](#)(See 30.19) does not obey [#MaxThreads](#)(See 20.1.6) (it will always launch when needed). In addition, while it is running, it cannot be interrupted by any [thread](#)(See 30.19), including [hotkeys](#)(See 4.), [custom menu items](#)(See 22.13), and [timed subroutines](#)(See 17.21). However, it will be interrupted (and the script will terminate) if the user chooses Exit from the tray menu or main menu, or the script is asked to terminate as a result of [Reload](#)(See 20.1.13) or [#SingleInstance](#)(See 22.2). Because of this, the OnExit subroutine should be designed to finish quickly unless the user is aware of what it is doing.

If the OnExit [thread](#)(See 30.19) encounters a failure condition such as a runtime error, the script will terminate. This prevents a flawed OnExit subroutine from making a script impossible to terminate.

If the OnExit subroutine was launched due to an [Exit](#)(See 17.6) or [ExitApp](#)(See 17.7) command that specified an exit code, that code is ignored and no longer available. A new exit code can be specified by the OnExit subroutine if/when it calls [ExitApp](#)(See 17.7).

Whenever the OnExit subroutine is called by an exit attempt, it starts off fresh with the default values for settings such as [SendMode](#)(See 20.13). These defaults can be changed in the [auto-execute section](#)(See 8.).

The built-in variable **A_ExitReason** is blank unless the OnExit subroutine is currently running or has been called at least once by a prior exit attempt. If not blank, it is one of the following words:

Logoff	The user is logging off.
Shutdown	The system is being shut down or restarted, such as by the Shutdown (See 24.7) command.
Close	The script was sent a WM_CLOSE or WM_QUIT message, had a critical error, or is being closed in some other way. Although all of these are unusual, WM_CLOSE might be caused by WinClose (See 28.14) having been used on the script's main window. To prevent this, dismiss the main window with <i>Send</i> , !{F4}.
Error	A runtime error occurred in a script that has no hotkeys and that is not persistent (See 29.21). An example of a runtime error is Run/RunWait (See 24.5) being unable to launch the specified program or document.
Menu	The user selected Exit from the main window's menu or from the standard tray menu.
Exit	The Exit (See 17.6) or ExitApp (See 17.7) command was used (includes custom menu items (See 22.13)).
Reload	The script is being reloaded via the Reload (See 20.1.13) command or menu item.
Single	The script is being replaced by a new instance of itself as a result of #SingleInstance (See 22.2).

相关命令

[OnMessage\(\)](#)(See 18.11), [RegisterCallback\(\)](#)(See 18.14), [OnClipboardChange](#)(See 30.6), [ExitApp](#)(See 17.7), [Shutdown](#)(See 24.7), [#Persistent](#)(See 29.21), [Threads](#)(See 30.19), [Gosub](#)(See 17.8), [Return](#)(See 17.19), [Menu](#)(See 22.13)

示例

```
#Persistent ; For demonstration purposes, keep the script running if the user chooses "No".
OnExit, ExitSub
return
```

```

ExitSub:

if A_ExitReason not in Logoff,Shutdown ; Avoid spaces around the comma in this line.

{

    MsgBox, 4, , Are you sure you want to exit?

    IfMsgBox, No

        return

}

ExitApp ; The only way for an OnExit script to terminate itself is to use ExitApp in the
OnExit subroutine.

```

24.4 Process

Performs one of the following operations on a process: checks if it exists; changes its priority; closes it; waits for it to close.

Process, Cmd, PID-or-Name [, Param3]

参数

	<p>One of the following words:</p> <p>Exist: Sets ErrorLevel(See 30.8) to the Process ID (PID) if a matching process exists, or 0 otherwise. If the <i>PID-or-Name</i> parameter is blank, the script's own PID is retrieved. An alternate, single-line method to retrieve the script's PID is <i>PID := DllCall("GetCurrentProcessId")</i></p> <p>Close: If a matching process is successfully terminated, ErrorLevel(See 30.8) is set to its former Process ID (PID). Otherwise (there was no matching process or there was a problem terminating it), it is set to 0. Since the process will be abruptly terminated -- possibly interrupting its work at a critical point or resulting in the loss of unsaved data in its windows (if it has any) -- this method should be used only if a process cannot be closed by using WinClose(See 28.14) on one of its windows.</p> <p>List: Although <i>List</i> is not yet supported, the examples section demonstrates how to retrieve a list of processes via DllCall.</p> <p>Priority: Changes the priority (as seen in Windows Task Manager) of the first matching process to <i>Param3</i> and sets ErrorLevel(See 30.8) to its Process ID (PID). If the <i>PID-or-Name</i> parameter is blank, the script's own priority will be changed. If there is no matching process or there was a problem changing its</p>
Cmd	

	<p>priority, ErrorLevel is set to 0.</p> <p><i>Param3</i> should be one of the following letters or words: L (or Low), B (or BelowNormal), N (or Normal), A (or AboveNormal), H (or High), R (or Realtime). Since BelowNormal and AboveNormal are not supported on Windows 95/98/Me/NT4, <i>normal</i> will be automatically substituted for them on those operating systems. Note: Any process not designed to run at Realtime priority might reduce system stability if set to that level.</p> <p>Wait: Waits up to <i>Param3</i> seconds (can contain a decimal point) for a matching process to exist. If <i>Param3</i> is omitted, the command will wait indefinitely. If a matching process is discovered, ErrorLevel(See 30.8) is set to its Process ID (PID). If the command times out, ErrorLevel is set to 0.</p> <p>WaitClose: Waits up to <i>Param3</i> seconds (can contain a decimal point) for ALL matching processes to close. If <i>Param3</i> is omitted, the command will wait indefinitely. If all matching processes are closed, ErrorLevel(See 30.8) is set to 0. If the command times out, ErrorLevel is set to the Process ID (PID) of the first matching process that still exists.</p>
PID-or-Name	<p>This parameter can be either a number (the PID) or a process name as described below. It can also be left blank to change the priority of the script itself.</p> <p>PID: The Process ID, which is a number that uniquely identifies one specific process (this number is valid only during the lifetime of that process). The PID of a newly launched process can be determined via the Run(See 24.5) command. Similarly, the PID of a window can be determined with WinGet(See 28.15). The Process command itself can also be used to discover a PID.</p> <p>Name: The name of a process is usually the same as its executable (without path), e.g. notepad.exe or winword.exe. Since a name might match multiple running processes, only the first process will be operated upon. The name is not case sensitive.</p>
Param3	See <i>Cmd</i> above for details.

ErrorLevel

See *Cmd* above for details.

注意

For *Wait* and *WaitClose*: Processes are checked every 100 milliseconds; the moment the condition is satisfied, the command stops waiting. In other words, rather than waiting for the timeout to expire, it immediately sets [ErrorLevel](#) (See 30.8)as described above, then continues

execution of the script. Also, while the command is in a waiting state, new [threads](#)(See 30.19) can be launched via [hotkey](#)(See 4.), [custom menu item](#)(See 22.13), or [timer](#)(See 17.21).

To work under Windows NT4, the Process command requires the file PSAPI.DLL, which is normally already present in the AutoHotkey installation directory (i.e. no extra installation steps should be needed even for NT). However, to use it from within a [compiled script](#)(See 8.) on Windows NT4, include a copy of PSAPI.DLL in the same folder as the script or in one of the directories in the system's PATH (though some NT4 systems might already have the DLL).

相关命令

[Run](#)(See 24.5), [WinGet](#)(See 28.15), [WinClose](#)(See 28.14), [WinKill](#)(See 28.23), [WinWait](#)(See 28.32), [WinWaitClose](#)(See 28.34), [IfWinExist](#)(See 28.7)

示例

; Example #1:

```
Run Notepad.exe, , , NewPID
Process, priority, %NewPID%, High
MsgBox The newly launched notepad's PID is %NewPID%.
```

; Example #2:

```
Process, wait, Notepad.exe, 5.5
NewPID = %ErrorLevel% ; Save the value immediately since ErrorLevel is often changed.
if NewPID = 0
{
    MsgBox The specified process did not appear within 5.5 seconds.
    return
}
; Otherwise:
MsgBox A matching process has appeared (Process ID is %NewPID%).
Process, priority, %NewPID%, Low
Process, priority, , High ; Have the script set itself to high priority.
```

```
WinClose Untitled - Notepad
```

```
Process, WaitClose, %NewPID%, 5
```

```
if ErrorLevel ; The PID still exists.
```

```
    MsgBox The process did not close within 5 seconds.
```

; Example #3: A hotkey to change the priority of the active window's process:

```
#z::: ; Win+Z hotkey
```

```
WinGet, active_pid, PID, A
```

```
WinGetTitle, active_title, A
```

```
Gui, 5:Add, Text,, Press ESCAPE to cancel, or double-click a new`npriority level for the  
following window: `n%active_title%
```

```
Gui, 5:Add, ListBox, vMyListBox gMyListBox r5,  
Normal|High|Low|BelowNormal|AboveNormal
```

```
Gui, 5:Add, Button, default, OK
```

```
Gui, 5>Show,, Set Priority
```

```
return
```

```
5GuiEscape:
```

```
5GuiClose:
```

```
Gui, Destroy
```

```
return
```

```
MyListBox:
```

```
if A_GuiEvent <> DoubleClick
```

```
    return
```

```
; else fall through to the next label:
```

```
5ButtonOK:
```

```
GuiControlGet, MyListBox
```

```
Gui, Destroy
```

```

Process, Priority, %active_pid%, %MyListBox%

if ErrorLevel

    MsgBox Success: Its priority was changed to "%MyListBox%".

else

    MsgBox Error: Its priority could not be changed to "%MyListBox%".

return

```

; Example #4: Retrieves a list of running processes via DllCall(See 18.3) then shows them in a MsgBox.

```

d = `n ; string separator

s := 4096 ; size of buffers and arrays (4 KB)

Process, Exist ; sets ErrorLevel to the PID of this running script

; Get the handle of this script with PROCESS_QUERY_INFORMATION (0x0400)

h := DllCall("OpenProcess", "UInt", 0x0400, "Int", false, "UInt", ErrorLevel)

; Open an adjustable access token with this process (TOKEN_ADJUST_PRIVILEGES = 32)

DllCall("Advapi32.dll\OpenProcessToken", "UInt", h, "UInt", 32, "UIntP", t)

VarSetCapacity(ti, 16, 0) ; structure of privileges

NumPut(1, ti, 0, 4) ; one entry in the privileges array...

; Retrieves the locally unique identifier of the debug privilege:

DllCall("Advapi32.dll\LookupPrivilegeValueA", "UInt", 0, "Str", "SeDebugPrivilege",
"UIntP", luid)

NumPut(luid, ti, 4, 8)

NumPut(2, ti, 12, 4) ; enable this privilege: SE_PRIVILEGE_ENABLED = 2

; Update the privileges of this process with the new access token:

DllCall("Advapi32.dll\AdjustTokenPrivileges", "UInt", t, "Int", false, "UInt", &ti, "UInt", 0,
"UInt", 0, "UInt", 0)

DllCall("CloseHandle", "UInt", h) ; close this process handle to save memory

```

```

hModule := DllCall("LoadLibrary", "Str", "Psapi.dll") ; increase performance by
preloading the library

s := VarSetCapacity(a, s) ; an array that receives the list of process identifiers:
DllCall("Psapi.dll\EnumProcesses", "UInt", &a, "UInt", s, "UIntP", r)

Loop, % r // 4 ; parse array for identifiers as DWORDs (32 bits):
{
    id := NumGet(a, A_Index * 4)

    ; Open process with: PROCESS_VM_READ (0x0010) |
    PROCESS_QUERY_INFORMATION (0x0400)

    h := DllCall("OpenProcess", "UInt", 0x0010 | 0x0400, "Int", false, "UInt", id)

    VarSetCapacity(m, s) ; an array that receives the list of module handles:
    DllCall("Psapi.dll\EnumProcessModules", "UInt", h, "UInt", &m, "UInt", s, "UIntP", r)

    VarSetCapacity(n, s, 0) ; a buffer that receives the base name of the module:
    e := DllCall("Psapi.dll\GetModuleBaseNameA", "UInt", h, "UInt", m, "Str", n, "Chr", s)

    DllCall("CloseHandle", "UInt", h) ; close process handle to save memory

    If n ; if image is not null add to list:
        l = %l%%n%`d%
}

DllCall("FreeLibrary", "UInt", hModule) ; unload the library to free memory

; Remove the first and last items in the list (possibly ASCII signatures)

StringMid, l, l, InStr(l, d) + 1, InStr(l, d, false, 0) - 2 - InStr(l, d)

StringReplace, l, l, %d%, %d%, UseErrorLevel ; gets the number of processes

;Sort, l, C ; uncomment this line to sort the list alphabetically

MsgBox, 0, %ErrorLevel% Processes, %l%

```

24.5 Run/RunWait

运行一外部程序。与 Run 不同，RunWait 会等待运行的程序结束再继续下一行命令。

Run, Target [, WorkingDir, Max|Min|Hide|UseErrorLevel, OutputVarPID]

参数

Target	<p>运行一文档、网址、可执行文件 (.exe, .com, .bat 等等)、快捷方式(.lnk)或系统动作(见备注)。如果 Target 是个本地文件并且没有给它指定路径, A_WorkingDir(See 9.) 将先被搜索。如果没在那找到匹配的文件, 系统将查找并且运行假如是被整合的("已知")文件, 例如通过被包含在 PATH 文件夹其中的一个。(译注: PATH 文件夹应该指的是操作系统环境变量中的 PATH, 被包含在这个 PATH 中的文件夹下的文件, 都可以按 Win 键+R 后直接输入文件名来运行)</p> <p>要传递参数, 将他们立即添加在程序或文档名字后面。如果一参数包含空格, 将其用双引号括起是最安全的(虽然它也能在某些情况下不需要他们而运作)。</p>
WorkingDir	<p>被运行项目的工作目录。不要将名称括在双引号中即使它包含空格。如果此参数忽略, 脚本所在的工作目录(A_WorkingDir(See 9.))将被使用。</p>
Max Min Hide UseErrorLevel	<p>如果此参数忽略, Target 将只被正常地运行。或者, 它能包含一个或多个这些单词:</p> <p>Max: 最大化运行</p> <p>Min: 最小化运行</p> <p>Hide: 隐藏运行 (以上任一个不能组合使用)</p> <p>注意: 一些程序(例如 Calc.exe)不遵守请求的启动状态, 因此 Max/Min/Hide 将无效。</p> <p>UseErrorLevel: UseErrorLevel 可单独被指定或加在以上单词后(将它与其他单词用空格分开)。如果命令运行失败, 此选项跳过提示对话框, 设置 ErrorLevel(See 30.8) 为单词 ERROR, 并允许当前线程(See 30.19)继续。如果命令运行成功, RunWait 将 ErrorLevel(See 30.8) 设置为程序的退出代码, Run 设置其为零。</p> <p>当 UseErrorLevel 被指定时, 变量 A_LastError 被设置成操作系统的 GetLastError() 函数的结果。A_LastError 是一个在 0 与 4294967295 之间的数字(总为十进制格式, 不是十六进制)。零(0)意味着成功, 除此之外其他数字意味着运行失败。每一数字相当于一个特定的错误状态(要取得一个列表, 在 www.microsoft.com 以"system error codes"搜索)。与 ErrorLevel(See 30.8) 相同, A_LastError 是一个独立线程设定; 也就是说, 被其他线程(See 30.19)打断并不能改变它。但是 A_LastError 也能被 DIICall(See 18.3) 设定。</p>
OutputVarPID	<p>变量的名字用来储存最近运行的程序的唯一 Process ID (PID)(See 24.4)。如果 PID 不能被定义, 变量将为空, 其通常发生在一个系统动作、文档或快捷方式被运行, 而不是一个直接可执行的文件。RunWait 也支持此参数, 尽管它的 OutputVarPID 必须在另一个线程(See 30.19)核实(否则, PID 将因为进程在 RunWait 下一行执行的时候终止而无效)。</p> <p>在 Run 命令取得一个 PID 后, 任何由进程创建的窗口可能还不存在。为了等待至少一个窗口被创建, 使用 WinWait(See 28.32) <code>ahk_pid %OutputVarPID%</code></p>

ErrorLevel

Run: 不设置 [ErrorLevel](#)(See 30.8) 除非使用了 [UseErrorLevel](#)(上面的), 这样的话, **ErrorLevel** 在命令执行失败时被设置成单词 **ERROR** 或者在成功时设置为 **0**。

RunWait: 将 **ErrorLevel** 设为程序的退出代码（一个标记成 32 位的整数型）。如果使用了 [UseErrorLevel](#) 且命令执行失败, 则 **ErrorLevel** 被存作单词 **ERROR**。

备注

不同于 **Run**, **RunWait** 会等到 *Target* 关闭或者退出, 这时 [ErrorLevel](#)(See 30.8) 被设为程序的退出代码（作为一个标记为 32 位的整数型）。即使有些程序仍在运行, **RunWait** 也会马上出现返回；这些程序创建了另外的进程。

如果 *Target* 包含任何逗号, 它们一定要像下面的例子里出现的三次一样被[转义](#)(See 29.5):

```
Run rundll32.exe shell32.dll `,Control_RunDLL desk.cpl `, `, 3 ;打开 控制面板 > 显示 >
设置
```

当通过 [Comspec](#)(See 9.) (即 `cmd.exe`, 译注: 命令解释程序 `cmd.exe` 的路径变量) 运行一个程序时--可能因为你需要重定向此程序的输入或输出--如果可执行文件的路径或名字包含空格, 整个字串应该被一对外引号括起来。下例中, 外引号用红色表示, 所有的内引号用黑色表示:

```
Run %comspec% /c ""C:\My Utility.exe" "param 1" "second param" >"C:\My File.txt""
```

如果 *Target* 不能被运行, 将显示一个错误提示窗口, 这之后脚本会表现为像遇到了一个 [Exit](#)(See 17.6) 命令。为了避免此情况, 在第三个参数里包含 **UseErrorLevel** 字串。

如果 *Target* 是一个精确的路径, 性能可能有细微地改善, 例如 **Run, C:\Windows\Notepad.exe "C:\My Documents\Test.txt"** 而不是 **Run, C:\My Documents\Test.txt**

特殊的 [CLSID](#) 文件夹(See 30.7)可以通过 **Run** 被打开。例如:

```
Run ::{20d04fe0-3aea-1069-a2d8-08002b30309d} ;打开“我的电脑”文件夹。
```

```
Run ::{645ff040-5081-101b-9f08-00aa002f954e} ;打开回收站。
```

系统动作相当于在资源管理器中一个文件的右键菜单动作可用。如果一个文件没有一个动作来运行, 这特殊的文件类型的默认动作（通常是“打开”）将被使用。如果指定了, 这动作应该跟在 **target** 文件名的后面。下列动作是当前被支持的:

properties	为指定文件显示资源管理器的属性窗口。例如: Run, properties "C:\My File.txt" 注意: 当脚本终止的时候属性窗口会自动关闭。为避免这种情况, 使用 WinWait (See 28.32) 来等待窗口出现, 然后用 WinWaitClose (See 28.34) 等待用户去关闭它。
find	打开一个资源管理器的搜索助理窗口或者在指定文件夹的查找文件窗口。例如: Run, find D:\
explore	在指定文件夹打开一个资源管理器窗口。例如: Run, explore %A_ProgramFiles%
edit	打开一个指定的文件来编辑。如果指定的文件类型没有“编辑”动作与之关联, 它可能不起作用。例如: Run, edit "C:\My File.txt"
open	打开指定的文件 (一般不需要, 因为这是大多数文件类型的默认动作)。例如: Run, open "My File.txt"

print

使用与之关联的应用程序来打印指定的文件，如果有的话。例如：Run, print "My File.txt"

当 RunWait 在等待状态时，新的线程(See 30.19)能通过热键(See 4.)，自定义菜单项(See 22.13)或者定时器(See 17.21)来运行。

相关命令

[RunAs](#)(See 24.6), [Process](#)(See 24.4), [Exit](#)(See 17.6), [CLSID List](#)(See 30.7), [DIICall](#)(See 18.3)

范例

```
Run, Notepad.exe, C:\My Documents, max

Run, mailto:someone@domain.com?subject=This is the subject line&body=This is the
message body's text.
Run, ReadMe.doc, , Max UseErrorLevel ;以最小化运行并且如果失败则不显示对话框。
if ErrorLevel = ERROR
    MsgBox 无法启动文档。

RunWait, %comspec% /c dir c:\ >>c:\DirTest.txt, , min

Run, c:\DirTest.txt

Run, properties c:\DirTest.txt

Run, www.autohotkey.com ;也就是说能运行任何 URL。
Run, mailto:someone@somedomain.com ;这将打开默认的 e-mail 程序。

Run ::{20d04fe0-3aea-1069-a2d8-08002b30309d} ;打开“我的电脑”文件夹。
Run ::{645ff040-5081-101b-9f08-00aa002f954e} ;打开回收站。

;连续执行多个命令，在每个命令之间用“&&”:
Run, %comspec% /c dir /b > C:\list.txt && type C:\list.txt && pause
```

翻译：天堂之门 menk33@163.com 2008 年 10 月 24 日

24.6 RunAs

为所有随后要使用的 [Run\(See 24.5\)](#) 和 [RunWait\(See 24.5\)](#) 指定一组用户凭证。需要 Windows 2000/XP 或之后版本。

RunAs [, User, Password, Domain]

参数

User	如果省略了此参数以及所有其它参数, RunAs 特性将被关闭, 这将使 Run(See 24.5) 和 RunWait(See 24.5) 恢复到它们默认的特性。不然它就是将被创建的新进程所属的用户名。
Password	User 的密码。
Domain	User 的域。要使用本地账户的话, 将此留空。如果那样不行, 试着使用 @你的计算机名称。

注意

只支持 Windows XP/2000 或之后版本; 此命令对其它操作系统无效。NT4 用户应该从 NT 资源工具箱中安装并使用 SU 命令代替。

此命令只会通知 AutoHotkey 对所有随后要使用的 [Run\(See 24.5\)](#) 和 [RunWait\(See 24.5\)](#) 使用(或不使用)轮流的用户凭证。

[ErrorLevel\(See 30.8\)](#) 不会被此命令改变。如果指定了一个无效的 *User, Password, 或 Domain, Run(See 24.5)* 和 [RunWait\(See 24.5\)](#) 将显示一个错误消息来解释问题所在(除非它们的 [UseErrorLevel 选项\(See 24.5\)](#) 处于有效状态)。

当 RunAs 特性生效时, [Run\(See 24.5\)](#) 和 [RunWait\(See 24.5\)](#) 将不能启动文档、URL 或系统动作。换言之, 要启动的文件必须是一个可执行文件。

要使用此命令, "Secondary Logon" 服务必须设为手动或者自动(如果设为手动, 操作系统应该能够在需要时自动地启用此服务)。

相关命令

[Run\(See 24.5\)](#), [RunWait\(See 24.5\)](#)

示例

```
RunAs, Administrator, MyPassword
```

```
Run, RegEdit.exe
```

```
RunAs ;重设为普通特性。
```

翻译: tcgbp tcguobaoping@gmail.com 修正: 天堂之门 menk33@163.com 2009 年 1 月 5 日

24.7 Shutdown

关机、重启或注销操作系统。

Shutdown, Code

参数

Code	一组 shutdown 代码如下所列。
------	---------------------

注意

shutdown 代码是下列数值的一个组合：

Logoff (注销)	0
Shutdown (关机)	1
Reboot (重启)	2
Force (强制)	4
Power down (切断电源)	8
Suspend/Hibernate (挂起/休眠)	请看此页面底部的 DII Call 示例。

将需要的数值相加。例如，要关机并切断电源的代码将是 **9** (关机 + 切断电源 = $1 + 8 = 9$)。或者，也能指定一个像 **1+8** 这样的 [expression](#)(See 9.)(表达式)。

"Force" 数值(4) 强制关闭所有打开的应用程序。它仅仅应该在一个紧急情况时使用，因为它可能导致任何打开的应用程序丢失数据。

"Power down" 数值 关闭操作系统并且切断电源。

一个相关的提示，当操作系统关闭或用户通过 [OnExit](#)(See 17.17) 注销时，脚本能探测到。

相关命令

[Run](#)(See 24.5), [ExitApp](#)(See 17.7), [OnExit](#)(See 17.17)

示例

```
; 强制重启 (重启 + 强制 = 2 + 4 = 6):
Shutdown, 6

; 调用 Windows API 函数 "SetSuspendState" 来使操作系统挂起或者休眠。
; Windows 95/NT4: 由于此函数不存在，下面的调用将无效。
; 参数 #1: 传递 1 来代替 0 进行休眠而不是挂起。
```

; 参数 #2: 传递 1 来代替 0 立即进行挂起而不是询问每个应用程序获得许可。

; 参数 #3: 传递 1 来代替 0 去禁止所有的唤醒事件。

```
DllCall("PowrProf\SetSuspendState", "int", 0, "int", 0, "int", 0)
```

翻译: 天堂之门 menk33@163.com 2008 年 8 月 12 日

24.8 Sleep

在继续前等待指定的时间量。

[Sleep](#), [DelayInMilliseconds](#)

参数

Delay	要停顿的时间量(以毫秒形式)在 0 和 2147483647 (24 天) 之间, 其可以是一个 表达式 (See 9.)。
-------	--

说明

由于操作系统的时间控制系统的间隔尺寸, *Delay* 典型地上舍入为最临近 10 的倍数。例如, 在大多数 Windows NT/2000/XP 系统上, 一个在 1 和 10 (包含的)之间的 *delay* 相当于 10。不过, 由于硬件差异, 一些系统将上舍入为一个不同的值像 15。

如果 CPU (中央处理器)处于负担状态, 实际延迟时间可能比它请求的要更久结束。这是因为操作系统在给脚本另一个时间片之前, 给每个有需要的进程一个 CPU 时间片(典型地有 20 毫秒)。

一个为 0 的 *delay* 让出脚本当前的时间片的剩余给任何其他需要它的进程(只要它们不在 [priority](#)(See 24.4)(优先权)上比脚本显著地较低)。因此, 一个为 0 的 *delay* 产生一个在 0 和 20ms (或更久)之间的实际延迟, 取决于有需要的进程的数量 (如果无有需要的进程, 也就根本没有 *delay*)。不过, 一个为 0 的 *Delay* 将总是比任何更长的 *Delay* 结束得更早。

当脚本停顿时, 新的 [threads](#)(See 30.19)(线程) 能通过 [hotkey](#)(See 4.)、[自定义菜单项](#)(See 22.13) 或 [timer](#)(See 17.21) 被运行。

"Sleep -1": 一个为 -1 的 *delay* 不会停顿, 而是让脚本立即检查它的消息队列。这能被用来强制任何待定的 [中断](#)(See 30.19) 发生在一个特定的地方, 而不是更随机的某处。详见 [Critical](#)(See 22.7)。

相关命令

[SetKeyDelay](#)(See 20.14), [SetMouseDelay](#)(See 23.8), [SetControlDelay](#)(See 28.1.13),
[SetWinDelay](#)(See 28.9), [SetBatchLines](#)(See 17.20)

范例

```
Sleep, 1000 ; 1 秒
```

翻译: 天堂之门 menk33@163.com 2008 年 7 月 24 日

25. 注册表管理

25.1 Loop (registry)

Retrieves the contents of the specified registry subkey, one item at a time.

`Loop, RootKey [, Key, IncludeSubkeys?, Recurse?]`

参数

RootKey	Must be either HKEY_LOCAL_MACHINE (or HKLM), HKEY_USERS (or HKU), HKEY_CURRENT_USER (or HKCU), HKEY_CLASSES_ROOT (or HKCR), or HKEY_CURRENT_CONFIG (or HKCC). To access a remote registry, prepend the computer name and a colon, as in this example: \\workstation01:HKEY_LOCAL_MACHINE
Key	The name of the key (e.g. Software\SomeApplication). If blank or omitted, the contents of <i>RootKey</i> will be retrieved.
IncludeSubkeys?	0 (default) Subkeys contained within <i>Key</i> are not retrieved (only the values). 1 All values and subkeys are retrieved. 2 Only the subkeys are retrieved (not the values).
Recurse?	0 (default) Subkeys are not recursed into. 1 Subkeys are recursed into, so that all values and subkeys contained therein are retrieved.

注意

A registry-loop is useful when you want to operate on a collection registry values or subkeys, one at a time. The values and subkeys are retrieved in reverse order (bottom to top) so that [RegDelete](#)(See 25.2) can be used inside the loop without disrupting the loop.

The following variables exist within any registry-loop. If an inner registry-loop is enclosed by an outer registry-loop, the innermost loop's registry item will take precedence:

A_LoopRegName	Name of the currently retrieved item, which can be either a value name or the name of a subkey. Value names displayed by Windows RegEdit as "(Default)" will be retrieved if a value has been assigned to them, but A_LoopRegName will be blank for them.
A_LoopRegType	The type of the currently retrieved item, which is one of the following words: KEY (i.e. the currently retrieved item is a subkey not a value), REG_SZ, REG_EXPAND_SZ, REG_MULTI_SZ, REG_DWORD, REG_QWORD, REG_BINARY, REG_LINK, REG_RESOURCE_LIST,

	REG_FULL_RESOURCE_DESCRIPTOR, REG_RESOURCE_REQUIREMENTS_LIST, REG_DWORD_BIG_ENDIAN (probably rare on most Windows hardware). It will be empty if the currently retrieved item is of an unknown type.
A_LoopRegKey	The name of the root key being accessed (HKEY_LOCAL_MACHINE, HKEY_USERS, HKEY_CURRENT_USER, HKEY_CLASSES_ROOT, or HKEY_CURRENT_CONFIG). For remote registry access, this value will not include the computer name.
A_LoopRegSubKey	Name of the current SubKey. This will be the same as the <i>Key</i> parameter unless the <i>Recurse</i> parameter is being used to recursively explore other subkeys. In that case, it will be the full path of the currently retrieved item, not including the root key. For example: Software\SomeApplication\My SubKey
A_LoopRegTimeModified	The time the current subkey or any of its values was last modified. Format YYYYMMDDHH24MISS (See 16.26). This variable will be empty if the currently retrieved item is not a subkey (i.e. A_LoopRegType is not the word KEY) or if the operating system is Win9x (since Win9x does not track this info).

When used inside a registry-loop, the following commands can be used in a simplified way to indicate that the currently retrieved item should be operated upon:

RegRead (See 25.3), OutputVar	Reads the current item. If the current item is a key, ErrorLevel will be set to 1 and <i>OutputVar</i> will be made empty.
RegWrite (See 25.4) [, Value]	Writes to the current item. If <i>Value</i> is omitted, the item will be made 0 or blank depending on its type. If the current item is a key, ErrorLevel will be set to 1 and there will be no other effect.
RegDelete (See 25.2)	Deletes the current item. If the current item is a key, it will be deleted along with any subkeys and values it contains.

When accessing a remote registry (via the *RootKey* parameter described above), the following notes apply:

- The target machine must be running the Remote Registry service, or (for Windows Me/98/95) have the Microsoft Remote Registry service installed (this can be obtained from Microsoft).
- Access to a remote registry may fail if the target computer is not in the same domain as yours or the local or remote username lacks sufficient permissions (however, see below for possible workarounds).
- Depending on your username's domain, workgroup, and/or permissions, you may have to connect to a shared device, such as by mapping a drive, prior to attempting remote registry access. Making such a connection -- using a remote username and password

that has permission to access or edit the registry -- may as a side-effect enable remote registry access.

- If you're already connected to the target computer as a different user (for example, a mapped drive via user Guest), you may have to terminate that connection to allow the remote registry feature to reconnect and re-authenticate you as your own currently logged-on username.
- For Windows Server 2003 and Windows XP Professional, MSDN states: "If the [registry server] computer is joined to a workgroup and the *Force network logons using local accounts to authenticate as Guest* policy is enabled, the function fails. Note that this policy is enabled by default if the computer is joined to a workgroup."
- For Windows XP Home Edition, MSDN states that "this function always fails". Home Edition lacks both the registry server and client, though the client can be extracted from one of the OS cab files.

See [Loop\(See 17.12\)](#) for information about [Blocks\(See 17.2\)](#), [Break\(See 17.3\)](#), [Continue\(See 17.4\)](#), and the A_Index variable (which exists in every type of loop).

相关命令

[Loop\(See 17.12\)](#), [Break\(See 17.3\)](#), [Continue\(See 17.4\)](#), [Blocks\(See 17.2\)](#), [RegRead\(See 25.3\)](#), [RegWrite\(See 25.4\)](#), [RegDelete\(See 25.2\)](#)

示例

```
; Example: Delete Internet Explorer's history of URLs typed by the user:
```

```
Loop, HKEY_CURRENT_USER, Software\Microsoft\Internet Explorer\TypedURLs
    RegDelete
```

```
; Example: A working test script:
```

```
Loop, HKEY_CURRENT_USER, Software\Microsoft\Windows, 1, 1
{
    if a_LoopRegType = key
        value =
    else
        {
            RegRead, value
            if ErrorLevel
```

```

        value = *error*

    }

MsgBox, 4, , %a_LoopRegName% = %value% (%a_LoopRegType%)`n`nContinue?

IfMsgBox, NO, break

}

```

; Example: A working example to recursively search the entire

; registry for particular value(s).

SetBatchLines -1 ; Makes searching occur at maximum speed.

RegSearchTarget = Notepad ; Tell the subroutine what to search for.

Gosub, RegSearch

return

RegSearch:

ContinueRegSearch = y

Loop, HKEY_LOCAL_MACHINE, , 1, 1

{

Gosub, CheckThisRegItem

if ContinueRegSearch = n ; It told us to stop.

return

}

Loop, HKEY_USERS, , 1, 1

{

Gosub, CheckThisRegItem

if ContinueRegSearch = n ; It told us to stop.

return

}

Loop, HKEY_CURRENT_CONFIG, , 1, 1

{

```

Gosub, CheckThisRegItem

if ContinueRegSearch = n ; It told us to stop.

    return

}

; Note: I believe HKEY_CURRENT_USER does not need to be searched if HKEY_USERS
; is being searched. The same might also be true for HKEY_CLASSES_ROOT if
; HKEY_LOCAL_MACHINE is being searched.

return

CheckThisRegItem:

if A_LoopRegType = KEY ; Remove these two lines if you want to check key names too.

    return

RegRead, RegValue

if ErrorLevel

    return

IfInString, RegValue, %RegSearchTarget%

{

    MsgBox, 4, , The following match was
found: `n%A_LoopRegKey%\%A_LoopRegSubKey%\%A_LoopRegName%`nValue
= %RegValue%`n`nContinue?

    IfMsgBox, No

        ContinueRegSearch = n ; Tell our caller to stop searching.

}

return

```

25.2 RegDelete

从注册表中删除一个子键或一个值。

RegDelete, RootKey, SubKey [, ValueName]

参数

RootKey	只能是 HKEY_LOCAL_MACHINE 、 HKEY_USERS 、 HKEY_CURRENT_USER 、
---------	---

	HKEY_CLASSES_ROOT、HKEY_CURRENT_CONFIG 其中之一（或者它们的缩写，例如 HKLM）。要访问远程注册表，在前面加上电脑名和一个冒号，例如：\\workstation01:HKEY_LOCAL_MACHINE
SubKey	子键的名字（例如 Software\SomeApplication）。
ValueName	需要删除的项的名字。如果省略，整个 SubKey 将会被删除。要删除 Subkey 的默认值——也就是在 RegEdit 中显示为“(Default)/ 默认”的值——这个参数使用 AHK_DEFAULT。

ErrorLevel

如果遇到问题，[ErrorLevel/错误级别\(See 30.8\)](#) 被设置为 1，否则为 0。

注意

对注册表进行删除操作可能会引发潜在的问题——使用前请三思！

要返回多个注册表值或对多个注册表值进行操作，推荐使用 [registry-loop\(See 17.16\)](#)。

要获取访问远程注册表的详细资料，可以查看 [registry-loop\(See 17.16\)](#) 中的注释。

相关命令

[RegRead\(See 25.3\)](#), [RegWrite\(See 25.4\)](#), [Registry-loop\(See 17.16\)](#), [IniDelete\(See 16.28\)](#)

示例

```
RegDelete, HKEY_LOCAL_MACHINE, Software\SomeApplication, TestValue
```

25.3 RegRead

从注册表中读取一个值。

`RegRead, OutputVar, RootKey, SubKey [, ValueName]`

参数

OutputVar	存储返回值的变量名。如果获取值失败，它会被设置为空，同时 ErrorLevel/错误级别(See 30.8) 被设置为 1。
RootKey	只能是 HKEY_LOCAL_MACHINE、HKEY_USERS、HKEY_CURRENT_USER、HKEY_CLASSES_ROOT、HKEY_CURRENT_CONFIG 其中之一（或者它们的缩写，例如 HKLM）。要访问远程注册表，在前面加上电脑名和一个冒号，例如：\\workstation01:HKEY_LOCAL_MACHINE
SubKey	子键的名字（例如 Software\SomeApplication）。
ValueName	需要读取的项的名称。如果省略，将返回 Subkey 的默认值，也就是在 RegEdit 中显示

为"(Default)/默认"的值。如果没有默认值（也就是 RegEdit 中显示"value not set/数值未设置"），OutputVar 被设置为空，同时 ErrorLevel/错误级别 被设置为 1。

ErrorLevel/错误级别

如果遇到问题（例如读取不存在的键或值），ErrorLevel/错误级别(See 30.8) 被设置为 1，否则为 0。

注意

目前只支持这些类型的值: REG_SZ/字符串值，REG_EXPAND_SZ/可扩充字符串值，REG_MULTI_SZ/多字符串值，REG_DWORD/DWORD 值，以及 REG_BINARY/二进制值。

REG_DWORD/DWORD 类型的值总是被转换为正的十进制数字。

REG_BINARY/二进制 类型的值最多只能读取 64KB，其它类型的值则没有限制。但是在 Windows 95/98/ME 下，所有类型的值最多只能读取 64KB（Win9x 的注册表不能存储过大的数据）。

当读取 REG_BINARY/二进制 类型的值的时候，会返回一个 16 进制的字符串。例如，REG_BINARY/二进制 值为 01,a9,ff,77，读取出来的结果为 01A9FF77。

当读取一个 REG_MULTI_SZ/多字符串 类型的值的时候，值的每一个部分以换行符分隔 (`n)。如果值为空，OutputVar 被设置为空。FileSelectFile(See 16.23) 中的示例演示了怎么将 OutputVar 分解为各个独立的部分。注意：Windows 95 不支持 REG_MULTI_SZ/多字符串 类型的值。

要返回多个注册表值或对多个注册表值进行操作，推荐使用 registry-loop(See 17.16)。

要获取访问远程注册表的详细资料，可以查看 registry-loop(See 17.16) 中的注释。

相关命令

[RegDelete](#)(See 25.2), [RegWrite](#)(See 25.4), [Registry-loop](#)(See 17.16), [IniRead](#)(See 16.29)

示例

```
RegRead, OutputVar, HKEY_LOCAL_MACHINE,  
SOFTWARE\Microsoft\Windows\CurrentVersion, ProgramFilesDir  
MsgBox, Program files are in: %OutputVar%
```

; 下面的例子返回一个注册表值的类型（例如 REG_SZ/字符串值 或 REG_DWORD/DWORD 值）。

```
 MsgBox % RegKeyType("HKCU", "Environment", "TEMP")  
return
```

RegKeyType(RootKey, SubKey, ValueName) ; 这个函数返回指定值的类型。

```
{
    Loop, %RootKey%, %SubKey%
        if (A_LoopRegName = ValueName)
            return A_LoopRegType
        return "Error"
}
```

25.4 RegWrite

写入一个值到注册表中。

`RegWrite, ValueType, RootKey, SubKey [, ValueName, Value]`

参数

<code>ValueType</code>	只能是 <code>REG_SZ</code> , <code>REG_EXPAND_SZ</code> , <code>REG_MULTI_SZ</code> , <code>REG_DWORD</code> 或者 <code>REG_BINARY</code> 。
<code>RootKey</code>	只能是 <code>HKEY_LOCAL_MACHINE</code> 、 <code>HKEY_USERS</code> 、 <code>HKEY_CURRENT_USER</code> 、 <code>HKEY_CLASSES_ROOT</code> 、 <code>HKEY_CURRENT_CONFIG</code> 其中之一 (或者它们的缩写, 例如 <code>HKLM</code>)。要访问远程注册表, 在前面加上电脑名和一个冒号, 例如: <code>\\\workstation01:HKEY_LOCAL_MACHINE</code>
<code>SubKey</code>	子键的名字 (例如 <code>Software\SomeApplication</code>)。如果 <code>SubKey</code> 不存在, 将会自动创建 (在它的父类之下, 如果有的话)。如果 <code>SubKey</code> 留空, 将把值直接写入到 <code>RootKey</code> 中 (某些系统会拒绝将值直接写入 <code>HKEY_CURRENT_USER</code> 根键之下)。
<code>ValueName</code>	需要写入值的项的名称。如果留空或省略, 默认修改的是 <code>Subkey</code> 的默认值, 也就是在 <code>RegEdit</code> 中显示为“(Default)/默认”的值
<code>Value</code>	需要写入的值。如果省略, 默认是一个空字符串, 或 <code>0</code> , 这个取决于 <code>ValueType</code> 。如果写入的文本过长, 可以使用 continuation section/字符串分段(See 8.) 的方法将它分为几个短小的段落, 这样可以增加代码的可读性和可维护性。

ErrorLevel

如果遇到问题, [ErrorLevel/错误级别\(See 30.8\)](#) 被设置为 `1` , 否则为 `0` 。

注意

如果 `ValueType` 是 `REG_DWORD` , `Value` 的取值范围是 -2147483648 到 4294967295
(`0xFFFFFFFF`) 。

`REG_BINARY/二进制` 和 `REG_MULTI_SZ/多字符串` 类型的值最多只能写入 64KB , 其它类型的值则没有限制。但是在 Windows 95/98/ME 下, 所有类型的值最多只能写入 64KB 。如果超过 64KB , 则

超过范围之后的数据将不会被写入。也就是说，在一个比较大的字符串中，只有最开始的 64KB 数据会被写入注册表。

当写入一个 REG_BINARY/二进制 类型的值的时候，使用一个 16 进制的字符串。例如，REG_BINARY/二进制 值为 01,a9,ff,77，写入的字符串为 01A9FF77。

当写入一个 REG_MULTI_SZ/多字符串 类型的值的时候，你必须使用换行符 (`n) 来分隔各个部分，最后一个部分的结尾可以不使用换行符。任何一个部分都不能为空。也就是说，不要在一行中连着使用两个换行符 (`n`n)，因为这样会写入一个 shorter-than-expected/比预期短 的值。注意：Windows 95 不支持 REG_MULTI_SZ/多字符串 类型的值。

要返回多个注册表值或对多个注册表值进行操作，推荐使用 [registry-loop\(See 17.16\)](#)。

要获取访问远程注册表的详细资料，可以查看 [registry-loop\(See 17.16\)](#) 中的注释。

相关命令

[RegDelete\(See 25.2\)](#), [RegRead\(See 25.3\)](#), [Registry-loop\(See 17.16\)](#), [IniWrite\(See 16.30\)](#)

示例

```
RegWrite, REG_SZ, HKEY_LOCAL_MACHINE, SOFTWARE\TestKey, MyValueName, Test
Value

RegWrite, REG_BINARY, HKEY_CURRENT_USER, Software\TEST_APP, TEST_NAME,
01A9FF77

RegWrite, REG_MULTI_SZ, HKEY_CURRENT_USER, Software\TEST_APP, TEST_NAME,
Line1`nLine2
```

26. 声音管理

26.1 SoundBeep

从个人计算机的扬声器发出一个音调。

SoundBeep [, Frequency, Duration]

参数

Frequency	声音的频率，可以是一个 表达式(See 9.) 。必须是 37 和 32767 之间的一个数字。如果省略，频率将为 523。
Duration	声音持续的时间，以毫秒计（可以是一个 表达式(See 9.) ）。如果省略，持续时间将为 150。

注意

等声音结束脚本才会继续。此外，发音期间系统的响应可能被减慢。

在 Windows 9x 上, *Frequency* 和 *Duration* 参数被忽略。取而代之，声卡将播放系统默认的事件声音。如果计算机没有声卡，扬声器将播放一个标准的哔音。

要生成一个标准的系统声音而不是个人计算机扬声器的哔音，请看 [SoundPlay\(See 26.4\)](#) 命令中的星号模式。

相关命令

[SoundPlay\(See 26.4\)](#)

示例

```
SoundBeep ; 播放默认的音高和持续时间。
```

```
SoundBeep, 750, 500 ; 以更高的音调播放半秒钟。
```

翻译：天堂之门 menk33@163.com 2008 年 10 月 11 日

26.2 SoundGet

Retrieves various settings from a sound device (master mute, master volume, etc.)

SoundGet, OutputVar [, ComponentType, ControlType, DeviceNumber]

参数

OutputVar	The name of the variable in which to store the retrieved setting, which is either a floating point number between 0 and 100 (inclusive) or the word ON or OFF (used only for <i>ControlTypes</i> ONOFF, MUTE, MONO, LOUDNESS, STEREOENH, and BASSBOOST). The variable will be made blank if there was a problem retrieving the setting. The format of the floating point number, such as its decimal places, is determined by SetFormat(See 21.10) .
ComponentType	<p>If omitted or blank, it defaults to the word MASTER. Otherwise, it can be one of the following words: MASTER (synonymous with SPEAKERS), DIGITAL, LINE, MICROPHONE, SYNTH, CD, TELEPHONE, PCSPEAKER, WAVE, AUX, ANALOG, HEADPHONES, or N/A. If the sound device lacks the specified <i>ComponentType</i>, ErrorLevel will indicate the problem.</p> <p>The component labeled Auxiliary in some mixers might be accessible as ANALOG rather than AUX.</p> <p>If a device has more than one instance of <i>ComponentType</i> (two of type LINE, for example), usually the first contains the playback settings and the second contains the recording settings. To access an instance other than the first, append a colon and a number to this parameter. For example: Analog:2 is the</p>

	second instance of the analog component.
ControlType	If omitted or blank, it defaults to VOLUME. Otherwise, it can be one of the following words: VOLUME (or VOL), ONOFF, MUTE, MONO, LOUDNESS, STEREOENH, BASSBOOST, PAN, QSOUNDPAN, BASS, TREBLE, EQUALIZER, or the number of a valid control type (see soundcard analysis script (See 26.5)). If the specified <i>ComponentType</i> lacks the specified <i>ControlType</i> , ErrorLevel will indicate the problem.
DeviceNumber	If this parameter is omitted, it defaults to 1 (the first sound device), which is usually the system's default device for recording and playback. Specify a number higher than 1 to operate upon a different sound device. This parameter can be an expression (See 9.).

ErrorLevel

[ErrorLevel](#)(See 30.8) is set to 0 if the command succeeded. Otherwise, it is set to one of the following phrases:

Invalid Control Type or Component Type
Can't Open Specified Mixer
Mixer Doesn't Support This Component Type
Mixer Doesn't Have That Many of That Component Type
Component Doesn't Support This Control Type
Can't Get Current Setting

注意

On Windows Vista, SoundGet retrieves the script's own settings, not those of the active window. This may be resolved in a future version. To work around it, in the properties dialog for the file "AutoHotkey.exe" (or a [compiled script](#)(See 8.)), change the compatibility setting to "Windows XP".

To discover the capabilities of the sound devices (mixers) installed on the system -- such as the available component types and control types -- run the [soundcard analysis script](#)(See 26.5).

Use [SoundSet](#)(See 26.5) to change a setting.

相关命令

[SoundSet](#)(See 26.5), [SoundGetWaveVolume](#)(See 26.3), [SoundSetWaveVolume](#)(See 26.6), [SoundPlay](#)(See 26.4)

示例

```

SoundGet, master_volume

MsgBox, Master volume is %master_volume% percent.

SoundGet, master_mute, , mute

MsgBox, Master Mute is currently %master_mute%.

SoundGet, bass_level, Master, bass

if ErrorLevel

    MsgBox, Error Description: %ErrorLevel%

else

    MsgBox, The BASS level for MASTER is %bass_level% percent.

SoundGet, microphone_mute, Microphone, mute

if microphone_mute = Off

    MsgBox, The microphone is not muted.

```

26.3 SoundGetWaveVolume

获取一个音频设备的波形输出音量。

SoundGetWaveVolume, OutputVar [, DeviceNumber]

参数

OutputVar	储存了取得的音量等级的变量名称，它是一个包括 0 到 100 的浮点数。如果在获取音量的过程中遇到问题，此变量将为空。浮点数的格式比如它的小数位数，由 SetFormat(See 21.10) 命令决定。
DeviceNumber	如果省略此参数，其默认为 1 (第一个音频设备)，通常是系统默认的录音和播放设备。可以通过指定一个大于 1 的数字来操作其他的音频设备。

ErrorLevel

如果出现问题，[ErrorLevel\(See 30.8\)](#) 将设为 1，其他情况为 0。

注意

SoundGetWaveVolume 还不被 Windows Vista 支持；它将使 *OutputVar* 为空。

可以通过 [SoundSetWaveVolume\(See 26.6\)](#) 设定当前波形输出音量的等级。其他设置例如总音量、合成器、麦克风、静音、高音和低音可以用 [SoundSet\(See 26.5\)](#) 和 [SoundGet\(See 26.2\)](#) 进行设定和获取。

相关命令

[SoundSetWaveVolume\(See 26.6\)](#), [SoundSet\(See 26.5\)](#), [SoundGet\(See 26.2\)](#), [SoundPlay\(See 26.4\)](#)

示例

```
SoundGetWaveVolume, OutputVar
```

```
MsgBox, 当前的波形输出音量的等级为 %OutputVar%`%。
```

翻译: [Kookee](#) 修正: 天堂之门 menk33@163.com 2008年10月18日

26.4 SoundPlay

播放一个音频、视频或者其他支持的文件类型。

`SoundPlay, Filename [, wait]`

参数

Filename	<p>要被播放的文件名称，如果绝对路径未指定将假设在 %A_WorkingDir%(See 9.)。</p> <p>要发出标准的系统声音，像下面显示的那样在一个星号后面指定一个数字。注意: <code>wait</code> 参数在此模式中无效。</p> <ul style="list-style-type: none"> *-1: 简单的哔哔声。如果声卡不可用，此声音将使用主板扬声器来生成。 *16: 手型(停止/错误声) *32: 问号声 *48: 感叹声 *64: 星号(消息声)
wait	<p>如果省略，当文件在播放时脚本的 current thread(See 30.19)(当前线程) 将移动到下一个命令。要避免这样，指定此参数为 1 或者是单词 WAIT，这能使当前线程在继续前等待文件播放结束。甚至在等待时，新的 threads(See 30.19) 也能通过 hotkey(See 4.)、custom menu item(See 22.13) 或者 timer(See 17.21) 来启动。</p> <p>已知限制：如果省略 WAIT 参数，操作系统可能会认为播放的文件“在使用中”直到脚本关闭或者直到另一个文件被播放(甚至一个不存在的文件)。</p>

ErrorLevel

如果遇到一个问题 [ErrorLevel\(See 30.8\)](#) 设为 1，否则是 0。

注意

所有的 Windows 操作系统都能够播放 .wav 文件。不过，如果操作系统没有安装正确的编解码器或者功能，那么其他文件(.mp3、.avi 等) 可能无法播放。

如果一个文件正在播放并且当前脚本播放了第二个文件，那么第一个文件将停止播放以便第二个文件能播放。在一些操作系统上，即使当一个完全独立的脚本播放一个新的文件时，特定的文件类型也可能停止播放。

要停止一个当前播放的文件，像这个例子一样，在一个不存在的文件名称上使用 SoundPlay: SoundPlay, Nonexistent.avi

如果脚本退出，任何由它启动的当前正在播放的文件将会停止。

相关命令

[SoundBeep](#)(See 26.1), [SoundGet](#)(See 26.2), [SoundSet](#)(See 26.5),
[SoundGetWaveVolume](#)(See 26.3), [SoundSetWaveVolume](#)(See 26.6), [MsgBox](#)(See 19.11),
[Threads](#)(See 30.19)

示例

```
SoundPlay, %A_WinDir%\Media\ding.wav
```

翻译：天堂之门 menk33@163.com 2008 年 8 月 12 日

26.5 SoundSet

Changes various settings of a sound device (master mute, master volume, etc.)

SoundSet, NewSetting [, ComponentType, ControlType, DeviceNumber]

参数

NewSetting	<p>Percentage number between -100 and 100 inclusive (it can be a floating point number or expression(See 9.)). If the number begins with a plus or minus sign, the current setting will be adjusted up or down by the indicated amount. Otherwise, the setting will be set explicitly to the level indicated by <i>NewSetting</i>.</p> <p>For <i>ControlTypes</i> with only two possible settings -- namely ONOFF, MUTE, MONO, LOUDNESS, STEREOENH, and BASSBOOST -- any positive number will turn on the setting and a zero will turn it off. However, if the number begins with a plus or minus sign, the setting will be toggled (set to the opposite of its current state).</p>
ComponentType	If omitted or blank, it defaults to the word MASTER. Otherwise, it can be one

	<p>of the following words: MASTER (synonymous with SPEAKERS), DIGITAL, LINE, MICROPHONE, SYNTH, CD, TELEPHONE, PCSPEAKER, WAVE, AUX, ANALOG, HEADPHONES, or N/A. If the sound device lacks the specified <i>ComponentType</i>, ErrorLevel will indicate the problem.</p> <p>The component labeled Auxiliary in some mixers might be accessible as ANALOG rather than AUX.</p> <p>If a device has more than one instance of <i>ComponentType</i> (two of type LINE, for example), usually the first contains the playback settings and the second contains the recording settings. To access an instance other than the first, append a colon and a number to this parameter. For example: Analog:2 is the second instance of the analog component.</p>
ControlType	If omitted or blank, it defaults to VOLUME. Otherwise, it can be one of the following words: VOLUME (or VOL), ONOFF, MUTE, MONO, LOUDNESS, STEREOENH, BASSBOOST, PAN, QSOUNDPAN, BASS, TREBLE, EQUALIZER, or the number of a valid control type (see soundcard analysis script). If the specified <i>ComponentType</i> lacks the specified <i>ControlType</i> , ErrorLevel will indicate the problem.
DeviceNumber	If this parameter is omitted, it defaults to 1 (the first sound device), which is usually the system's default device for recording and playback. Specify a number higher than 1 to operate upon a different sound device. This parameter can be an expression (See 9.).

ErrorLevel

[ErrorLevel](#)(See 30.8) is set to 0 if the command succeeded. Otherwise, it is set to one of the following phrases:

Invalid Control Type or Component Type
Can't Open Specified Mixer
Mixer Doesn't Support This Component Type
Mixer Doesn't Have That Many of That Component Type
Component Doesn't Support This Control Type
Can't Get Current Setting
Can't Change Setting

注意

On Windows Vista, SoundSet and SoundGet affect only the script itself (this may be resolved in a future version). There are at least two ways to work around this:

- 1) In the properties dialog for the file "AutoHotkey.exe" (or a [compiled script](#)(See 8.)), change the compatibility setting to "Windows XP".
- 2) Have the script send volume-control keystrokes to change the master volume for the entire system. For example:

```
Send {Volume_Up} ; Raise the master volume by 1 interval (typically 5%).
Send {Volume_Down 3} ; Lower the master volume by 3 intervals.
Send {Volume_Mute} ; Mute/unmute the master volume.
```

To discover the capabilities of the sound devices (mixers) installed on the system -- such as the available component types and control types -- run the [soundcard analysis script](#).

When SoundSet changes the volume of a component, all of that component's channels (e.g. left and right) are set to the same level. In other words, any off-center "balance" that may have been set previously is lost. This can be avoided for the WAVE component by using [SoundSetWaveVolume](#)(See 26.6) instead, which attempts to preserve the existing balance when changing the volume level.

Use [SoundGet](#)(See 26.2) to retrieve the current value of a setting.

相关命令

[SoundGet](#)(See 26.2), [SoundGetWaveVolume](#)(See 26.3), [SoundSetWaveVolume](#)(See 26.6), [SoundPlay](#)(See 26.4)

示例

```
; BASIC EXAMPLES:

SoundSet, 50 ; Set the master volume to 50%
SoundSet +10 ; Increase master volume by 10%
SoundSet -10 ; Decrease master volume by 10%
SoundSet, 1, Microphone, mute ; mute the microphone
SoundSet, +1, , mute ; Toggle the master mute (set it to the opposite state)
SoundSet, +20, Master, bass ; Increase bass level by 20%.

if ErrorLevel

    MsgBox, The BASS setting is not supported for MASTER.
```

```
; SOUND CARD ANALYSIS
```

; Use the following script to discover your soundcard's capabilities (component types and control types).

; It displays the results in a simple ListView.

```
SetBatchLines -1
```

```
SplashTextOn,,, Gathering Soundcard Info...
```

; Most of the pure numbers below probably don't exist in any mixer, but they're queried for completeness.

; The numbers correspond to the following items (in order): CUSTOM, BOOLEANMETER, SIGNEDMETER, PEAKMETER,

; UNSIGNEDMETER, BOOLEAN, BUTTON, DECIBELS, SIGNED, UNSIGNED, PERCENT, SLIDER, FADER, SINGLESELECT, MUX,

; MULTIPLESELECT, MIXER, MICROTIME, MILLITIME

```
ControlTypes =
```

```
VOLUME,ONOFF,MUTE,MONO,LOUDNESS,STEREOENH,BASSBOOST,PAN,QSOUNDPAN,BASS,TREBLE,EQUALIZER,0x00000000,  
0x10010000,0x10020000,0x10020001,0x10030000,0x20010000,0x21010000,0x30040  
00,0x30020000,0x30030000,0x30050000,0x40020000,0x50030000,0x70010000,0x70  
010001,0x71010000,0x71010001,0x60030000,0x61030000
```

```
ComponentTypes =
```

```
MASTER,HEADPHONES,DIGITAL,LINE,MICROPHONE,SYNTH,CD,TELEPHONE,PCSPEAKER,  
WAVE,AUX,ANALOG,N/A
```

; Create a ListView and prepare for the main loop:

```
Gui, Add, Listview, w400 h400 vMyListView, Component Type|Control Type|Setting|Mixer  
LV_ModifyCol(4, "Integer")
```

```
SetFormat, Float, 0.2 ; Limit number of decimal places in percentages to two.
```

Loop ; For each mixer number that exists in the system, query its capabilities.

```
{
```

```
CurrMixer := A_Index
```

```

SoundGet, Setting,,, %CurrMixer%

if ErrorLevel = Can't Open Specified Mixer ; Any error other than this indicates that
the mixer exists.

break

; For each component type that exists in this mixer, query its instances and control
types:

Loop, parse, ComponentTypes, `,
{

CurrComponent := A_LoopField

; First check if this component type even exists in the mixer:

SoundGet, Setting, %CurrComponent%,, %CurrMixer%

if ErrorLevel = Mixer Doesn't Support This Component Type

    continue ; Start a new iteration to move on to the next component type.

Loop ; For each instance of this component type, query its control types.

{

CurrInstance := A_Index

; First check if this instance of this instance even exists in the mixer:

SoundGet, Setting, %CurrComponent%:%CurrInstance%,, %CurrMixer%

; Checking for both of the following errors allows this script to run on older
versions:

if ErrorLevel in Mixer Doesn't Have That Many of That Component Type, Invalid
Control Type or Component Type

    break ; No more instances of this component type.

; Get the current setting of each control type that exists in this instance of this
component:

Loop, parse, ControlTypes, `,
{

CurrControl := A_LoopField

SoundGet,
Setting, %CurrComponent%:%CurrInstance%, %CurrControl%, %CurrMixer%

```

```

; Checking for both of the following errors allows this script to run on older
versions:

if ErrorLevel in Component Doesn't Support This Control Type, Invalid
Control Type or Component Type

    continue

if ErrorLevel ; Some other error, which is unexpected so show it in the
results.

    Setting := ErrorLevel

    ComponentString := CurrComponent

    if CurrInstance > 1

        ComponentString = %ComponentString%:%CurrInstance%

        LV_Add("", ComponentString, CurrControl, Setting, CurrMixer)

    } ; For each control type.

} ; For each component instance.

} ; For each component type.

} ; For each mixer.

Loop % LV_GetCount("Col") ; Auto-size each column to fit its contents.

    LV_ModifyCol(A_Index, "AutoHdr")

SplashTextOff

Gui, Show

return

GuiClose:

ExitApp

```

26.6 SoundSetWaveVolume

更改一个音频设备的波形输出音量。

SoundSetWaveVolume, Percent [, DeviceNumber]

参数

Percent	包括 -100 至 100 的百分率数值 (它可以是一个浮点数或者表达式(See 9.))。如果数值以加号或者减号开头, 当前的音量等级 将往上或者往下调指示的数值。否则, 音量将明确地设置为 <i>Percent</i> 指示的等级。
DeviceNumber	如果省略此参数, 其默认为 1 (第一个音频设备), 通常是系统默认的录音和播放设备。可以通过指定一个大于 1 的数字来操作其他的音频设备。

ErrorLevel

如果遇到问题, [ErrorLevel\(See 30.8\)](#) 设为 1, 其他情况为 0。

注意

`SoundSetWaveVolume` 在 Windows Vista 上不起作用(可能会在将来的版本中得到解决)。下面是一种更改整个系统的总音量的方法:

```
Send {Volume_Up} ;将总音量增加 1 个音程 (一般是 5%)。
Send {Volume_Down 3} ;将总音量降低 3 个音程。
Send {Volume_Mute} ;将总音量静音/取消静音。
```

当前的波形输出音量的等级可通过 [SoundGetWaveVolume\(See 26.3\)](#) 来获取。其他设置例如总音量、合成器、麦克风、静音、高音和低音可以用 [SoundSet\(See 26.5\)](#) 和 [SoundGet\(See 26.2\)](#) 进行设定和获取。

与 [SoundSet\(See 26.5\)](#) 不同, 此命令在更改音量等级时会尝试维持现有的声道(例如左和右)平衡。

相关命令

[SoundGetWaveVolume\(See 26.3\)](#), [SoundSet\(See 26.5\)](#), [SoundGet\(See 26.2\)](#), [SoundPlay\(See 26.4\)](#)

示例

```
SoundSetWaveVolume, 50 ;将波形音量设为一半。
SoundSetWaveVolume, -10 ;将当前的等级减少 10 (例如 80 将变成 70)。
SoundSetWaveVolume, +20 ;将当前的等级增加 20。
```

翻译: 天堂之门 menk33@163.com 2008 年 10 月 20 日

27. 字串管理

27.1 FormatTime

Transforms a [YYYYMMDDHH24MISS](#)(See 16.26) timestamp into the specified date/time format.

`FormatTime, OutputVar [, YYYYMMDDHH24MISS(See 16.26), Format]`

参数

OutputVar	The name of the variable in which to store the result.
YYYYMMDD...	Leave this parameter blank to use the current local date and time. Otherwise, specify all or the leading part of a timestamp in the YYYYMMDDHH24MISS (See 16.26) format. If the date and/or time portion of the timestamp is invalid -- such as February 29th of a non-leap year -- the date and/or time will be omitted from <i>OutputVar</i> . Although only years between 1601 and 9999 are supported, a formatted time can still be produced for earlier years as long as the time portion is valid.
Format	If omitted, it defaults to the time followed by the long date, both of which will be formatted according to the current user's locale. For example: 4:55 PM Saturday, November 27, 2004 Otherwise, specify one or more of the date-time formats below, along with any literal spaces and punctuation in between (commas do not need to be escaped; they can be used normally). In the following example, note that M must be capitalized: M/d/yyyy h:mm tt

Date Formats (case sensitive)

d	Day of the month without leading zero (1 - 31)
dd	Day of the month with leading zero (01 - 31)
ddd	Abbreviated name for the day of the week (e.g. Mon) in the current user's language
dddd	Full name for the day of the week (e.g. Monday) in the current user's language
M	Month without leading zero (1 - 12)
MM	Month with leading zero (01 - 12)
MMM	Abbreviated month name (e.g. Jan) in the current user's language
MMMM	Full month name (e.g. January) in the current user's language
y	Year without century, without leading zero (0 - 99)

yy	Year without century, with leading zero (00 - 99)
yyyy	Year with century. For example: 2005
gg	Period/era string for the current user's locale (blank if none)

Time Formats (case sensitive)

h	Hours without leading zero; 12-hour format (1 - 12)
hh	Hours with leading zero; 12-hour format (01 - 12)
H	Hours without leading zero; 24-hour format (0 - 23)
HH	Hours with leading zero; 24-hour format (00- 23)
m	Minutes without leading zero (0 - 59)
mm	Minutes with leading zero (00 - 59)
s	Seconds without leading zero (0 - 59)
ss	Seconds with leading zero (00 - 59)
t	Single character time marker, such as A or P (depends on locale)
tt	Multi-character time marker, such as AM or PM (depends on locale)

The following formats must be used alone; that is, with no other formats or text present in the *Format* parameter. These formats are not case sensitive.

(Blank)	Leave <i>Format</i> blank to produce the time followed by the long date. For example, in some locales it might appear as 4:55 PM Saturday, November 27, 2004
Time	Time representation for the current user's locale, such as 5:26 PM
ShortDate	Short date representation for the current user's locale, such as 02/29/04
LongDate	Long date representation for the current user's locale, such as Friday, April 23, 2004
YearMonth	Year and month format for the current user's locale, such as February, 2004
YDay	Day of the year without leading zeros (1 - 366)
YDay0	Day of the year with leading zeros (001 - 366)
WDay	Day of the week (1 - 7). Sunday is 1.
YWeek	The ISO 8601 full year and week number. For example: 200453. If the week containing January 1st has four or more days in the new year, it is considered week 1. Otherwise, it is the last week of the previous year, and the next week is week 1. Consequently, both January 4th and the first Thursday of January are always in week 1.

Additional Options

The following options can appear inside the `YYYYMMDDHH24MISS` parameter immediately after the timestamp (if there is no timestamp, they may be used alone). In the following example, note the lack of commas between the last four items:

```
FormatTime, OutputVar, 20040228 LSys D1 D4
```

R: Reverse. Have the date come before the time (meaningful only when *Format* is blank).

Ln: If this option is *not* present, the current user's locale is used to format the string. To use the system's locale instead, specify LSys. To use a specific locale, specify the letter L followed by a hexadecimal or decimal locale identifier (LCID). For information on how to construct an LCID, search www.microsoft.com for the following phrase: Locale Identifiers

Dn: Date options. Specify for **n** one of the following numbers:

- 0: Force the default options to be used. This also causes the short date to be in effect.
- 1: Use short date (meaningful only when *Format* is blank; not compatible with 2 and 8).
- 2: Use long date (meaningful only when *Format* is blank; not compatible with 1 and 8).
- 4: Use alternate calendar (if any).
- 8: Use Year-Month format (meaningful only when *Format* is blank; not compatible with 1 and 2).
- 2). Requires Windows 2000 or later.
- 0x10: Add marks for left-to-right reading order layout. Has no effect except on Windows 2000 or later.
- 0x20: Add marks for right-to-left reading order layout. Has no effect except on Windows 2000 or later.
- 0x80000000: Do not obey any overrides the user may have in effect for the system's default date format.
- 0x40000000: Use the system ANSI code page for string translation instead of the locale's code page.

Tn: Time options. Specify for **n** one of the following numbers:

- 0: Force the default options to be used. This also causes minutes and seconds to be shown.
- 1: Omit minutes and seconds.
- 2: Omit seconds.
- 4: Omit time marker (e.g. AM/PM).
- 8: Always use 24-hour time rather than 12-hour time.
- 12: Combination of the above two.
- 0x80000000: Do not obey any overrides the user may have in effect for the system's default time format.
- 0x40000000: Use the system ANSI code page for string translation instead of the locale's code page.

Note: Dn and Tn may be repeated to put more than one option into effect, such as this example:
`FormatTime, OutputVar, 20040228 D2 D4 T1 T8`

Remarks

Letters and numbers that you want to be transcribed literally from *Format* into *OutputVar* should be enclosed in single quotes as in this example: 'Date:' MM/dd/yy 'Time:' hh:mm:ss tt

By contrast, non-alphanumeric characters such as spaces, tabs, linefeeds (` n), slashes, colons, commas, and other punctuation do not need to be enclosed in single quotes. The exception to this is the single quote character itself: to produce it literally, use four consecutive single quotes ("'"), or just two if the quote is already inside an outer pair of quotes.

If *Format* contains date and time elements together, they must not be intermixed. In other words, the string should be dividable into two halves: a time half and a date half. For example, a format string consisting of "hh yyyy mm" would not produce the expected result because it has a date element in between two time elements.

When *Format* contains a numeric day of the month (either d or dd) followed by the full month name (MMMM), the genitive form of the month name is used (if the language has a genitive form).

If *Format* contains more than 2000 characters, *OutputVar* will be made blank.

On a related note, addition and subtraction of dates and times can be performed with [EnvAdd](#)(See 21.1) and [EnvSub](#)(See 21.4).

相关命令

To convert in the reverse direction -- that is, *from* a formatted date/time *to* [YYYYMMDDHH24MISS](#)(See 16.26) format -- see www.autohotkey.com/forum/topic20405.html

See also: [Gui DateTime control](#)(See 19.4), [SetFormat](#)(See 21.10), [Transform](#)(See 21.11), [built-in date and time variables](#)(See 9.), [FileGetTime](#)(See 16.13)

示例

```
FormatTime, TimeString
```

```
MsgBox The current time and date (time first) is %TimeString%.
```

```
FormatTime, TimeString, R
```

```
MsgBox The current time and date (date first) is %TimeString%.
```

```
FormatTime, TimeString,, Time
```

```
MsgBox The current time is %TimeString%.
```

```
FormatTime, TimeString, T12, Time
```

```
MsgBox The current 24-hour time is %TimeString%.
```

```
FormatTime, TimeString,, LongDate
```

```
MsgBox The current date (long format) is %TimeString%.
```

```
FormatTime, TimeString, 20050423220133, dddd MMMM d, yyyy hh:mm:ss tt
```

```
MsgBox The specified date and time, when formatted, is %TimeString%.
```

```
FormatTime, TimeString, 200504, 'Month Name': MMMM`n'Day Name': dddd
```

```
MsgBox %TimeString%
```

```
FormatTime, YearWeek, 20050101, YWeek
```

```
MsgBox January 1st of 2005 is in the following ISO year and week number: %YearWeek%
```

```
; Change the date-time stamp of a file:
```

```
FileSelectFile, FileName, 3,, Pick a file
```

```
if FileName = ; The user didn't pick a file.
```

```
    return
```

```
FileGetTime, FileTime, %FileName%
```

```
FormatTime, FileTime, %FileTime% ; Since the last parameter is omitted, the long date  
and time are retrieved.
```

```
MsgBox The selected file was last modified at %FileTime%.
```

```
; The following function converts the specified number of seconds into the corresponding
```

```
; number of hours, minutes, and seconds (hh:mm:ss format).
```

```
MsgBox % FormatSeconds(7384) ; 7384 = 2 hours + 3 minutes + 4 seconds. It yields:
```

```
2:03:04
```

```
FormatSeconds(NumberOfSeconds) ; Convert the specified number of seconds to
hh:mm:ss format.

{
    time = 19990101 ; *Midnight* of an arbitrary date.

    time += %NumberOfSeconds%, seconds

    FormatTime, mmss, %time%, mm:ss

    return NumberOfSeconds//3600 ":" mmss ; This method is used to support more
than 24 hours worth of sections.

}
```

27.2 If/IfEqual/IfLess/IfGreater

指定一个[变量](#)(See 9.)和一个值比较得出 TRUE 时要执行的命令。当存在多个命令时，将它们括入一个[区块](#)(See 17.2)。

IfEqual, var, value (同: if var = value)
 IfNotEqual, var, value (同: if var <> value) (!= 能用来替代 <>)
 IfGreater, var, value (同: if var > value)
 IfGreaterOrEqual, var, value (同: if var >= value)
 IfLess, var, value (同: if var < value)
 IfLessOrEqual, var, value (同: if var <= value)
 If var ;如果变量的内容为空或为 0，那么它被视为 false。否则，它为 true。

另外可见: [IfInString](#)(See 27.3)

参数

var	变量(See 9.)名称。
value	原义的字串、数字或者变量引用(例如 %var2%)。Value 可以被省略，如果你想要将 var 与一个空字符串(空白的)相比较。

说明

如果 **var** 和 **value** 都是纯粹的数值型，它们将被作为数字比较，而不是作为字符串。否则，它们将作为字符串按字母顺序来比较（就是说，字母的次序将决定 **var** 是否小于、等于或者大于 **value**）。**译注：A<B**

当一个 IF 或者 [ELSE](#)(See 17.5) 拥有多行时，那些行必须被括入大括号。例如：

```
if count <= 0
{
    WinClose Untitled - Notepad
```

```
MsgBox There are no items present.
```

```
}
```

然而，如果仅有一行属于 **IF** 或者 **ELSE**，大括号就是可选的。

如果你使用命令名称类型，那么另一个命令就只能出现在 **IF** 语句的同一行。换句话说，这些是有效的：

```
IfEqual, x, 1, Sleep, 1
```

```
IfGreater, x, 1, EnvAdd, x, 2
```

但是这些是无效的：

```
if x = 1 Sleep 1
```

```
IfGreater, x, 1, x += 2
```

[单个正确的大括号\(OTB\)类型](#)(See 17.11)不能和这些类型的 **IF** 语句一起使用。它只能和表达式类型的 **IF** 语句一起使用。

做一个相关的提示，“[if var \[not\] between LowerBound and UpperBound](#)(See 21.7)” 命令检查一个变量是否在两个 **values** 之间，而 “[if var \[not\] in value1,value2](#)(See 27.4)” 能用来检查一个变量的内容是否在 **values** 列表里存在。

相关命令

[IF \(expression\)](#)(See 17.11), [StringCaseSense](#)(See 27.13), [Assign expression \(:=\)](#)(See 21.12), [if var in/contains MatchList](#)(See 27.4), [if var between](#)(See 21.7), [IfInString](#)(See 27.3), [Blocks](#)(See 17.2), [Else](#)(See 17.5)

范例

```
if counter >= 1
```

```
    Sleep, 10
```

```
if counter >=1 ;如果一个 IF 有多行，将那些行括入大括号:
```

```
{
```

```
    WinClose, Untitled - Notepad
```

```
    Sleep 10
```

```
}
```

```
if MyVar = %MyVar2%
```

```
    MsgBox MyVar 和 MyVar2 的内容是相同的。
```

```

else if MyVar =
{
    MsgBox, 4,, MyVar 是空的/空白的。继续?
    IfMsgBox, No
        Return
}
else if MyVar <> ,
    MsgBox value 在 MyVar 里不是一个逗号。
else
    MsgBox value 在 MyVar 里是一个逗号。

if Done
    MsgBox 变量 Done 即不为空也不为零。

```

翻译: 天堂之门 menk33@163.com 2008 年 11 月 14 日

27.3 IfInString/IfNotInString

检查一个 [variable\(See 9.\)](#)(变量) 是否包含指定的字符串。

IfInString, var, searchString
IfNotInString, var, searchString
Position := [InStr\(See 10.\)](#)(haystack, needle [, caseSensitive?, startPos]) ; 详看 [InStr\(\)](#) 函数([See 10.](#))。

参数

var	variable(See 9.) (变量) 的名称, 为得到一个匹配而搜索它的内容。
searchString	要搜索的字符串。匹配过程不区分大小写, 除非 StringCaseSense(See 27.13) 被启用。

注意

内置变量 [%A_Space%](#)([See 9.](#)) 和 [%A_Tab%](#)([See 9.](#)) 分别地包含了单个空格和单个 tab(制表符)字符, 当单独地搜索这些字符时, 它们可能非常有用。

另一个命令能出现在同一行, 比如下面这个例子一样。换句话说, 它们是等效的:

```

IfInString, MyVar, abc, Gosub, Process1
IfInString, MyVar, abc

```

Gosub, Process1

不过，在同一行上，已命名的命令以外的东西不被支持。例如：

```
IfInString, MyVar, abc, found := true ; 无效。
```

相关命令

[InStr\(\)](#)(See 10.), [RegExMatch\(\)](#)(See 18.12), [StringGetPos](#)(See 27.14), [StringCaseSense](#)(See 27.13), [IfEqual](#)(See 17.10), [if var in/contains MatchList](#)(See 27.4), [if var between](#)(See 21.7), [if var is type](#)(See 21.8), [Blocks](#)(See 17.2), [Else](#)(See 17.5)

示例

```
Haystack = abcdefghijklmnopqrs
Needle = abc
IfInString, Haystack, %Needle%
{
    MsgBox, 字符串已找到。
    return
}
else
    Sleep, 1
```

翻译：天堂之门 menk33@163.com 2008 年 8 月 25 日

27.4 If var [not] in/contains MatchList

If var [not] contains value1, value2, ...

Checks whether a [variable's](#)(See 9.) contents match one of the items in a list.

```
if Var in MatchList
if Var not in MatchList
if Var contains MatchList
if Var not contains MatchList
```

参数

Var	The name of the variable (See 9.) whose contents will be checked. For the "in" operator, an exact match with one of the list items is required. For the "contains" operator, a match occurs more easily: whenever <i>Var</i> contains one of the list items as a substring.
MatchList	<p>A comma-separated list of strings, each of which will be compared to the contents of <i>Var</i> for a match. Any spaces or tabs around the delimiting commas are significant, meaning that they are part of the match string. For example, if <i>MatchList</i> is set to ABC , XYZ then <i>Var</i> must contain either ABC with a trailing space or XYZ with a leading space to cause a match.</p> <p>Two consecutive commas results in a single literal comma. For example, the following would produce a single literal comma at the end of string1: <i>If Var In string1,,,string2</i>. Similarly, the following list contains only a single item with a literal comma inside it: <i>If Var In single,,item</i>. To include a blank item in the list, make the first character a comma as in this example: <i>If Var In ,string1,string2</i> (when using the "contains" operator, a blank item will always result in a match since the empty string is found in all strings).</p> <p>Because the items in <i>MatchList</i> are not treated as individual parameters, the list can be contained entirely within a variable. In fact, all or part of it must be contained in a variable if its length exceeds 16383 since that is the maximum length of any script line. For example, <i>MatchList</i> might consist of %List1%,%List2%,%List3% -- where each of the sublists contains a large list of match phrases.</p> <p>Any single item in the list that is longer than 16384 characters will have those extra characters treated as a new list item. Thus, it is usually best to avoid including such items.</p>

注意

The comparison is always done alphabetically, not numerically. For example, the string "11" would not match the list item "11.0".

The "contains" operator is the same as using [IfInString/IfNotInString](#)(See 27.3) except that multiple search strings are supported (any one of which will cause a match).

"[StringCaseSense](#)(See 27.13) On" can be used to make the comparison case sensitive.

If *MatchList* is long, it can be broken up into several shorter lines by means of a [continuation section](#)(See 8.), which might improve readability and maintainability.

The operators "between", "is", "in", and "contains" are not supported in expressions(See 9.).

相关命令

[if var between](#)(See 21.7), [IfEqual/Greater/Less](#)(See 17.10), [IfInString](#)(See 27.3),
[StringCaseSense](#)(See 27.13), [Blocks](#)(See 17.2), [Else](#)(See 17.5)

示例

```
if var in exe,bat,com
    MsgBox The file extension is an executable type.

if var in 1,2,3,5,7,11 ; Avoid spaces in list.
    MsgBox %var% is a small prime number.

if var contains 1,3 ; Note that it compares the values as strings, not numbers.
    MsgBox Var contains the digit 1 or 3 (Var could be 1, 3, 10, 21, 23, etc.)

if var in %MyItemList%
    MsgBox %var% is in the list.

InputBox, UserInput, Enter YES or NO
if UserInput not in yes,no
    MsgBox Your input is not valid.

WinGetTitle, active_title, A
if active_title contains Address List.txt,Customer List.txt
    MsgBox One of the desired windows is active.
if active_title not contains metapad,Notepad
    MsgBox But the file is not open in either Metapad or Notepad.
```

27.5 If var is [not] type

检查一个[变量](#)(See 9.)的内容是否为数值、大写字母或其他。

```
if var is type
if var is not type
```

参数

<code>var</code>	变量(See 9.)名。
<code>type</code>	请见下面的说明。

注意

支持的 *Types*:

<code>integer</code>	若 <code>var</code> 非空且包含一个没有小数点的纯数字字符串(十进制或十六进制)，则为真。允许开头与结尾有空格和 <code>tab</code> 。字符串可以是加号或减号开头。
<code>float</code>	若 <code>var</code> 非空且包含一个浮点数；也就是说一个包含小数点的纯数字字符串，则为真。允许开头与结尾有空格和 <code>tab</code> 。字符串可以是加号、减号或小数点开头。
<code>number</code>	若 <code>var</code> 包含一个整数或浮点数(两者都在上面解释过了)，则为真。
<code>digit</code>	若 <code>var</code> 为空或者只包含由 0 到 9 组成的数字，则为真。不允许出现后面这样的其他字符：空格、 <code>tab</code> 、加号、减号、小数点、十六进制数和 <code>0x</code> 前缀。
<code>xdigit</code>	十六进制数值：除了还允许包含 A 到 F(大写或小写)这些字符外，其他与 <code>digit</code> 相同。在 v1.0.44.09 及之后版本中，也容许存在 <code>0x</code> 前缀。
<code>alpha</code>	若 <code>var</code> 为空或只包含按字母排序的字符，则为真。若在字符串的任何地方有任何数字、空格、 <code>tab</code> 、标点符号或其他不按字母排序的字符出现，则为假。例如，如果 <code>var</code> 包含一个空格后跟一个字母，它将不被视为 <code>alpha</code> 。
<code>upper</code>	若 <code>var</code> 为空或只包含大写字符，则为真。若在字符串的任何地方有任何数字、空格、 <code>tab</code> 、标点符号或其他未大写的字符出现，则为假。
<code>lower</code>	若 <code>var</code> 为空或只包含小写字符，则为真。若在字符串的任何地方有任何数字、空格、 <code>tab</code> 、标点符号或其他未小写的字符出现，则为假。
<code>alnum</code>	除了还允许包含 0 到 9 的字符外，其他与 <code>alpha</code> 相同。
<code>space</code>	若 <code>var</code> 为空或只包含由下列字符组成的空白字符：空格 (%A_Space% (See 9.))、 <code>tab</code> (%A_Tab% (See 9.)) 或 `t)、换行(`n)、回车(`r)、垂直 <code>tab</code> (`v) 和进纸符(`f)，则为真。
<code>time</code>	<p>若 <code>var</code> 包含一个有效的日期时间标记，则为真，日期时间标记可以全部是 YYYYMMDDHH24MISS(See 16.26) 格式或者仅仅开头部分是。例如，一个 4 位的数字字符串比如 2004 被视为有效。使用 StringLen(See 27.17) 可以确定是否存在额外的时间部分。</p> <p>小于 1601 的年份将被视为无效，因为操作系统通常不支持它们。被视为有效的最大的年份是 9999。</p> <p>可以使用单词 DATE 来代替 TIME，结果相同。</p>

注意：**表达式**(See 9.)中不支持 "between", "is", "in" 和 "contains" 运算符。

相关命令

[%A_YYYY%](#)(See 9.), [SetFormat](#)(See 21.10), [FileGetTime](#)(See 16.13), [IfEqual](#)(See 17.10), [if var in/contains MatchList](#)(See 27.4), [if var between](#)(See 21.7), [StringLen](#)(See 27.17), [IfInString](#)(See 27.3), [StringUpper](#)(See 27.18), [EnvAdd](#)(See 21.1), [Blocks](#)(See 17.2), [Else](#)(See 17.5)

示例

```
if var is float
    MsgBox, %var% 是一个浮点数。
else if var is integer
    MsgBox, %var% 是一个整数。
if var is time
    MsgBox, %var% 也是一个有效的日期时间。
```

翻译: wz520 修正: 天堂之门 menk33@163.com 2008 年 11 月 24 日

27.6 InStr()

InStr(Haystack, Needle [, CaseSensitive = false, StartingPos = 1]): 返回 *Haystack* 字符串中首个匹配的 *Needle* 字符串的位置。与 [StringGetPos](#) 不同, 第一个字符的位置是 1; 这是因为 0 同义于 "false", 会使其成为一个直观的"未找到"的指示。如果参数 *CaseSensitive* 被省略或者是 *false*, 则搜索将不区分大小写(不区分的模式取决于 [StringCaseSense](#) 命令); 否则就得精确地匹配大小写。如果省略 *StartingPos*, 其默认为 1(*Haystack* 字符串的起点)。否则, 指定 2 从 *Haystack* 的第二个字符开始搜索, 3 从第三个开始, 等等。如果 *StartingPos* 超出 *Haystack* 的长度, 那么函数将返回 0。如果 *StartingPos* 为 0, 那么搜索将逆序执行(从右到左)以便找到在最右边匹配的结果。不管 *StartingPos* 的值是多少, 返回的位置将始终相对于 *Haystack* 的首个字符而言。例如: 在"123abc789"中"abc"的位置永远是 4。相关链接: [RegExMatch\(\)](#), [IfInString](#) 和 [StringGetPos](#)。（译注: 例如 InStr("123abc789","abc",false,1))

RegExMatch(Haystack, NeedleRegEx [, UnquotedOutputVar = "", StartingPos = 1]):
请看 [RegExMatch\(\)](#)。

RegExReplace(Haystack, NeedleRegEx [, Replacement = "", OutputVarCount = "", Limit = -1, StartingPos = 1]): 请看 [RegExReplace\(\)](#)。

27.7 Loop (parse a string)

根据分隔符, 从一个字符串中循环获取子字符串(片段), 一次获取一段。

[Loop](#), [Parse](#), [InputVar](#) [, Delimiters, OmitChars]

参数

Parse	这个参数必须使用单词 PARSE ，而且和其它的循环命令不同，这里不能引用变量的值。
InputVar	需要被解析的变量。不要用百分号将变量名包起来，除非该变量的 内容 就是需要被解析的变量名。
Delimiters	<p>分隔符。如果这个参数为空或省略，循环将依次获取 <i>InputVar</i> 中的每个字符。</p> <p>如果这个参数使用 CSV，<i>InputVar</i> 将按照标准的逗号分隔格式进行解析。这里有一个 MS Excel 生成的 CSV 文件的示例: "first field",SecondField,"the word ""special"" is quoted literally",,"last field, has literal comma"</p> <p>其它情况下，<i>Delimiters</i> 中可以包含一个或多个字符(区分大小写)，每一个字符都会作为分隔符，用来将 <i>InputVar</i> 中的字符串分隔为相应的子字符串。</p> <p>分隔符将不会出现在被分隔出来子字符串中。另外，如果在 <i>InputVar</i> 中两个分隔符之间没有任何内容，则相应解析出来的子字符串为空。</p> <p>例如：`，(经过转义的逗号)会按照字符串中逗号的位置将字符串分隔为多个子字符串。相似的，%A_Tab%%A_Space% 则会按照字符串中空格和制表符的位置将字符串分隔。</p> <p>如果需要使用一个字符串作为分隔符，可以先使用 StringReplace(See 27.20) 命令将 <i>InputVar</i> 中相应的字符串替换为一个在 <i>InputVar</i> 中从未使用的字符，例如这些特殊字符：¤¥!§©¤®µ¶，然后再使用这些特殊字符作为分隔符进行解析。参考下面这个例子，使用字符串
 作为分隔符：</p> <pre>StringReplace, NewHTML, HTMLString,
, ¢, All Loop, parse, NewHTML, ¢ ; 使用 ¢ 解析字符串。 { ... }</pre>
OmitChars	<p>忽略字符。这个可选参数中可以包含一个或多个字符(区分大小写)，这些字符会从解析出来的子字符串的开头和结尾部分移除。例如，<i>OmitChars</i> 参数使用 %A_Space%%A_Tab%，则如果解析出来的子字符串的开头或结尾部分有空格或制表符的话，这些空格和制表符会被删除(在子字符串中间的空格和制表符不会被删除)。</p> <p>如果 <i>Delimiters</i> 参数留空，<i>OmitChars</i> 参数指定的字符将不会出现在循环中。</p> <p>和其它大多数命令最后一个参数不同，在 <i>OmitChars</i> 参数中，逗号必须进行转义(`,)。</p>

注意

当需要对一个字符串的各个片段进行逐个操作的时候，经常使用字符串解析循环。字符串解析循环也比 [StringSplit\(See 27.21\)](#) 命令更省内存(因为 `StringSplit` 命令创建了一个持久的数组)，而且在大多数情况下更加易用。

内置变量 **A_LoopField** 存在于任何一种解析循环中，它包含了从 *InputVar* 解析出来的当前子字符串(或片段)的内容。如果一个内部解析循环被装在一个外部解析循环中，则最里层循环中解析出来的片段将享有优先。

InputVar 或它的片段的大小没有限制。另外，即使在循环过程中 *InputVar* 的内容改变了，`Loop` 也会“无视”这些改变，因为它操作的是原始内容的临时副本。

要在解析之前将字符串中的片段进行排序，请用 [Sort\(See 27.12\)](#) 命令。

请看 [Loop\(See 17.12\)](#) 命令获取 [Blocks\(See 17.2\)](#), [Break\(See 17.3\)](#), [Continue\(See 17.4\)](#) 以及内置变量 **A_Index** (存在于任何类型的循环中) 的相关信息。

相关命令

[StringSplit\(See 27.21\)](#), [file-reading loop\(See 16.32\)](#), [Loop\(See 17.12\)](#), [Break\(See 17.3\)](#), [Continue\(See 17.4\)](#), [Blocks\(See 17.2\)](#), [Sort\(See 27.12\)](#), [FileSetAttrib\(See 16.25\)](#), [FileSetTime\(See 16.26\)](#)

示例

```
;示例 1:

Colors = red,green,blue
Loop, parse, Colors, `,
{
    MsgBox, 第 %A_Index% 种颜色是 %A_LoopField%。
}
```

;示例 2：逐行读取一个变量中的内容(类似文件读取循环([See 16.32](#))。)

;使用 [FileRead\(See 16.19\)](#) 命令可以将一个文件读取到一个变量中：

```
Loop, parse, FileContents, `n, `r ;将 `n 写在 `r 前面以保证 Windows 和 Unix 这两者的文件都能被正常解析。
{
    MsgBox, 4, , 第 %A_Index% 行是 %A_LoopField%。`n`n继续?
    IfMsgBox, No, break
}
```

;示例 3: 这个示例和上面的一样, 只不过解析对象是剪贴板。

;当剪贴板中包含文件时很有用, 例如从一个打开的资源管理器窗口中复制的文件(程序会自动将这些文件转换为它们完整的路径):

```
Loop, parse, clipboard, `n, `r
{
    MsgBox, 4, , 第 %A_Index% 个文件是 %A_LoopField%。`n`n继续?
    IfMsgBox, No, break
}
```

;示例 4: 解析一个逗号分隔(CSV)格式的文件:

```
Loop, read, C:\Database Export.csv
{
    LineNumber = %A_Index%
    Loop, parse, A_LoopReadLine, CSV
    {
        MsgBox, 4, , 片段 %LineNumber%-%A_Index% 是: `n%A_LoopField%`n`n继续?
        IfMsgBox, No
            return
    }
}
```

翻译: okey3m 修正: 天堂之门 menk33@163.com 2009 年 1 月 7 日

27.8 RegExMatch()

确定一个字符串中所包含的匹配模式 (即: 正则表达式)。

FoundPos := RegExMatch(Haystack, NeedleRegEx [, UnquotedOutputVar = "", StartingPosition = 1])

参数

FoundPos	RegExMatch() 返回从源字符串 <i>Haystack</i> 最左边开始找到的第一个匹配
----------	--

	<p><i>NeedleRegEx</i> 模式的位置。首字符的位置是 1。要是在字符串中没有找到匹配模式的时候就会返回 0。如果出现了错误（比如：正则表达式 <i>NeedleRegEx</i> 的语法错误），它将会返回一个空的字符串和 <i>ErrorLevel</i>。<i>ErrorLevel</i> 将是 下面 值中的一个，包括 0。</p>
<i>Haystack</i>	源字符串。
<i>NeedleRegEx</i>	这种匹配模式是和 Perl 兼容的正则表达式 (PCRE)。如果有必要的话，可以在正则表达式前加上 选项 ，并以 ")" 结束。例如：这个匹配模式 " <i>i</i>)abc.*123" 将匹配不区分大小写的 "abc"，中间可以是任何字符，并以 123 结尾的模式进行搜索。要是没有 选项 ，这个 ")" 是可选的，例如：")abc" 就等同于 "abc"。
<i>UnquotedOutputVar</i>	<p>形式 1 (默认): <i>OutputVar</i> 是 unquoted 的一个变量名，它将存储 <i>Haystack</i> 中符合匹配模式的那一部分字符串。如果没有找到符合这个模式的子串（那么，<i>RegExMatch()</i> 将返回 0），而输出变量和所有的数组元素将被置为空。</p> <p>如果在 <i>NeedleRegEx</i> 中包含 捕捉子模式(See 30.14)，那么它们将存储在以 <i>OutputVar</i> 为基变量的 数组(See 30.5) 里。例如：输出的变量名是 <i>Match</i>，那它第一个能匹配子模式的子串将存储在 <i>Match1</i>，第二个存储在 <i>Match2</i>，等等。有一个例外是 命名子模式：它用名字代替了数字的形式来存储。例如：这种模式 <i>(?P<Year>\d{4})</i>，会把输出的子串存储在 <i>MatchYear</i> 里面。如果没有任何字符串（或者此函数返回的值是 0），那么这一输出变量置为空。</p> <p>在 函数(See 10.) 中创建一个全局变量的数组来替代局部变量，就要在使用前 声明(See 10.) 数组名（比如刚才的 <i>Match</i>）为全局变量。</p> <p>形式 2 (位置 - 长度): 若以 P 开头的正则表达式，比如，"<i>P</i>)abc.*123"，则完全匹配正则表达式的字符串的 长度 将存储在 <i>OutputVar</i> 里（没有匹配到，长度就是 0）。如果现在使用了 捕捉子模式(See 30.14)，就会有两个数组 <i>OutputVarPos</i> 和 <i>OutputVarLen</i> 分别存储匹配到的字符串的位置和长度。例如：若基变量名为 <i>Match</i>，则符合模式的第一个子串的位置存储在 <i>MatchPos1</i>，长度存储在 <i>MatchLen1</i>（没到匹配到任何子串则全部返回 0，函数值也会为 0）。仍有一个例外，就是 命名子模式：它将存储名用名字代替了数字（比如，<i>MatchPosYear</i> 和 <i>MatchLenYear</i>）。</p>
<i>StartingPosition</i>	<p>如果省略 <i>StartingPosition</i> 的值，它会默认为 1（从源字符串 <i>Haystack</i> 的第一个字符开始）。另外，设置为 2 则以第二个字符开始，值为 3 就从第三个开始，依次类推。如果 <i>StartingPosition</i> 的值超出了 <i>Haystack</i> 的长度范围，则函数返回为 0，变量值为空。</p> <p>如果 <i>StartingPosition</i> 的值小于 1，函数会认为是从 <i>Haystack</i> 的末尾处开始。例如：0 就是倒数一个字符的位置，-1 就是倒数第二个字符，依次类推。如果 <i>StartingPosition</i> 超过了 <i>Haystack</i> 最左边的位置，它就会搜索整个 <i>Haystack</i>。</p> <p>无论 <i>StartingPosition</i> 的值是什么，函数返回的值都是根据 <i>Haystack</i> 的第一个字符所确定的。例如："123abc789" 中 "abc" 的位置总是 4。</p>

ErrorLevel

[ErrorLevel](#)(See 30.8) 的值是下面中的一个:

- 0, 表示没发现错误。
- 一个有如下格式的字符串: *Complie error N at offset M: description.* 在这个字符串里, *N* 是 PCRE (Perl 兼容正则表达式) 的错误值, *M* 是正则表达式中出现错误的位置, *description* 是描述这个错误的文本信息。
- 一个负数, 是指在 执行 正则表达式过程中出现的错误。尽管这种错误非常罕见, 如这些情况就容易出现这种错误: “太多的空匹配” (-22), “递归太深” (-21), “太到匹配极限” (-8)。如果出现了这些错误, 就试着重新写更严格的正则表达式, 如用 ?, + 或者用有限制的方式, 如 {0,3} 来取代 *。

选项 (区分大小写)

在正则表达式之前加上 0 个或者多个选项并以右括号结束。例如: 模式 "im)abc" 将搜索多行中不区分大小写的 abc , 若没有选项可以省略圆括号。尽管这将原来的语法打断, 但是它不需要新的前分割符 (比如右斜杠), 因此没有必要转换模式内的分割符。另外, 由于使用了选项, 函数性能有所提升。

i	不区分大小写。
m	<p>多行是指由多个单行 (有换行符) 所组成的一个集合 (它包括换行符)。但在下列情况会有所改变:</p> <ol style="list-style-type: none"> 1) 弯折符号 (^) 在所有行内匹配 -- 总是从源字符串开头的地方开始匹配 (但不是匹配源字符串 <i>Haystack</i> 中换行 靠后 的地方)。 2) 美元符号 (\$) 在源字符串中任意换行符之前匹配 (也就是说, 它总是匹配每行靠近结尾的地方)。 <p>例如: 源字符串是 "xyz`r`nabc" 中的 abc 要用模式 "m)^abc\$" 来精确匹配。</p> <p>如果选项是 "D", 它就将会忽略当前的 "m"。</p>
s	该选项会使点(.) 能匹配一行内所有的字符 (通常不会换行符后的)。如果换行符是默认的 CRLF (`r`n) , 就要用两个点去匹配它 (而不是一个点)。无论这个选项如何, [^a] 总是匹配换行符。
x	忽略模式中空白字符类。如字符 `n 和 `t 这类的字符将被在正则表达式中的忽略, 这些字符都是出现在正则表达式中 (与此相反, 使用 \n 和 \t 之类的字符就不会被正则表达式忽略)。x 选项也是一个复杂的模式。然而, 这种模式必须遵循这个条件: 只能 应用在数据字符中; 空白字符也许不会出现特殊的序列, 如 ?(。
A	强制将匹配模式固定; 也就是说, 它只能从源字符串的第一个字符开始匹配。大多数情况下, 和 "^" 的功能一样。
D	强制用 (\$) 的方式来从靠近源字符串末尾的地方开始匹配, 即使源字符串以的换行符结尾的。没有这个选项时, \$ 就只能匹配换行符 (如果有的话) 前的字符了。注意: 这个选项会让 "m" 选项不起作用。
J	允许重复 命名子模式 。这是一个很有用的模式, 它在一组相同命名子模式下也能够匹配。注意: 如果有多个实例命名去匹配字符串的话, 就只有最左边的那个命名才会被存储 (变量名不区分大小写)。

U	非贪婪的。只有在绝对必要的情况下才会使用量词 <code>*+?{}</code> ，并将剩下的部分提供给下一个模式。当 "U" 选项无效时，可以通过一个问号作为量词进行非贪婪匹配。相反地，当 "U" 选项有效，问号就会进行贪婪搜索。
X	PCRE 扩展。它使 PCRE 不完全符合 Perl 的正则表达式。目前，仅仅只有一个这样的特点，那就是在模式中的任何的反斜杠后加一个字符就没有特殊的意义并导致匹配失败和设置 <code>ErrorLevel</code> 相应的值。这个选项帮助保留未使用的反斜杠序列供以后使用。没有这个选项，反斜杠后的字符就会看成一般的字符而没有特殊的意义（例如： <code>\g</code> 和 <code>g</code> 都被认为是字母 <code>g</code> ）。不论此选项怎样，非字母反斜杠序列都没有特别的意义，它将总是认为是普通的字符（例如： <code>\V</code> 和 <code>/</code> 都被认为是 <code>/</code> ）。
P	位置模式。它会使函数 <code>RegExMatch()</code> 产生匹配到的子串的位置和长度。更详细的了解，请看上面的 输出变量 。
S	研究模式可以提高函数的性能。当匹配模式比较复杂或者被重复执行很多次的情况下，这个选项很有用。它会让匹配模式存储在高速缓存中，以便下次使用，通过这种方法来提高了正则表达式的性能。
<code>\n</code>	替换默认的换行符（`r`n）为符合 UNIX 系统的标准的（`n）。所选择的换行符会影响到 anchors (^ and \$) 和 dot/period pattern 。
<code>\r</code>	替换默认的换行符（`r`n）为符合 Windwos 标准的（`r）。
<code>\a</code>	在 1.0.46.06+ 的版本中，`a 看作是换行符，也就是指这些符号，`r, `n, `r`n, `v/VT/vertical tab/chr(0xB), `f/FF/formfeed/chr(0xC), and NEL/next-line/chr(0x85)。在 1.0.47.05+ 的版本中，换行符被限制为 CR, LF 和 CRLF 为 (*ANYCRLF)，在选项后的正则表达式之前大写；例如：im)(*ANYCRLF)^abc\$

注意：可用空格和 Tab 随意的隔开每个选项。

性能

如果从一个较长的字符串中搜索一个简单的子串，请使用 [InStr\(\)](#)(See 10.) 函数，因为它比 `RegExMatch()` 更有效率。

为了提高性能，它将最近使用的 100 个正则表达式缓存在内存里（在编译的时候）。

当一个正则表达式重复使用的情况下，使用 **S 选项** 来提高性能。（比如在 loop 循环中）

注意

一个命名子模式都有一个名字，如上文模式 `(?P<Year>\d{4})` 中的 `Year`。这些名字可能包括多达 32 个字母、数字和下划线。虽然这些命名子模式都是通过数字在对正则表达式本身的操作（如：[向后引用](#)(See 18.13)），结果是仅仅用名字存储 [输出数组](#)，而不是用数字。例如 "Year" 是第一个名字，把匹配到的子串存储在 `OutputVarYear` 里，但是 `OutputVar1` 的值是没有改变（它将保留以前的值，如果说有的话）。如果是 [非命名子模式](#)(See 30.14) 的 "Year"，它将匹配到的子串存储在 `OutputVar2` 里，而不是 `OutputVar1` 里。

大多数的字符串（像 abc123）就能用一般的字符来匹配。而匹配像 `\.*?+[{}|()^$` 这样被保护的字符就要在其前面加是一个反斜杠。比如：`\.` 代表一个点，`\\\` 代表一个反斜杠。转义符可避免使用 `\Q...\E`。比如 `\QLiteral Text\E`。

在正则表达式中，一些特殊的字符，如制表符和换行符就用重音符号 (`) 或反斜杠 (\) 来表示。例如：`t is the same as \t。

如学习正则表达式的基础（或者重新记忆一下正则表达式的语法），请看 [RegEx Quick Reference\(See 30.14\)](#)。

AutoHotKey 所使用的正则表达式是来自于 www.pcre.org 的兼容 Perl 语言的正则表达式。

相关命令

[RegExReplace\(\)\(See 18.13\)](#), [RegEx Quick Reference\(See 30.14\)](#), [InStr\(\)\(See 10.\)](#),
[IfInString\(See 27.3\)](#), [StringGetPos\(See 27.14\)](#), [SubStr\(\)\(See 10.\)](#), [SetTitleMatchMode RegEx\(See 28.8\)](#), [Global matching and Grep \(forum link\)](#)

Common sources of text data: [FileRead\(See 16.19\)](#), [UrlDownloadToFile\(See 22.24\)](#),
[Clipboard\(See 30.6\)](#), [GUI Edit controls\(See 19.4\)](#)

示例

```
FoundPos := RegExMatch("xxxabc123xyz", "abc.*xyz") ; 返回值 4，它就是匹配到的位置。
```

```
FoundPos := RegExMatch("abc123123", "123$") ; 返回值 7，因为 $ 要求从靠近最后的字符处开始匹配。
```

```
FoundPos := RegExMatch("abc123", "i)^ABC") ; 返回值 1，因为是要求从第一个字符处开始匹配，而且不区分大小写。
```

```
FoundPos := RegExMatch("abcXYZ123", "abc(.*)123", SubPat) ; 返回 1 并且 SubPat1 的值是 "XYZ"。
```

```
FoundPos := RegExMatch("abc123abc456", "abc\d+", "", 2) ; 返回值 7，由于它是从第二个字符开始匹配的。
```

; 更多正则表达式的例子，请看 [RegEx Quick Reference\(See 30.14\)](#).

27.9 RegExReplace()

Replaces occurrences of a pattern (regular expression) inside a string.

```
NewStr := RegExReplace(Haystack, NeedleRegEx [, Replacement = "", OutputVarCount = "", Limit = -1, StartingPosition = 1])
```

参数

NewStr	RegExReplace() returns a version of <i>Haystack</i> whose contents have been replaced by the operation. If no replacements are needed, <i>Haystack</i> is returned unaltered. If an error occurs (such as a syntax error inside
--------	---

	<i>NeedleRegEx</i>), <i>Haystack</i> is returned unaltered (except in versions prior to 1.0.46.06, which return "") and ErrorLevel is set to one of the values below instead of 0.
Haystack	The string whose content is searched and replaced.
NeedleRegEx	The pattern to search for, which is a Perl-compatible regular expression (PCRE). The pattern's options (See 18.12) (if any) must be included at the beginning of the string followed by an close-parenthesis. For example, the pattern " <i>i</i>)abc.*123" would turn on the case-insensitive option and search for "abc", followed by zero or more occurrences of any character, followed by "123". If there are no options, the ")" is optional; for example, ")abc" is equivalent to "abc".
Replacement	<p>The string to be substituted for each match, which is plain text (not a regular expression). It may include backreferences like \$1, which brings in the substring from <i>Haystack</i> that matched the first subpattern(See 30.14). The simplest backreferences are \$0 through \$9, where \$0 is the substring that matched the entire pattern, \$1 is the substring that matched the first subpattern, \$2 is the second, and so on. For backreferences above 9 (and optionally those below 9), enclose the number in braces; e.g. \${10}, \${11}, and so on. For named subpatterns(See 18.12), enclose the name in braces; e.g. \${SubpatternName}. To specify a literal \$, use \$\$ (this is the only character that needs such special treatment; backslashes are never needed to escape anything).</p> <p>To convert the case of a subpattern, follow the \$ with one of the following characters: U or u (uppercase), L or l (lowercase), T or t (title case, in which the first letter of each word is capitalized but all others are made lowercase). For example, both \$U1 and \$U{1} transcribe an uppercase version of the first subpattern.</p> <p>Nonexistent backreferences and those that did not match anything in <i>Haystack</i> -- such as one of the subpatterns in <i>(abc) (xyz)</i> -- are transcribed as empty strings.</p>
OutputVarCount	The unquoted name of a variable in which to store the number of replacements that occurred (0 if none).
Limit	If <i>Limit</i> is omitted, it defaults to -1, which replaces all occurrences of the pattern found in <i>Haystack</i> . Otherwise, specify the maximum number of replacements to allow. The part of <i>Haystack</i> to the right of the last replacement is left unchanged.
StartingPosition	If <i>StartingPosition</i> is omitted, it defaults to 1 (the beginning of <i>Haystack</i>). Otherwise, specify 2 to start at the second character, 3 to start at the third, and so on. If <i>StartingPosition</i> is beyond the length of <i>Haystack</i> , the search

starts at the empty string that lies at the end of *Haystack* (which typically results in no replacements).

If *StartPosition* is less than 1, it is considered to be an offset from the end of *Haystack*. For example, 0 starts at the last character and -1 starts at the next-to-last character. If *StartPosition* tries to go beyond the left end of *Haystack*, all of *Haystack* is searched.

Regardless of the value of *StartPosition*, the return value is always a complete copy of *Haystack* -- the only difference is that more of its left side might be unaltered compared to what would have happened with a *StartPosition* of 1.

ErrorLevel

[ErrorLevel](#)(See 30.8) is set to one of the following:

- 0, which means that no error occurred.
- A string in the following form: *Compile error N at offset M: description*. In that string, *N* is the PCRE error number, *M* is the position of the offending character inside the regular expression, and *description* is the text describing the error.
- A negative number, which means an error occurred during the *execution* of the regular expression. Although such errors are rare, the ones most likely to occur are "too many possible empty-string matches" (-22), "recursion too deep" (-21), and "reached match limit" (-8). If these happen, try to redesign the pattern to be more restrictive, such as replacing each * with a ?, +, or a limit like {0,3} wherever feasible.

Options

See [Options](#)(See 18.12) for modifiers such as "i)abc", which turns off case-sensitivity in the pattern "abc".

Performance

To replace simple substrings, use [StringReplace](#)(See 27.20) because it is faster than `RegExReplace()`.

If you know what the maximum number of replacements will be, specifying that for the *Limit* parameter improves performance because the search can be stopped early (this might also reduce the memory load on the system during the operation). For example, if you know there can be only one match near the beginning of a large string, specify a limit of 1.

To improve performance, the 100 most recently used regular expressions are kept cached in memory (in compiled form).

The [study option \(S\)](#)(See 18.12) can sometimes improve the performance of a regular expression that is used many times (such as in a loop).

注意

Most characters like abc123 can be used literally inside a regular expression. However, the characters `\.*?+[{}|()^$` must be preceded by a backslash to be seen as literal. For example, `\.` is a literal period and `\\"` is a literal backslash. Escaping can be avoided by using `\Q...\\E`. For example: `\QLiteral Text\\E`.

Within a regular expression, special characters such as tab and newline can be escaped with either an accent (`) or a backslash (\). For example, `t is the same as \t.

To learn the basics of regular expressions (or refresh your memory of pattern syntax), see the [RegEx Quick Reference](#)(See 30.14).

相关命令

[RegExMatch\(\)](#)(See 18.12), [RegEx Quick Reference](#)(See 30.14), [StringReplace](#)(See 27.20), [InStr\(\)](#)(See 10.)

Common sources of text data: [FileRead](#)(See 16.19), [UrlDownloadToFile](#)(See 22.24), [Clipboard](#)(See 30.6), [GUI Edit controls](#)(See 19.4)

示例

```
NewStr := RegExReplace("abc123123", "123$", "xyz") ; Returns "abc123xyz" because  
the $ allows a match only at the end.
```

```
NewStr := RegExReplace("abc123", "i)^ABC") ; Returns "123" because a match was  
achieved via the case-insensitive option.
```

```
NewStr := RegExReplace("abcXYZ123", "abc(.*)123", "aaa$1zzz") ; Returns  
"aaaXYZzzz" by means of the $1 backreference.
```

```
NewStr := RegExReplace("abc123abc456", "abc\d+", "", ReplacementCount) ; Returns  
"" and stores 2 in ReplacementCount.
```

; For general RegEx examples, see the [RegEx Quick Reference](#)(See 30.14).

27.10 SetEnv (var=value)

为一个 [变量](#)(See 9.) 赋值。

`SetEnv, Var, Value`

`Var = Value`

参数

Var	变量(See 9.) 的名称。变量中存储了指定的 值。
Value	需要存储的值，可以是字符串或数字。如果字符串太长，可以使用 continuation section/字符串分段(See 8.) 的方法将它分为几个短小的段落，这样可以增加代码的可读性和可维护性。

注意

默认情况下，任何在 值 的开头和结尾处的空格和 Tab 是被忽略的，不会存入变量中。要避免这种情况，可以在赋值之前使用 [AutoTrim Off \(See 22.3\)](#) 命令。但是，Tab 在任何情况下都是被忽略的。要想保留 Tab，可以使用内置变量 [%A_Tab%\(See 9.\)](#)。

命令的名字“SetEnv”容易使人产生误解，保留它仅仅是为了兼容 AutoIt v2。和 AutoIt v2 不同，AutoHotkey 并不将变量存储在系统环境中，因为这样性能比较差而且系统限制这样的变量最大为 32K。使用 [EnvSet\(See 15.3\)](#) 命令来设置一个 环境变量(See 9.)，而不是 SetEnv。

可以设置变量为空来释放它所占用的内存。例如 var =。

这个命令，以及所有接收 *OutputVar* 参数的命令都可以创建一个 数组(See 30.5)，只要让 *OutputVar* 包含一个对其它变量的引用就行了。例如 array%i% = 123。查看 数组(See 30.5) 获取更多信息。

相关命令

[AutoTrim\(See 22.3\)](#), [EnvSet\(See 15.3\)](#), [EnvAdd\(See 21.1\)](#), [EnvSub\(See 21.4\)](#), [EnvMult\(See 21.3\)](#), [EnvDiv\(See 21.2\)](#), [If\(See 17.10\)](#), [Arrays\(See 30.5\)](#)

示例

```
Var1 = This is a string.  
Color2 = 450  
Color3 = %Var1%  
Array%i% = %A_TICKCOUNT%
```

27.11 SetFormat

设置数学运算得到的整数或浮点数的格式。

`SetFormat, NumberType, Format`

参数

NumberType	必须是 INTEGER/整数 或者 FLOAT/浮点数。
Format	上一个参数 <i>NumberType</i> 使用 INTEGER 的话，这个参数必须是表示十六进制的 HEX

(可以简写为 H)，或者表示十进制的 D。十六进制的数字全部是以 0x 开头的（例如 0xFF）。

上一个参数 *NumberType* 使用 FLOAT 的话，这个参数的格式为 *总宽度.小数位数*（例如 0.6）。在 v1.0.46.11 以上的版本中，可以在后面添加字母“e”表示使用科学计数法显示结果，例如 0.6e 或者 0.6E（使用大写字母 E 的话在结果中显示的就是大写字母 E 而不是小写字母 e）。注意：在 AutoHotkey 1.x 版本中，使用科学计数法显示的时候，这个参数必须包含小数点，例如 1.0e1 是正确的，而 1e1 是错误的。

总宽度 一般使用 0，表示计算结果不需要使用空格或 0 来填充。如果 *总宽度* 大于实际宽度，将会使用空格或 0（参看下面）来填充至指定宽度。注意：*总宽度* 包括小数点。

小数位数 表示要显示的小数部分宽度（超出部分的四舍五入）。如果留空或使用 0，结果中将不会显示小数点以及小数部分，也就是说，结果会以整数的方式存储，而不是浮点数。

填充：如果 *总宽度* 大于实际宽度，结果中将会使用空格来填充左边缺少的宽度，也就是说，数字将会右对齐显示。要使数字左对齐显示，在 *总宽度* 中使用负数。要使用 0 来填充，在 *总宽度* 中加前缀 0（例如 06.2）。

注意

如果在脚本中未使用这个命令，整数默认使用十进制，浮点数默认使用 *总宽度.小数位数* = 0.6。每一个新运行的 Thread/线程(See 30.19)（例如一个 hotkey/热键(See 4.)，custom menu item/自定义菜单(See 22.13)，或 timed/定时器(See 17.21) 事件）会将该命令的设置重置为默认值。要更改该命令的默认值，可以将该命令放在脚本的自动执行区域（脚本的顶部）。

你可以检测一个变量是否存储了数字类型的值，使用 if var is number/integer/float(See 21.8)。

一个包含整数的变量与 0 相加，可以强制转换为当前设置的整数格式（十六进制或十进制），例如：Var += 0。类似的，一个包含浮点数（或整数，需要的话）的变量与 0.0 相加，可以强制转换为当前设置的浮点数格式，例如：Var += 0.0。如果需要将一个浮点数转换为一个整数，使用 Round()(See 10.)，Ceil()(See 10.)，或者 Floor()(See 10.) 会更加方便。

内置变量 **A_FormatInteger** 存储了当前整数格式的设置（H 或 D）。**A_FormatFloat** 存储了当前浮点数格式的设置。

要对整数使用空格或 0 来填充至指定宽度的话，参考下面的例子：

```
Var := "           ".Var ; 引号中包含了 10 个空格。要使用 0 来填充的话，用 0 替换空格。
```

```
StringRight, Var, Var, 10 ; 使用空格将变量 Var 中的数字填充至 10 位宽度。
```

```
Var := SubStr("           ".Var, -9) ; 上面两行代码的整合。
```

浮点格式

因为 AutoHotkey 将所有的变量当作字符串来存储，所以存储的浮点数的精度是由 *小数位数* 决定。也就是说，一旦浮点数存入了变量中或者变量进行了舍入操作，丧失的精度就再也不能还原了，除非再进行一次相同的运算，并设置更大的 *小数位数*。

在将一个浮点数变量赋值给另外一个变量的时候，浮点数前面的空格填充会被删除掉。要避免这种情况，使用 [冒号-等号 运算符 \(:=\)](#)(See 21.12) 或者 [AutoTrim](#)(See 22.3)。

要使当前设置的浮点数格式在数学运算中生效，例如 [addition](#)(See 21.1)，运算式中至少要有一个数字包含小数点。

一个包含整数的变量可以通过与 `0.0` 相加转换为浮点数，例如 `Var += 0.0`。但是，如果当前的浮点数格式中没有指定 小数位数，使用这个来代替：

```
if Var is integer  
    Var = %Var%.0
```

十六进制格式

十六进制数字以 `0x` 为前缀（例如 `0xA9`）。它可以使用在任何一个需要数字做参数的地方。例如，`Sleep 0xFF` 等同于 `Sleep 255`，并且这与当前 `SetFormat` 中对整数格式的设置无关。

要将一个整数从十进制转换为十六进制，参考这个例子（也可以反过来转换）：

```
SetFormat, integer, hex  
  
VariableContainingAnInteger += 0  
  
SetFormat, integer, d
```

相关命令

[Expression assignment \(:=\)](#)(See 21.12), [EnvAdd](#)(See 21.1), [EnvSub](#)(See 21.4), [EnvMult](#)(See 21.3), [EnvDiv](#)(See 21.2), [AutoTrim](#)(See 22.3), [if var is type](#)(See 21.8)

示例

```
Var = 11.333333  
  
SetFormat, float, 6.2  
  
Var -= 1 ; Var 的值为 10.33，并且开头有一个空格，因为总宽度是 6。  
  
SetFormat, float, 0.2  
  
Var += 1 ; Var 的值为 11.33 并且没有空格。
```

```
SetFormat, float, 06.0  
  
Var += 0 ; Var 的值为 000011
```

```
SetFormat, integer, hex  
  
Var += 0 ; Var 的值为 0xb
```

SetFormat, integer, d

27.12 Sort

Arranges a variable's contents in alphabetical, numerical, or random order (optionally removing duplicates).

Sort, VarName [, Options]

参数

VarName	The name of the variable whose contents will be sorted.
Options	See list below.

Options

A string of zero or more of the following letters (in any order, with optional spaces in between):

C: Case sensitive sort (ignored if the **N** option is also present). If both **C** and **CL** are omitted, the uppercase letters A-Z are considered identical to their lowercase counterparts for the purpose of the sort.

CL [v1.0.43.03+]: Case insensitive sort based on the current user's locale. For example, most English and Western European locales treat the letters A-Z and ANSI letters like Ä and Ü as identical to their lowercase counterparts. This method also uses a "word sort", which treats hyphens and apostrophes in such a way that words like "coop" and "co-op" stay together. Depending on the content of the items being sorted, the performance will be 1 to 8 times worse than the default method of insensitivity.

Dx: Specifies **x** as the delimiter character, which determines where each item in *VarName* begins and ends. If this option is not present, **x** defaults to linefeed (`n), which correctly sorts *VarName* if its lines end in either LF (`n) or CR+LF (`r`n).

F MyFunction [v1.0.47+]: Uses custom sorting according to the criteria in *MyFunction* (though sorting takes much longer). Specify the letter "F" followed by optional spaces/tabs followed by the name of a [function](#)(See 10.) to be used for comparing any two items in the list. The function must accept two or three parameters. When the function deems the first parameter to be greater than the second, it should return a positive integer; when it deems the two parameters to be equal, it should return 0, "", or nothing; otherwise, it should return a negative integer. If a decimal point is present in the returned value, that part is ignored (i.e. 0.8 is the same as 0). If present, the third parameter receives the offset (in characters) of the second item from the first as seen in the original/unsorted list (see examples). Finally, the function uses the same global settings (e.g. [StringCaseSense](#)(See 27.13)) as the Sort command that called it.

Note: the **F** option causes all other options except **D**, **Z**, and **U** to be ignored (though **N**, **C**, and **CL** still affect how [duplicates](#) are detected). Also, sorting does not occur when the specified

function: 1) does not exist; 2) accepts fewer than two parameters; or 3) the first or second parameter is [ByRef](#)(See 10.).

N: Numeric sort: Each item is assumed to be a number rather than a string (for example, if this option is not present, the string 233 is considered to be less than the string 40 due to alphabetical ordering). Both decimal and hexadecimal strings (e.g. 0xF1) are considered to be numeric. Strings that do not start with a number are considered to be zero for the purpose of the sort. Numbers are treated as 64-bit floating point values so that the decimal portion of each number (if any) is taken into account.

Pn: Sorts items based on character position **n** (do not use hexadecimal for **n**). If this option is not present, **n** defaults to 1, which is the position of the first character. The sort compares each string to the others starting at its **n**th character. If **n** is greater than the length of any string, that string is considered to be blank for the purpose of the sort. When used with option **N** (numeric sort), the string's character position is used, which is not necessarily the same as the number's digit position.

R: Sorts in reverse order (alphabetically or numerically depending on the other options).

Random: Sorts in random order. This option causes all other options except **D**, **Z**, and **U** to be ignored (though **N**, **C**, and **CL** still affect how duplicates are detected). Examples:

Sort, MyVar, Random

Sort, MyVar, Random Z D|

U: Removes duplicate items from the list so that every item is unique. [ErrorLevel](#)(See 30.8) is set to the number of items removed (0 if none). If the **C** option is in effect, the case of items must match for them to be considered identical. If the **N** option is in effect, an item such as 2 would be considered a duplicate of 2.0. If either the **Pn** or **** (backslash) option is in effect, the entire item must be a duplicate, not just the substring that is used for sorting. If the **Random** option is in effect, duplicates are removed only if they appear adjacent to each other as a result of the sort. For example, when "A|B|A" is sorted randomly, the result could contain either one or two A's.

Z: To understand this option, consider a variable that contains RED`nGREEN`nBLUE`n. If the **Z** option is not present, the last linefeed (`n) is considered to be part of the last item, and thus there are only 3 items. But by specifying **Z**, the last `n (if present) will be considered to delimit a blank item at the end of the list, and thus there are 4 items (the last being blank).

\: Sorts items based on the substring that follows the last backslash in each. If an item has no backslash, the entire item is used as the substring. This option is useful for sorting bare filenames (i.e. excluding their paths), such as the example below, in which the AAA.txt line is sorted above the BBB.txt line because their directories are ignored for the purpose of the sort:

C:\BBB\AAA.txt

C:\AAA\BBB.txt

Note: Options **N** and **P** are ignored when the backslash option is present.

注意

This command is typically used to sort a variable that contains a list of lines, with each line ending in a linefeed character (`n). One way to get a list of lines into a variable is to load an entire file via [FileRead](#)(See 16.19).

If *VarName* is *Clipboard* and the clipboard contains files (such as those copied from an open Explorer window), those files will be replaced with a sorted list of their filenames. In other words, after the operation, the clipboard will no longer contain the files themselves.

[ErrorLevel](#)(See 30.8) is changed by this command only when the **U** option is in effect.

The maximum capacity of a variable can be increased via [#MaxMem](#)(See 29.15).

If a large variable was sorted and later its contents are no longer needed, you can free its memory by making it blank, e.g. MyVar =

相关命令

[FileRead](#)(See 16.19), [file-reading loop](#)(See 16.32), [parsing loop](#)(See 17.14), [StringSplit](#)(See 27.21), [RegisterCallback\(\)](#)(See 18.14), [clipboard](#)(See 30.6), [#MaxMem](#)(See 29.15)

示例

```
MyVar = 5,3,7,9,1,13,999,-4

Sort MyVar, N D, ; Sort numerically, use comma as delimiter.

MsgBox %MyVar% ; The result is -4,1,3,5,7,9,13,999

; The following example sorts the contents of a file:

FileRead(See 16.19), Contents, C:\Address List.txt

if not ErrorLevel ; Successfully loaded.

{

    Sort, Contents

    FileDelete, C:\Address List (alphabetical).txt

    FileAppend, %Contents%, C:\Address List (alphabetical).txt

    Contents = ; Free the memory.

}

; The following example makes Win+C a hotkey to copy files from an open
; Explorer window and put their sorted filenames onto the clipboard:
```

```
#c::

Clipboard = ; Must be blank for detection to work.

Send ^c

ClipWait 2

if ErrorLevel

    return

Sort Clipboard

MsgBox Ready to be pasted:`n%Clipboard%

return

; The following examples demonstrate custom sorting via a callback function.

MyVar = def`nabc`nmno`nFGH`nco-op`ncoop`ncop`ncon`n

Sort, MyVar, F StringSort

StringSort(a1, a2)

{

    return a1 > a2 ? 1 : a1 < a2 ? -1 : 0

}

MyVar = 5,3,7,9,1,13,999,-4

Sort, MyVar, F IntegerSort D,

IntegerSort(a1, a2)

{

    return a1 - a2 ; This method works only if the result never overflows a signed 64-bit
integer.

}

MyVar = 1,2,3,4

Sort, MyVar, F ReverseDirection D, ; Reverses the list so that it contains 4,3,2,1

ReverseDirection(a1, a2, offset)
```

```
{
    return offset ; Offset is positive if a2 came after a1 in the original list; negative
otherwise.
}
```

27.13 StringCaseSense

决定在字符串比较的时候是否区分大小写（默认是“不区分大小写”）。

StringCaseSense, On|Off|Locale

参数

	<p>On: 字符串比较的时候区分大小写。这个设置也让 expression equal sign operator (=)(See 9.) 以及 InStr()(See 10.) 的大小写不敏感模式按照下面的 <i>locale</i> 方式进行处理。</p>
On Off Locale	<p>Off: 不区分大小写，字母 A-Z 被认为和小写字母相同。为了保持高效和向后兼容性，这是所有脚本的默认设置（使用 <i>Locale</i> 比使用 <i>Off</i> 慢 1 - 8 倍，取决于被比较的字符串）。</p> <p>Locale [v1.0.43.03+]: 字符串不区分大小写，并且服从区域规则。例如，大多数英国以及西欧地区，不仅将 A-Z 等同于它们所对应的小写字母，同时也将 ANSI 字母比如 Ä 和 Ü 也等同于它们所对应的小写形式。</p>

注意

这个设置适用于：

- [Expression comparison operators/字符串比较符号](#)(See 9.) (除了 ==)。但是，由于 [equal-sign operator \(=\)](#)(See 9.) 总是不区分大小写的，当 *StringCaseSense* 设置为 *On* 的时候，它使用 *Locale* 模式。
- [IfInString](#)(See 27.3) , [tringGetPos](#)(See 27.14) , 以及 [InStr\(\)](#)(See 10.) 。但是 [InStr\(\)](#) 在 *CaseSensitive* 参数是 *true* 的时候，该设置不起作用。
- [StringReplace](#)(See 27.20)
- [if var in/contains MatchList](#)(See 27.4) , [if var between](#)(See 21.7) , 以及 [IfEqual and its family](#)(See 17.10) 。

内置变量 **A_StringCaseSense** 存储了当前的设置（单词 *On* , *Off* , 或者 *Locale* ）。

每一个新运行的 [Thread/线程](#)(See 30.19) (例如一个 [hotkey/热键](#)(See 4.) , [custom menu item/自定义菜单](#)(See 22.13) , 或 [timed/定时器](#)(See 17.21) 事件) 会将 *StringCaseSense* 的设置重置为默认值。要更改 *StringCaseSense* 的默认值，可以将命令放在脚本的自动执行区域（脚本的顶部）。

相关命令

[IfEqual\(See 17.10\)](#), [IfInString\(See 27.3\)](#), [if var between\(See 21.7\)](#), [StringReplace, \(See 27.20\)](#)[StringGetPos\(See 27.14\)](#)

示例

[StringCaseSense](#) [Locale](#)

27.14 StringGetPos

返回指定的子字符串在一个字符串中的位置。

`StringGetPos, OutputVar, InputVar, SearchText [, L#|R#, Offset]`
`Position := InStr(See 10.)(Haystack, Needle [, CaseSensitive?, StartingPos]) ; 具体请查看
 InStr() 函数(See 10.) 。`

参数

<code>OutputVar</code>	存储子字符串在 <i>InputVar</i> 中的位置的变量名。在 <code>StringGetPos</code> 命令中，第一个字符的位置是 0，而在 InStr()(See 10.) 函数中第一个字符的位置是 1。
<code>InputVar</code>	需要被查找的字符串变量名。不要在变量名外加百分号，除非你想使用变量中的 内容 作为变量名。
<code>SearchText</code>	查找的文本。不区分大小写，除非打开了 StringCaseSense(See 27.13) 。
<code>L# R#</code>	<p>当 <i>SearchText</i> 在 <i>InputVar</i> 中出现不止一次的时候，这个参数决定查找的方向。如果省略该参数，将从 <i>InputVar</i> 的左边开始查找，直到找到首个匹配的结果。如果该参数使用 1 或者 R，将从 <i>InputVar</i> 的右边开始查找，直到找到首个匹配的结果。</p> <p>要查找非首个匹配，在字母 L 或 R 后跟上相应的数字。例如，要从右边开始查找第四个匹配的结果，指定参数 r4。注意：如果指定的数字小于或等于 0，不会找到匹配的结果。</p>
<code>Offset</code>	偏移。从左边或右边（取决于上面的参数）开始跳过多少个字符再开始查找。如果省略，默认是 0。例如，下面的例子会从左边第十个字符开始查找： <code>StringGetPos, OutputVar, InputVar, abc, , 9</code> 。这个参数可以是一个 expression/表达式(See 9.) 。

ErrorLevel

当 *InputVar* 中未包含 *SearchText* 的时候 [ErrorLevel/错误级别\(See 30.8\)](#) 被设置为 1，包含的时候设置为 0。

注意

与 [StringMid\(See 27.19\)](#) 以及 [InStr\(\)\(See 10.\)](#) 不同的是，在 `StringGetPos` 中第一个字符的位置是 0。

返回的位置总是相对于 *InputVar* 的第一个字符的，而与是否使用 `L#|R#` 以及 `Offset` 参数无关。例如，字符串 “abc” 在 123abc789 中的位置永远是 3，不管使用了什么参数。

如果在指定的条件下，在 *InputVar* 中未能找到 *SearchText*，*OutputVar* 会被设置为 -1，
[ErrorLevel\(See 30.8\)](#) 会被设置为 1。

使用 [SplitPath\(See 16.34\)](#) 可以更容易将一个文件路径分解为目录，文件名，扩展名。

内置变量 [%A_Space%](#)(See 9.) 和 [%A_Tab%](#)(See 9.) 包含了一个单独的空格和一个单独的制表符。
 当你需要单独搜索空格或制表符，或者在 *SearchText* 的开头或结尾使用空格或制表符的时候会非常有用。

相关命令

[InStr\(\)](#)(See 10.), [RegExMatch\(\)](#)(See 18.12), [IfInString\(See 27.3\)](#), [if var in/contains MatchList\(See 27.4\)](#), [StringCaseSense\(See 27.13\)](#), [StringReplace\(See 27.20\)](#), [SplitPath\(See 16.34\)](#), [StringLeft\(See 27.15\)](#), [StringRight\(See 27.15\)](#), [StringMid\(See 27.19\)](#), [StringTrimLeft\(See 27.22\)](#), [StringTrimRight\(See 27.22\)](#), [StringLen\(See 27.17\)](#), [StringLower\(See 27.18\)](#), [StringUpper\(See 27.18\)](#), [if var is type\(See 21.8\)](#)

示例

```
Haystack = abcdefghijklmnopqrs
Needle = def
StringGetPos, pos, Haystack, %Needle%
if pos >= 0
    MsgBox, The string was found at position %pos%.
```

```
; 示例 #2:
; 将一个完整的文件路径分解为各个组成部分。
; 注意，使用 StringSplit\(See 27.21\) 或者
; parsing loop\(See 17.14\) 会更加方便，这里只是举个例子。
FileSelectFile, file, , , Pick a filename in a deeply nested folder:
if file <>
{
    StringLen, pos_prev, file
    pos_prev += 1 ; 调整位置到最后一个字符之后。
    Loop
    {
        ; 从右边开始查找第 N 个匹配
```

```

StringGetPos, pos, file, \, R%A_Index%
if ErrorLevel
    break

length := pos_prev - pos - 1

pos_prev := pos

pos += 2 ; 为了使用 StringMid 进行调整。

StringMid, path_component, file, %pos%, %length%
MsgBox Path component #%a_index% (from the right) is: `n%path_component%
}

}

```

27.15 StringLeft/StringRight

从一个字符串的左边或右边开始提取一定数量的字符。

StringLeft, OutputVar, InputVar, Count

StringRight, OutputVar, InputVar, Count

NewStr := [SubStr\(See 10.\)](#)(String, StartPos [, Length]) ; 具体请查看 [SubStr\(\) 函数\(See 10.\)](#)。

参数

OutputVar	存储从 <i>InputVar</i> 中提取出来的子字符串的变量名。
InputVar	被提取的字符串变量名。不要在变量名外加百分号，除非你想使用变量中的 <i>内容</i> 作为变量名。
Count	需要提取的字符数量，可以是 expression/表达式(See 9.) 。如果 <i>Count</i> 小于或等于 0 ， <i>OutputVar</i> 会被设置为空。如果 <i>Count</i> 超过 <i>InputVar</i> 的长度， <i>OutputVar</i> 会被设置为和 <i>InputVar</i> 相同。

注意

在这个命令以及所有其它命令中，*OutputVar* 可以和 *InputVar* 相同。

相关命令

[SubStr\(See 10.\)](#), [StringMid\(See 27.19\)](#), [StringTrimLeft\(See 27.22\)](#), [StringTrimRight\(See 27.22\)](#), [IfInString\(See 27.3\)](#), [StringGetPos\(See 27.14\)](#), [StringLen\(See 27.17\)](#), [StringLower\(See 27.18\)](#), [StringUpper\(See 27.18\)](#), [StringReplace\(See 27.20\)](#)

示例

```
String = This is a test.

StringLeft, OutputVar, String, 4 ; 在 OutputVar 中存储"This"

StringRight, OutputVar, String, 5 ; 在 OutputVar 中存储"test."
```

27.16 StrLen()

StrLen(String): 返回 *String* 的长度。如果 *String* 是 [ClipboardAll](#) 先前分配的变量，将返回它的总大小。相关命令：[StringLen](#)。

27.17 StringLen

获取字符串长度。

```
StringLen, OutputVar, InputVar
Length := StrLen(See 10.)(String)
```

参数

OutputVar	存储字符串长度的变量名。
InputVar	需要统计长度的字符串变量名。不要在变量名外加百分号，除非你想使用变量中的 <i>内容</i> 作为变量名。

注意

如果 *InputVar* 的内容是由 [ClipboardAll](#)(See 30.6) 获得的，那么 **StringLen** 将会返回它的总大小。

相关命令

[StrLen\(\)](#)(See 10.), [IfInString](#)(See 27.3), [StringGetPos](#)(See 27.14), [StringMid](#)(See 27.19), [StringTrimLeft](#)(See 27.22), [StringTrimRight](#)(See 27.22), [StringLeft](#)(See 27.15), [StringRight](#)(See 27.15), [StringLower](#)(See 27.18), [StringUpper](#)(See 27.18), [StringReplace](#)(See 27.20)

示例

```
StringLen, length, InputVar
MsgBox, The length of InputVar is %length%.
MsgBox % "The length of MyVar is " . StrLen(MyVar)
```

27.18 StringLower/StringUpper

将一个字符串中的英文字母全部转换为大写或小写。

`StringLower, OutputVar, InputVar [, T]`
`StringUpper, OutputVar, InputVar [, T]`

参数

<code>OutputVar</code>	存储转换后字符串的变量名。
<code>InputVar</code>	需要进行转换的字符串变量名。不要在变量名外加百分号，除非你想使用变量中的 <code>内容</code> 作为变量名。
<code>T</code>	如果这个参数使用字母 <code>T</code> ，字符串将被转换为标题格式。例如，“GONE with the WIND” 将被转换成“Gone With The Wind”。

注意

要判断一个字符或字符串是否全部是大写或小写字母，可以使用“`if var is [not] upper/lower`”。(See 21.8)

在这个命令以及所有其它命令中，`OutputVar` 可以和 `InputVar` 相同。

相关命令

[IfInString](#)(See 27.3), [StringGetPos](#)(See 27.14), [StringMid](#)(See 27.19), [StringTrimLeft](#)(See 27.22), [StringTrimRight](#)(See 27.22), [StringLeft](#)(See 27.15), [StringRight](#)(See 27.15), [StringLen](#)(See 27.17), [StringReplace](#)(See 27.20)

示例

```
StringUpper, String1, String1 ; 就是说 output 和 input 可以相同。
```

```
StringLower, String2, String2
```

27.19 StringMid

从字符串中指定的位置返回一个或多个字符。

`StringMid, OutputVar, InputVar, StartChar [, Count , L]`
`NewStr := SubStr(See 10.)(String, StartPos [, Length])`；具体请查看 [SubStr\(\)](#) 函数(See 10.)。

参数

<code>OutputVar</code>	存储从 <code>InputVar</code> 中提取出来的子字符串的变量名。
<code>InputVar</code>	被提取的字符串变量名。不要在变量名外加百分号，除非你想使用变量中的 <code>内容</code> 作为变量名。
<code>StartChar</code>	提取的起始位置，可以是 expression/字符串(See 9.) 。和 StringGetPos (See 27.14) 不同，1 代表第一个字符。如果 <code>StartChar</code> 小于 1，会自动设置为 1。如果 <code>StartChar</code> 大于字符串长度， <code>OutputVar</code> 会被设置为空。

Count	在 v1.0.43.10 之后的版本中，这个参数如果省略或留空，默认是一个足够存储所有字符的值。 否则，该参数表示需要提取的字符数量，可以是 expression/表达式(See 9.) 。如果 <i>Count</i> 小于或等于 0， <i>OutputVar</i> 会被设置为空。如果 <i>Count</i> 超过 <i>InputVar</i> 从 <i>StartChar</i> 开始之后的字符数量， <i>OutputVar</i> 会被设置为 <i>InputVar</i> 从 <i>StartChar</i> 开始之后的所有字符。
L	使用字母 L 可以从 <i>StartChar</i> 开始向左边提取字符，而不是默认的向右边。在下面的例子中， <i>OutputVar</i> 存储的是 Red： <i>InputVar</i> = The Red Fox <code>StringMid, OutputVar, InputVar, 7, 3, L</code> 如果使用了 L 这个参数，同时 <i>StartChar</i> 小于 1， <i>OutputVar</i> 会被设置为空。如果 <i>StartChar</i> 大于 <i>InputVar</i> 的长度，只有处于 <i>InputVar</i> 之中的字符会被提取。例如，下面的例子中 <i>OutputVar</i> 存储的是 Fox： <i>InputVar</i> = The Red Fox <code>StringMid, OutputVar, InputVar, 14, 6, L</code>

注意

在这个命令以及所有其它命令中，*OutputVar* 可以和 *InputVar* 相同。

相关命令

[SubStr\(\)\(See 10.\)](#), [StringLeft\(See 27.15\)](#), [StringRight\(See 27.15\)](#), [StringTrimLeft\(See 27.22\)](#), [StringTrimRight\(See 27.22\)](#), [IfInString\(See 27.3\)](#), [StringGetPos\(See 27.14\)](#), [StringLen\(See 27.17\)](#), [StringLower\(See 27.18\)](#), [StringUpper\(See 27.18\)](#), [StringReplace\(See 27.20\)](#)

示例

```
Source = Hello this is a test.  
StringMid, the_word_this, Source, 7, 4
```

27.20 StringReplace

替换字符串中指定的子字符串为新的字符串。

`StringReplace, OutputVar, InputVar, SearchText [, ReplaceText, ReplaceAll?]`

参数

OutputVar	存储替换后的字符串的变量名。
-----------	----------------

InputVar	需要进行替换的字符串变量名。不要在变量名外加百分号，除非你想使用变量中的 <i>内容</i> 作为变量名。
SearchText	被替换的文本。不区分大小写，除非打开了 StringCaseSense(See 27.13) 。
ReplaceText	替换的文本。替换 <i>SearchText</i> 中的内容。如果省略或留空， <i>SearchText</i> 会被替换为空。也就是说，它在 <i>OutputVar</i> 中会被省略。
ReplaceAll?	<p>如果省略，只有第一个匹配 <i>SearchText</i> 的子字符串会被替换。如果这个参数是 1、A 或者 All，则所有匹配的子字符串都将被替换。</p> <p>使用单词 UseErrorLevel 的话可以在 <i>ErrorLevel/错误级别</i> 中存储被替换的子字符串的数量（如果没有任何子字符串被替换则是 0）。UseErrorLevel 包含了“All”的作用。</p>

ErrorLevel

当最后一个参数是 **UseErrorLevel** 的时候，[ErrorLevel/错误级别\(See 30.8\)](#) 中会存储被替换的子字符串的数量（如果没有任何子字符串被替换则是 **0**）。否则，当 *InputVar* 中未包含 *SearchText* 的时候 *ErrorLevel/错误级别* 被设置为 **1**，包含的时候设置为 **0**。

注意

在这个命令以及所有其它命令中，*OutputVar* 可以和 *InputVar* 相同。

内置变量 [%A_Space%\(See 9.\)](#) 和 [%A_Tab%\(See 9.\)](#) 包含了一个单独的空格和一个单独的制表符。当你需要单独搜索空格或制表符，或者在 *SearchText* 的开头或结尾使用空格或制表符的时候会非常有用。

在 v1.0.45 版本中，为了改进性能和内存利用，**AllSlow** 参数已过时。虽然仍然可以指定它，但是不会有任何效果。

相关命令

[RegExReplace\(\)\(See 18.13\)](#), [IfInString\(See 27.3\)](#), [StringCaseSense\(See 27.13\)](#),
[StringLeft\(See 27.15\)](#), [StringRight\(See 27.15\)](#), [StringMid\(See 27.19\)](#), [StringTrimLeft\(See 27.22\)](#), [StringTrimRight\(See 27.22\)](#), [StringLen\(See 27.17\)](#), [StringLower\(See 27.18\)](#),
[StringUpper\(See 27.18\)](#), [StringGetPos\(See 27.14\)](#), [if var is type\(See 21.8\)](#)

示例

; 在剪贴板中移除换行:

```
StringReplace, clipboard, clipboard, `r`n, , All
```

; 用加号替换所有空格:

```
StringReplace, NewStr, OldStr, %A_SPACE%, +, All
```

```
; 移除字符串中的所有空行:

Loop
{
    StringReplace, MyString, MyString, `r`n`r`n, `r`n, UseErrorLevel
    if ErrorLevel = 0 ; 不需要再进行替换。
    break
}
```

27.21 StringSplit

使用指定的分隔符将一个字符串分割为一个字符串数组。

`StringSplit, OutputArray, InputVar [, Delimiters, OmitChars]`

参数

OutputArray	<p>array/数组(See 30.5) 名，存储 <i>InputVar</i> 分割成的子字符串。例如，如果这个参数使用 <code>MyArray</code>，那么变量 <code>MyArray0</code> 会存储分割出来的子字符串的数量（如果未能分割则是 0），<code>MyArray1</code> 会存储分割出来的第一个子字符串，<code>MyArray2</code> 会存储分割出来的第二个子字符串，以此类推。</p> <p>在一个 function/函数(See 10.) 内，要创建一个全局数组而不是局部数组，需要在函数内将 <code>MyArray0</code> declare/声明(See 10.) 为全局变量（反过来在设置了 assume-global(See 10.) 的函数中要创建一个局部数组则需要将 <code>MyArray0</code> 声明为局部变量）。</p>
InputVar	被分割的字符串变量名。不要在变量名外加百分号，除非你想使用变量中的 <code>内容</code> 作为变量名。注意： <code>InputVar</code> 不能与 <code>OutputArray</code> 中的变量重名。
Delimiters	<p>分隔符。如果这个参数留空或省略，<i>InputVar</i> 中的每一个字符都将作为一个子字符串被分割。</p> <p>否则，<i>Delimiters</i> 可以包含一个或多个字符（区分大小写），每个字符都用来判断从哪里分割 <i>InputVar</i>。因为分隔符不被认为是子字符串的一部分，所以分隔符不会存入 <i>OutputArray</i> 中。另外，如果在 <i>InputVar</i> 中，一对分隔符之间什么也没有，则相应的数组元素会被设置为空。</p> <p>例如：`，（转义的逗号）可以将字符串按逗号分割。相似的，%A_Tab%%A_Space% 可以将 <i>InputVar</i> 按空格或制表符分割。</p> <p>如果需要使用一个字符串作为分隔符，而不是单个字符，首先使用 StringReplace(See 27.20) 将文本中与字符串分隔符相同的内容替换为一个在文本中未出现过的字符。参考这</p>

	<p>个例子，它使用 <code>
</code> 作为分隔符：</p> <pre>StringReplace, NewHTML, HTMLString,
, ``, All ; 将所有
 替换为重音符号。</pre> <pre>StringSplit, MyArray, NewHTML, `` ; 再使用重音符号分割字符串。</pre>
OmitChars	<p>可选参数，一组字符（区分大小写）。分割出来的字符串数组中每一个元素的开头和结尾如果包含这些字符，这些字符将被移除。例如，如果 <code>OmitChars</code> 参数指定 <code>%A_Space%%A_Tab%</code>，则所有数组元素中开头和结尾（中间的除外）的所有空格和制表符将被移除。</p> <p>如果 <code>Delimiters</code> 参数留空，则 <code>OmitChars</code> 表明哪些字符将从数组中移除。</p> <p>和其它命令的最后一个参数不同，逗号在 <code>OmitChars</code> 中需要转义（<code>\,</code>）。</p>

注意

如果 `OutputArray` 指定的数组已存在，该命令会替换其中 `N` 个元素的值。`N` 是 `InputVar` 分割出来的子字符串数量。超过 `N` 之后的元素的值将不会被更改。因此，为了安全最好使用数组第一个元素(`MyArray0`)来判断到底产生了多少个数组元素。

空格和制表符会被保留，除非使用它们做分隔符。在 [AutoTrim\(See 22.3\)](#) 打开的情况下（默认打开），可以将变量赋值给自己来自动移除两端的空格和制表符。例如：`MyArray1 = %MyArray1%`

如果需要分割的字符串具有标准的 CSV（逗号分隔值）格式，可以使用 [parsing loop\(See 17.14\)](#) 内置的 CSV 处理功能。

分割字符串之前需要将字符串排序的话，使用 [Sort\(See 27.12\)](#) 命令。

如果你不需要将分割出来的子字符串存储在内存中的话，建议使用 [parsing loop\(See 17.14\)](#) —— 特别是 `InputVar` 比较大的时候，这样将会节省大量内存。例如：

```
Colors = red,green,blue
Loop, parse, Colors, `,
MsgBox Color number %A_Index% is %A_LoopField%.
```

相关命令

[Parsing loop\(See 17.14\)](#), [Arrays\(See 30.5\)](#), [Sort\(See 27.12\)](#), [SplitPath\(See 16.34\)](#), [IfInString\(See 27.3\)](#), [StringGetPos\(See 27.14\)](#), [StringMid\(See 27.19\)](#), [StringTrimLeft\(See 27.22\)](#), [StringTrimRight\(See 27.22\)](#), [StringLen\(See 27.17\)](#), [StringLower\(See 27.18\)](#), [StringUpper\(See 27.18\)](#), [StringReplace\(See 27.20\)](#)

示例

```

TestString = This is a test.

StringSplit, word_array, TestString, %A_Space%, . ; 删除句号。

MsgBox, The 4th word is %word_array4%.


Colors = red,green,blue

StringSplit, ColorArray, Colors, `,

Loop, %ColorArray0%

{

    this_color := ColorArray%a_index%


    MsgBox, Color number %a_index% is %this_color%.


}

```

27.22 StringTrimLeft/StringTrimRight

从一个字符串的左边或右边移除指定数量的字符。

StringTrimLeft, OutputVar, InputVar, Count
StringTrimRight, OutputVar, InputVar, Count
NewStr := SubStr(See 10.)(String, StartPos [, Length]) ; 具体请查看 [SubStr\(\) 函数\(See 10.\)](#)。

参数

OutputVar	存储 <i>InputVar</i> 裁减后的字符串的变量名。
InputVar	被裁减的字符串变量名。不要在变量名外加百分号，除非你想使用变量中的 <i>内容</i> 作为变量名。
Count	需要裁减的字符数量，可以是 expression/表达式(See 9.) 。如果 <i>Count</i> 小于或等于 0， <i>OutputVar</i> 会被设置为与 <i>InputVar</i> 相同。如果 <i>Count</i> 超过 <i>InputVar</i> 的长度， <i>OutputVar</i> 会被设置为空。

注意

在这个命令以及所有其它命令中，*OutputVar* 可以和 *InputVar* 相同。

相关命令

[SubStr\(\)\(See 10.\)](#), [StringMid\(See 27.19\)](#), [StringLeft\(See 27.15\)](#), [StringRight\(See 27.15\)](#),
[IfInString\(See 27.3\)](#), [StringGetPos\(See 27.14\)](#), [StringLen\(See 27.17\)](#), [StringLower\(See 27.18\)](#), [StringUpper\(See 27.18\)](#), [StringReplace\(See 27.20\)](#)

示例

```
String = This is a test.

StringTrimLeft, OutputVar, String, 5 ; 在 OutputVar 中存储“is a test.”

StringTrimRight, OutputVar, String, 6 ; 在 OutputVar 中存储“This is a”
```

27.23 SubStr()

SubStr(String, StartingPos [, Length]) [v1.0.46+]: 在 *String* 字符串中从 *StartingPos* 起始点开始向右复制不超过 *Length* 长度的字符的子字符串(如果参数 *Length* 省略, 就默认为“所有字符”)。对于 *StartingPos*, 指定 1 则从首个字符开始, 2 则从第二个字符开始, 以此类推(如果 *StartingPos* 超出了 *String* 的长度, 将返回一个空字符串)。如果 *StartingPos* 小于 1, 将被视为从字符串末尾开始的位置。例如, 0 提取最后一个字符, -1 提取最后两个字符(但是如果 *StartingPos* 超过了字符串的左侧末尾, 提取又会从左侧首个字符开始)。*Length* 是要获取的字符的最大数目(每当字符串剩余部分太短的时候, 获得的长度会比最大数目少一些)。指定一个负的 *Length* 从而在返回的字符串的末尾省略这么多个字符(如果省略了全部或更多字符, 将返回一个空字符串)。相关链接: [RegExMatch\(\)](#), [StringMid](#), [StringLeft/Right](#), [StringTrimLeft/Right](#)。

28. 窗口管理

28.1 Controls

28.1.1 Control

Makes a variety of changes to a control.

Control, Cmd [, Value, Control, WinTitle, WinText, ExcludeTitle, ExcludeText]

Parameters

Cmd, Value	See list below.
Control	<p>Can be either ClassNN (the classname and instance number of the control) or the name/text of the control, both of which can be determined via Window Spy. When using name/text, the matching behavior is determined by SetTitleMatchMode(See 28.8). If this parameter is blank, the target window's topmost control will be used.</p> <p>To operate upon a control's HWND (window handle), leave the <i>Control</i> parameter blank and specify <i>ahk_id %ControlHwnd%</i> for the <i>WinTitle</i> parameter (this also works on hidden controls even when DetectHiddenWindows(See 28.5) is Off) . The HWND of a control is typically retrieved via ControlGet Hwnd(See</p>

	28.1.4), MouseGetPos (See 23.5), or DIICall (See 18.3).
WinTitle	The title or partial title of the target window (the matching behavior is determined by SetTitleMatchMode (See 28.8)). If this and the next 3 parameters are omitted, the Last Found Window (See 30.2) will be used. If this is the letter A and the next 3 parameters are omitted, the active window will be used. To use a window class, specify ahk_class ExactClassName (shown by Window Spy). To use a process identifier (PID) (See 24.4), specify ahk_pid %VarContainingPID%. To use a window group (See 28.2.2), specify ahk_group GroupName. To use a window's unique ID number (See 28.15), specify ahk_id %VarContainingID%. The search can be narrowed by specifying multiple criteria (See 30.2). For example: <i>My File.txt ahk_class Notepad</i>
WinText	If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if DetectHiddenText (See 28.4) is ON.
ExcludeTitle	Windows whose titles include this value will not be considered.
ExcludeText	Windows whose text include this value will not be considered.

Cmd, Value

The *Cmd* and *Value* parameters are dependent upon each other and their usage is described below.

Check: Turns on (checks) a radio button or checkbox.

Uncheck: Turns off a radio button or checkbox.

Enable: Enables a control if it was previously disabled.

Disable: Disables or "grays out" a control.

Show: Shows a control if it was previously hidden.

Hide: Hides a control. If you additionally want to prevent a control's shortcut key (underlined letter) from working, disable the control via "Control Disable".

Style,N or ExStyle,N: Changes the style or extended style of a control, respectively. If the first character of **N** is a plus or minus sign, the style(s) in **N** are added or removed, respectively. If the first character is a caret (^), the style(s) in N are each toggled to the opposite state. If the first character is a digit, the control's style is overwritten completely; that is, it becomes **N**. [ErrorLevel](#)(See 30.8) is set to 1 if the target window/control is not found or the style is not allowed to be applied (which happens more often on Windows 9x).

Certain style changes require that the entire window be redrawn using [WinSet Redraw](#)(See 28.29). Also, the [styles table](#)(See 30.18) lists some of the style numbers. For example:

```
Control, Style, ^0x800000, Edit1, WinTitle ; Set the WS_BORDER style to its opposite state.
```

ShowDropDown: Drops a ComboBox so that its choices become visible.

HideDropDown: Reverses the above.

TabLeft [, Count] and **TabRight [, Count]**: Moves left or right by one or more tabs in a SysTabControl32. *Count* is assumed to be 1 if omitted or blank. To instead select a tab directly by number, replace the number 5 below with one less than the tab number you wish to select. In other words, 0 selects the first tab, 1 selects the second, and so on:

```
SendMessage(See 28.1.12), 0x1330, 5,, SysTabControl321, WinTitle ; 0x1330 is  
TCM_SETCURFOCUS.
```

Sleep 0 ; This line and the next are necessary only for certain tab controls.

```
SendMessage, 0x130C, 5,, SysTabControl321, WinTitle ; 0x130C is  
TCM_SETCURSEL.
```

Add, String: Adds *String* as a new entry at the bottom of a ListBox, ComboBox (and possibly other types).

Delete, N: Deletes the *N*th entry from a ListBox or ComboBox. *N* should be 1 for the first entry, 2 for the second, etc.

Choose, N: Sets the selection in a ListBox or ComboBox to be the *N*th entry. *N* should be 1 for the first entry, 2 for the second, etc. To select or deselect all items in a *multi-select* listbox, follow these examples:

```
PostMessage(See 28.1.12), 0x185, 1, -1, ListBox1, WinTitle ; Select all listbox items.  
0x185 is LB_SETSEL.
```

ChooseString, String: Sets the selection (choice) in a ListBox or ComboBox to be the entry whose leading part matches *String*. The search is not case sensitive. For example, if a ListBox/ComboBox contains the item "UNIX Text", specifying the word unix (lowercase) would be enough to select it.

EditPaste, String: Pastes *String* at the caret/insert position in an Edit control (this does not affect the contents of the [clipboard](#)(See 30.6)).

ErrorLevel

[ErrorLevel](#)(See 30.8) is set to 1 if there was a problem or 0 otherwise.

Remarks

To improve reliability, a delay is done automatically after every use of this command (except for *Style* and *ExStyle*). That delay can be changed via [SetControlDelay](#)(See 28.1.13).

To discover the name of the control that the mouse is currently hovering over, use [MouseGetPos](#)(See 23.5).

Window titles and text are case sensitive. Hidden windows are not detected unless [DetectHiddenWindows](#)(See 28.5) has been turned on.

Related

[SetControlDelay\(See 28.1.13\)](#), [ControlGet\(See 28.1.4\)](#), [GuiControl\(See 19.5\)](#),
[ControlGetText\(See 28.1.7\)](#), [ControlSetText\(See 28.1.10\)](#), [ControlMove\(See 28.1.8\)](#),
[ControlGetPos\(See 28.1.6\)](#), [ControlClick\(See 23.2\)](#), [ControlFocus\(See 28.1.3\)](#),
[ControlSend\(See 20.6\)](#), [WinSet\(See 28.29\)](#)

Example

```
Control, HideDropDown, , ComboBox1, Some Window Title
```

28.1.2 ControlClick

向指定控件发送一个鼠标点击或鼠标滚轮事件。

`ControlClick [, Control-or-Pos, WinTitle, WinText, WhichButton, ClickCount, Options, ExcludeTitle, ExcludeText]`

参数

	<p>如果此 参数(parameter)为空，目标窗口的最顶端控件将被点击(如果窗口没有控件则点击窗口本身)。如果有参数，则分为以下两种模式。</p> <p>模式 1 (坐标): 被指定的 X 和 Y 坐标相对于目标窗口的左上角。X 坐标必须位于 Y 坐标的前面，并且他们之间至少要有一个空格或者制表符把他们隔开。例如: <code>X55 Y33</code>。如果有一个控件位于这个指定的坐标处，则将发送鼠标事件到该坐标处(控件将被点击)。如果此处没有控件，则目标窗口自身会被发送鼠标事件(也许并不会产生效果，取决于这个窗口)。在这种情况下,Options 参数(parameter)里的 X 和 Y 选项将被忽略。</p> <p>模式 2 (控件的 ClassNN 或 文本): 可以是控件的名称/文本或 ClassNN (控件的 <code>classname</code>(类名)和序号)，它们都可以通过 <code>Window Spy</code> 来确定。当使用名称/文本时，<code>matching behavior</code> (匹配模式)通过 SetTitleMatchMode(See 28.8) 决定。</p> <p>默认情况下，模式 2 优先于模式 1。打比方，有这样的一个控件，它的文本或 ClassNN 是这样的格式: <code>"Xnnn Ynnn"</code>，这种情况下模式 2 将会生效。如果想要使用模式 1(坐标)并且避免这种情况的发生，请在 Options 中指定 Pos 一词。例如: <code>ControlClick, x255 y152, WinTitle,,, Pos</code></p> <p>如果要通过控件的 <code>HWND</code> (句柄)来操作，请在此参数(parameter)上留空并在 WinTitle 上指定 <code>ahk_id %控件句柄%</code> (这样仍可以作用于隐藏的控件即使 DetectHiddenWindows(See 28.5) 设定为 Off)。控件的 <code>HWND</code> 可以通过以下方式来获取: ControlGet Hwnd(See 28.1.4), MouseGetPos(See 23.5), 或者 DII Call(See 18.3)。</p>
WinTitle	目标窗口的标题或标题中的部分文字 (匹配模式由 SetTitleMatchMode(See 28.8) 决定)。如果这个参数和另外 3 个参数(WinText, ExcludeTitle, ExcludeText)都被忽略， 默认目标是 上一次匹配窗口(See 30.2) 。如果这个参数使用字母 A，同时省略其它 3 个窗口参数，则以当前激活的窗口作为目标。要用窗口的 class 名进行匹配，使用

	<i>ahk_class</i> 精确 class 名 (Window Spy 中可以显示 class 名)。要用窗口的 进程标识符 (PID) (See 24.4) 进行匹配，使用 <i>ahk_pid %PID%</i> 变量%。要用 窗口组 (See 28.2.2)，使用 <i>ahk_group</i> 组名 (此时 <i>WinText</i> ， <i>ExcludeTitle</i> ，以及 <i>ExcludeText</i> 三个变量要省略)。要用窗口的 唯一 ID (See 28.15) 进行匹配，使用 <i>ahk_id %ID%</i> 变量%。要减小检测范围，使用 多重条件 (See 30.2)，例如: <i>My File.txt ahk_class Notepad</i>
WinText	如果使用这个参数(parameter)，则它应该是目标窗口中某个文本元素的子字符串 (在 Window Spy 中会显示出窗口中的文本元素)。隐藏文本只有在 DetectHiddenText (See 28.4) 设置为 ON 的时候才能检测到。
WhichButton	<p>要发送的鼠标事件: LEFT(左键), RIGHT(右键), MIDDLE(中键) (或者取这些单词的首字母)。如果忽略或留空，将点击左键。</p> <p>Windows 2000/XP 及以上: X1 (X 按键 1, 第 4 个鼠标按键) 和 X2 (X 按键 2, 第 5 个鼠标按键) 是被支持的。</p> <p>Windows NT/2000/XP 及以上: WheelUp (or WU) 和 WheelDown (or WD) 是被支持的。在这种情况下，<i>ClickCount</i> 是滚轮滚动的次数。</p> <p>Windows Vista 及以上 [v1.0.48+]: WheelLeft (or WL) 和 WheelRight (or WR) 是被支持的 (它们在旧的操作系统中没有效果)。在这种情况下，<i>ClickCount</i> 是滚轮滚动的次数。</p>
ClickCount	发送鼠标点击事件的次数，可以使用 expression(表达式) (See 9.)。如果忽略或者留空，默认次数为 1。
Options	<p>一个选项字母的序列，包含零个或更多的选项字母。例如: <i>d x50 y25</i></p> <p>NA [v1.0.45+]：可能提高可靠性。参考下面的 reliability (可靠性)。</p> <p>D：按住鼠标按键并不放开(产生一个按下的事件)。如果 D 和 U 两个选项都不存在的话，则将发送一个完整的点击(按下再放开)。</p> <p>U：放开鼠标按键(产生一个放开的事件)。如果 D 已经存在则不能同时使用这个选项 (反之亦然)。</p> <p>Pos: 在 Options 的任何位置指定 Pos 一词，以无条件地使用 X/Y 坐标模式，就像在上面 <i>Control-or-Pos</i> 参数里描述的那样。</p> <p>Xn: n 指定要点击的 X 坐标，相对于控件的左上角。如果没有指定，则点击会被发送到控件的横向中点。</p> <p>Yn: n 指定要点击的 Y 坐标，相对于控件的左上角。如果没有指定，则点击会被发送到控件的纵向中点。</p> <p>请对 X 和 Y 选项使用十进制(而非十六进制)数字。</p>
ExcludeTitle	标题含有此参数值的窗口将不被考虑。

ExcludeText	文本含有此参数值的窗口将不被考虑。
-------------	-------------------

ErrorLevel

如果出现问题, [ErrorLevel\(See 30.8\)](#) 值将被设置为 1 , 否则为 0 。

reliability (可靠性)

提高可靠性 -- 特别是当用户正在使用鼠标的时候进行 `ControlClick` -- 请尝试一下的两点或其一 :

- 1) 在 `ControlClick` 之前使用 [SetControlDelay -1\(See 28.1.13\)](#) 。这样可以避免在点击时鼠标按键被按住, 以降低用户对鼠标操作时对脚本运行造成的干扰。
- 2) 在任何地方指定 `NA` 在第 6 个参数 (*Options*), 如下所示:

```
SetControlDelay -1
ControlClick, Toolbar321, WinTitle,,, NA
```

"`NA`" 可以避免激活目标窗口, 也可以避免它的输入处理和脚本的相混淆, 而这可以免除物理鼠标移动带来的干扰 (通常是在目标窗口没有激活时)。然而, 这个方法可能对所有类型的窗口和控件都无效。

注意

不是所有的程序都会响应一个 `ClickCount` 大于 1 的鼠标滚轮事件。对于这些程序, 请使用一个循环来发送一个级数大于 1 的滚轮事件, 就像这个使其滚动 5 级的例子:

```
Loop, 5
ControlClick, Control, WinTitle, WinText, WheelUp
```

窗口的标题和包含的文本是区分大小写的。隐藏的窗口是不会被检测到的, 除非 [DetectHiddenWindows\(See 28.5\)](#) 被打开(`DetectHiddenWindows On`)。

相关命令

[SetControlDelay\(See 28.1.13\)](#), [Control\(See 28.1.1\)](#), [ControlGet\(See 28.1.4\)](#),
[ControlGetText\(See 28.1.7\)](#), [ControlMove\(See 28.1.8\)](#), [ControlGetPos\(See 28.1.6\)](#),
[ControlFocus\(See 28.1.3\)](#), [ControlSetText\(See 28.1.10\)](#), [ControlSend\(See 20.6\)](#), [Click\(See 23.1\)](#)

示例

```
ControlClick, OK, Some Window Title ; 点击 OK 按钮。
ControlClick, x55 y77, WinTitle ;点击一个设置的坐标。注意 X 和 Y 之间的空隙(至少一个空格或者制表符, 没有逗号)。
; 下面的方法可能会提高可靠性并且减少其他不好的效果:
```

```
SetControlDelay -1
```

ControlClick, Toolbar321, WinTitle,,, NA x192 y10 ; 在 NA 模式下点击相对于一个指定控件的坐标。

翻译：游否 校对：hsudatalks 2009 年 2 月 27 日

28.1.3 ControlFocus

在一个窗口特定的控件上设置输入焦点。

ControlFocus [, Control, WinTitle, WinText, ExcludeTitle, ExcludeText]

参数

F8::ControlFocus, TMylistBox1, ahk_class TTOTAL_CMD

Control	可以是控件的名称/文本或 ClassNN (控件的类名和序号)，它们都可以通过 Window Spy 来确定。当使用名称/文本时，匹配模式通过 SetTitleMatchMode(See 28.8) 决定。如果此参数为空或省略，目标窗口的顶端控件将被使用。
WinTitle	目标窗口的标题或副标题(匹配模式由 SetTitleMatchMode(See 28.8) 决定)。如果此参数和后面的 3 个参数被省略，Last Found Window(See 30.2)(最近找到的窗口) 将被使用。如果此参数是字母 A 并且后面的三个参数被省略，激活的窗口将被使用。要使用一个窗口类，指定 ahk_class 确切的类名(通过 Window Spy 显示)。要使用一个 process identifier (PID)(See 24.4)(进程标识符)，指定 ahk_pid % 包含 PID 的变量%。要使用一个 window group(See 28.2.2)(窗口组)，指定 ahk_group GroupName。要使用一个窗口的 unique ID number(See 28.15)(唯一标识符编号)，指定 ahk_id % 包含 ID 的变量%。通过指定 multiple criteria(See 30.2)(多个条件) 缩小搜索范围。例如：My File.txt ahk_class Notepad
WinText	如果用到，此参数必须是目标窗口的一个单独 text element(文本对象)的 substring(子字符串) (像内置的 Window Spy 工具显示的一样)。如果 DetectHiddenText(See 28.4) 是 ON 的状态，隐藏的文本对象将被探测。
ExcludeTitle	标题含有此参数值的窗口将不被考虑。
ExcludeText	文本含有此参数值的窗口将不被考虑。

ErrorLevel

如果遇到一个问题 ErrorLevel(See 30.8) 设为 1，否则是 0。

注意

要生效，控件的窗口一般来说必须不能最小化或者隐藏。

为了改善可靠性，在每次使用此命令后都自动地有一个延迟。这个延迟可以通过 [SetControlDelay\(See 28.1.13\)](#) 来改变。

要探测鼠标当前悬停处的控件的名称，使用 [MouseGetPos\(See 23.5\)](#) 命令。

窗口标题和文本是区分大小写的。隐藏的窗口将不被探测，除非 [DetectHiddenWindows\(See 28.5\)](#) 已被打开。

相关命令

[SetControlDelay\(See 28.1.13\)](#), [ControlGetFocus\(See 28.1.5\)](#), [Control\(See 28.1.1\)](#),
[ControlGet\(See 28.1.4\)](#), [ControlMove\(See 28.1.8\)](#), [ControlGetPos\(See 28.1.6\)](#),
[ControlClick\(See 23.2\)](#), [ControlGetText\(See 28.1.7\)](#), (See 23.2)[ControlSetText\(See 28.1.10\)](#),
[ControlSend\(See 20.6\)](#)

示例

```
ControlFocus, OK, Some Window Title ; 在 OK 按键上设置焦点
```

翻译：天堂之门 menk33@163.com 2008 年 8 月 12 日

28.1.4 ControlGet

Retrieves various types of information about a control.

[ControlGet](#), [OutputVar](#), [Cmd](#) [, [Value](#), [Control](#), [WinTitle](#), [WinText](#), [ExcludeTitle](#), [ExcludeText](#)]

参数

OutputVar	The name of the variable in which to store the result of <i>Cmd</i> .
Cmd , Value	See list below.
Control	Can be either ClassNN (the classname and instance number of the control) or the name/text of the control, both of which can be determined via Window Spy. When using name/text, the matching behavior is determined by SetTitleMatchMode(See 28.8) . If this parameter is blank, the target window's topmost control will be used. To operate upon a control's HWND (window handle), leave the <i>Control</i> parameter blank and specify <i>ahk_id %ControlHwnd%</i> for the <i>WinTitle</i> parameter (this also works on hidden controls even when DetectHiddenWindows(See 28.5) is Off). The HWND of a control is typically retrieved via ControlGet Hwnd(See 28.1.4) , MouseGetPos(See 23.5) , or DII Call(See 18.3) .
WinTitle	The title or partial title of the target window (the matching behavior is

	determined by SetTitleMatchMode (See 28.8)). If this and the next 3 parameters are omitted, the Last Found Window (See 30.2) will be used. If this is the letter A and the next 3 parameters are omitted, the active window will be used. To use a window class, specify ahk_class ExactClassName (shown by Window Spy). To use a process identifier (PID) (See 24.4), specify ahk_pid %VarContainingPID%. To use a window group (See 28.2.2), specify ahk_group GroupName. To use a window's unique ID number (See 28.15), specify ahk_id %VarContainingID%. The search can be narrowed by specifying multiple criteria (See 30.2). For example: My File.txt ahk_class Notepad
WinText	If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if DetectHiddenText (See 28.4) is ON.
ExcludeTitle	Windows whose titles include this value will not be considered.
ExcludeText	Windows whose text include this value will not be considered.

Cmd, Value

The *Cmd* and *Value* parameters are dependent upon each other and their usage is described below.

List: Retrieves a list of items from a ListView, ListBox, ComboBox, or DropDownList.

[ListView](#): The most common example of a ListView is Explorer's list of files and folders (the desktop is also a ListView). The syntax for ListView retrieval is:

ControlGet, OutputVar, List, **Options**, SysListView321, WinTitle, WinText

If the *Options* parameter is blank or omitted, all the text in the control is retrieved. Each row except the last will end with a linefeed character (`n). Within each row, each field (column) except the last will end with a tab character (`t).

Specify for *Options* zero or more of the following words, each separated from the next with a space or tab:

Selected: Retrieves only the selected (highlighted) rows rather than all rows. If none, *OutputVar* is made blank.

Focused: Retrieves only the focused row. If none, *OutputVar* is made blank.

Col4: Retrieves only the fourth column (field) rather than all columns (replace 4 with a number of your choice).

Count: Retrieves a single number that is the total number of rows in the control.

Count Selected: Retrieves the number of selected (highlighted) rows.

Count Focused: Retrieves the row number (position) of the focused row (0 if none).

Count Col: Retrieves the number of columns in the control (or -1 if the count cannot be determined).

NOTE: Some applications store their ListView text privately, which prevents their text from being retrieved. In these cases, ErrorLevel will usually be set to 0 (indicating success) but all

the retrieved fields will be empty. Also note that ListView text retrieval is not restricted by [#MaxMem](#)(See 29.15).

Upon success, `ErrorLevel` is set to 0. Upon failure, it is set to 1 and `OutputVar` is made blank. Failure occurs when: 1) the target window or control does not exist; 2) the target control is not of type `SysListView32`; 3) the process owning the ListView could not be opened, perhaps due to a lack of user permissions or because it is locked; 4) the [Co/N option](#) specifies a nonexistent column.

To extract the individual rows and fields out of a ListView, use a [parsing loop](#)(See 17.14) as in this example:

```
ControlGet, List, List, Selected, SysListView321, WinTitle
Loop, Parse, List, `n ; Rows are delimited by linefeeds (`n).
{
    RowNumber := A_Index
    Loop, Parse, A_LoopField, %A_Tab% ; Fields (columns) in each row are delimited
    by tabs (A_Tab).
    MsgBox Row %%RowNumber% Col %%A_Index% is %A_LoopField%.
}
```

On a related note, the columns in a ListView can be resized via [SendMessage](#)(See 28.1.12) as shown in this example:

```
SendMessage, 4126, 0, 80, SysListView321, WinTitle ; 4126 is the message
LVM_SETCOLUMNWIDTH.
; In the above, 0 indicates the first column (specify 1 for the second, 2 for the third, etc.)
Also, 80 is the new width.
; Replace 80 with -1 to autosize the column. Replace it with -2 to do the same but also
take into account the header text width.
```

[ListBox](#), [ComboBox](#), [DropDownList](#): All the text is retrieved from the control (that is, the ListView options above such as `Count` and `Selected` are not supported).

Each row except the last will be terminated by a linefeed character (`n). To access the items individually, use a [parsing loop](#)(See 17.14) as in this example:

```
ControlGet, List, List,, ComboBox1, WinTitle
Loop, Parse, List, `n
    MsgBox Item number %%A_Index%% is %%A_LoopField%%.
```

Checked: Sets `OutputVar` to be 1 if the checkbox or radio button is checked or 0 if not.

Enabled: Sets `OutputVar` to be 1 if `Control` is enabled, or 0 if disabled.

Visible: Sets *OutputVar* to be 1 if *Control* is visible, or 0 if hidden.

Tab: Sets *OutputVar* to the tab number of a SysTabControl32 control. The first tab is 1, the second is 2, etc. To instead discover how many tabs (pages) exist in a tab control, follow this example:

```
SendMessage(See 28.1.12), 0x1304,,, SysTabControl321, WinTitle ; 0x1304 is
TCM_GETITEMCOUNT.

TabCount = %ErrorLevel%
```

FindString, String: Sets *OutputVar* to the entry number of a ListBox or ComboBox that is an exact match for *String*. The first entry in the control is 1, the second 2, and so on. If no match is found, *OutputVar* is made blank and ErrorLevel is set to 1.

Choice: Sets *OutputVar* to be the name of the currently selected entry in a ListBox or ComboBox. To instead retrieve the position of the selected item, follow this example (use only one of the first two lines):

```
SendMessage(See 28.1.12), 0x188, 0, 0, ListBox1, WinTitle ; 0x188 is
LB_GETCURSEL (for a ListBox).

SendMessage(See 28.1.12), 0x147, 0, 0, ComboBox1, WinTitle ; 0x147 is
CB_GETCURSEL (for a DropDownList or ComboBox).

ChoicePos = %ErrorLevel% ; It will be -1 if there is no item selected.

ChoicePos += 1 ; Convert from 0-based to 1-based, i.e. so that the first item is known
as 1, not 0.
```

LineCount: Sets *OutputVar* to be the number of lines in an Edit control. All Edit controls have at least 1 line, even if the control is empty.

CurrentLine: Sets *OutputVar* to be the line number in an Edit control where the caret (insert point) resides. The first line is 1. If there is text selected in the control, *OutputVar* is set to the line number where the selection begins.

CurrentCol: Sets *OutputVar* to be the column number in an Edit control where the caret (text insertion point) resides. The first column is 1. If there is text selected in the control, *OutputVar* is set to the column number where the selection begins.

Line, N: Sets *OutputVar* to be the text of line *N* in an Edit control. Line 1 is the first line. Depending on the nature of the control, *OutputVar* might end in a carriage return (`r) or a carriage return + linefeed (`r`n). If the specified line number is blank or does not exist, [ErrorLevel](#)(See 30.8) is set to 1 and *OutputVar* is made blank.

Selected: Sets *OutputVar* to be the selected text in an Edit control. If no text is selected, *OutputVar* will be made blank and ErrorLevel will be set to 0 (i.e. no error). Certain types of controls, such as RichEdit20A, might not produce the correct text in some cases (e.g. Metapad).

Style: Retrieves an 8-digit hexadecimal number representing the style of the control. See the [styles table](#) (See 30.18) for a listing of some styles.

ExStyle: Retrieves an 8-digit hexadecimal number representing the extended style of the control.

Hwnd [v1.0.43.06+]: Retrieves the window handle (HWND) of the specified control. For example: *ControlGet, OutputVar, Hwnd,, Edit1, WinTitle*. A control's HWND is often used with [PostMessage](#)(See 28.1.12), [SendMessage](#)(See 28.1.12), and [DII Call](#)(See 18.3). On a related note, a control's HWND can also be retrieved via [MouseGetPos](#)(See 23.5). Finally, a control's HWND can be used directly as an [ahk_id WinTitle](#)(See 30.2) (this also works on hidden controls even when [DetectHiddenWindows](#)(See 28.5) is Off).

ErrorLevel

Upon success, [ErrorLevel](#)(See 30.8) is set to 0. If a problem occurred -- such as a nonexistent window or control -- [ErrorLevel](#)(See 30.8) is set to 1 and *OutputVar* is made blank.

注意

Unlike commands that change a control, *ControlGet* does not have an automatic delay; that is, [SetControlDelay](#)(See 28.1.13) does not affect it.

To discover the name of the control that the mouse is currently hovering over, use [MouseGetPos](#)(See 23.5). To get a list of controls in a window, use [WinGet ControlList](#)(See 28.15).

Window titles and text are case sensitive. Hidden windows are not detected unless [DetectHiddenWindows](#)(See 28.5) has been turned on.

相关命令

[Control](#)(See 28.1.1), [GuiControlGet](#)(See 19.6), [ControlMove](#)(See 28.1.8), [ControlGetText](#)(See 28.1.7), [ControlSetText](#)(See 28.1.10), [ControlGetPos](#)(See 28.1.6), [ControlClick](#)(See 23.2), [ControlFocus](#)(See 28.1.3), [ControlSend](#)(See 20.6), [WinGet](#)(See 28.15)

示例

```
ControlGet, OutputVar, Line, 1, Edit1, Some Window Title
```

```
ControlGet, WhichTab, Tab, , SysTabControl321, Some Window Title
```

```
if ErrorLevel
```

```
    MsgBox There was a problem.
```

```

else
    MsgBox Tab #%WhichTab% is active.

```

28.1.5 ControlGetFocus

检索目标窗口的哪个控件有输入焦点，若有的话。

ControlGetFocus, OutputVar [, WinTitle, WinText, ExcludeTitle, ExcludeText]

参数

OutputVar	存储控件标识符的变量名称，控件标识符在它的父窗口里由它的类名跟序号组成，例如 <code>Button12</code> 。
WinTitle	目标窗口的标题或副标题（通过 SetTitleMatchMode (See 28.8) 定义匹配行为）。如果此参数和后面的 3 个参数被省略， 最近找到的窗口 (See 30.2) 将被使用。如果此参数是字母 A 并且后面的 3 个参数被省略，活动的窗口将被使用。要使用一个窗口类，指定 <code>ahk_class</code> 确切的类名（通过 Window Spy 显示）。要使用一个 process 标识符 (PID)(See 24.4)，指定 <code>ahk_pid</code> % 包含 PID 的变量%。要使用一个 窗口组 (See 28.2.2)，指定 <code>ahk_group</code> 组名。要使用一个窗口的 unique ID 编号(See 28.15)，指定 <code>ahk_id</code> % 包含 ID 的变量%。通过指定 多个条件 (See 30.2) 缩小搜索范围。例如： <code>My File.txt ahk_class Notepad</code>
WinText	如果出现，此参数必须是目标窗口的一个单独文本对象的子字符串（像内置的 Window Spy 工具显示的一样）。如果 DetectHiddenText (See 28.4) 是 ON 的状态，隐藏的文本对象将被探测。
ExcludeTitle	标题包括此参数值的窗口将不被考虑。
ExcludeText	文本包括此参数值的窗口将不被考虑。

ErrorLevel

如果带输入焦点的控件被成功地检索到，[ErrorLevel](#)(See 30.8) 将设为 0。否则设为 1（例如目标窗口不存在或者它没有控件带输入焦点）。

注意

彼此命令检索到的控件是带键盘焦点的，也就是说，如果用户去键入任意内容，那控件将收到键击。

目标窗口必须被激活才获得一个焦点控件。如果窗口未激活，`OutputVar` 将为空。

如果 `ControlGetFocus` 以一个高频率（例如每 500 ms 一次或更快）重复地执行，它将可能扰乱用户的双击能力。一个应对方案是通过 [DII Call](#)(See 18.3) 调用操作系统的 `GetGUIThreadInfo()`。例如：

```

; 此脚本检索激活窗口焦点控件的 ahk_id (HWND)。
; 此脚本需要 Windows 98 或者 NT 4.0 SP3+，或近期其他系统。

```

```

GuiThreadInfoSize = 48

VarSetCapacity(GuiThreadInfo, GuiThreadInfoSize)

NumPut(See 10.)(GuiThreadInfoSize, GuiThreadInfo, 0)

if not DllCall("GetGUIThreadInfo", uint, 0, str, GuiThreadInfo)

{
    MsgBox GetGUIThreadInfo() indicated a failure.

    return
}

FocusedHWND := NumGet(See 10.)(GuiThreadInfo, 12) ; 从结构体检索 hwnd 焦点区域。

MsgBox % "The focused control's ahk_id (HWND) is " . FocusedHWND

; 此 ID 能被所有控件命令使用。例如:

; ControlGetText, OutputVar,, ahk_id %FocusedHWND%

```

窗口标题和文本是区分大小写的。隐藏的窗口将不被探测，除非已打开 DetectHiddenWindows(See 28.5)。

相关命令

[ControlFocus](#)(See 28.1.3), [ControlMove](#)(See 28.1.8), [ControlClick](#)(See 23.2),
[ControlGetText](#)(See 28.1.7), [ControlSetText](#)(See 28.1.10), [ControlSend](#)(See 20.6)

示例

```

ControlGetFocus, OutputVar, 无标题 - 记事本

if ErrorLevel

    MsgBox, 目标窗口不存在或者它没有控件带输入焦点。

else

    MsgBox, 带焦点的控件 = %OutputVar%

```

翻译：天堂之门 menk33@163.com 2008 年 7 月 26 日

28.1.6 ControlGetPos

取得一个 control (控件)的位置和大小。

`ControlGetPos [, X, Y, Width, Height, Control, WinTitle, WinText, ExcludeTitle, ExcludeText]`

参数

X, Y	储存 <i>Control</i> 左上角的 X 和 Y 坐标(以像素表示)的 variables (变量)名称。这些坐标和目标窗口的左上角相关联, 因而与被用在 ControlMove (See 28.1.8) 中的坐标是一样的。 如果 X 或 Y 被省略, 相应的坐标将不被储存。
Width/Height	储存 <i>Control</i> 的高和宽(以像素表示)的变量名称。如果省略, 相应的值将不被储存。
Control	可以是控件的名称/文本或 ClassNN (控件的 classname (类名)和序号), 它们都可以通过 Window Spy 来确定。当使用名称/文本时, matching behavior (匹配模式)通过 SetTitleMatchMode (See 28.8) 决定。如果此 parameter (参数)为空, 目标窗口的顶端控件将被使用。 要对一个控件的 HWND (窗口 handle (句柄))进行操作, 将 <i>Control</i> 参数留空并为 <i>WinTitle</i> 参数指定 <i>ahk_id %ControlHwnd%</i> (这样即使当 DetectHiddenWindows (See 28.5) 是 Off 状态, 它也能对隐藏的控件起作用)。一个控件的 HWND 往往由 ControlGet Hwnd (See 28.1.4)、 MouseGetPos (See 23.5) 或 DIIICall (See 18.3) 取得。
WinTitle	目标窗口的标题或副标题(匹配模式由 SetTitleMatchMode (See 28.8) 决定)。如果此参数和后面的 3 个参数被省略, Last Found Window (See 30.2) (最近找到的窗口)将被使用。如果此参数是字母 A 并且后面的三个参数被省略, active (激活的)窗口将被使用。要使用一个窗口类, 指定 <i>ahk_class</i> 确切的类名(通过 Window Spy 显示)。要使用一个 <i>process identifier (PID)</i> (See 24.4) (进程标识符), 指定 <i>ahk_pid</i> % 包含 PID 的变量%。要使用一个 <i>window group</i> (See 28.2.2) (窗口组), 指定 <i>ahk_group</i> <i>GroupName</i> 。要使用一个窗口的 <i>unique ID number</i> (See 28.15) (唯一标识符编号), 指定 <i>ahk_id</i> % 包含 ID 的变量%。通过指定 <i>multiple criteria</i> (See 30.2) (多个条件) 缩小搜索范围。例如: My File.txt ahk_class Notepad
WinText	如果用到, 此参数必须是目标窗口的一个单独 text element (文本对象)的 substring (子字符串)(像内置的 Window Spy 工具显示的一样)。如果 DetectHiddenText (See 28.4) 是 ON 的状态, 隐藏的文本对象将被探测。
ExcludeTitle	标题含有此参数值的窗口将不被考虑。
ExcludeText	文本含有此参数值的窗口将不被考虑。

注意

如果没有找到匹配的窗口或控件, 输出变量将为空。

与改变一个控件的命令不同, ControlGetPos 没有一个自动的延迟([SetControlDelay](#)(See 28.1.13) 不会影响它)。

要找出鼠标当前停留的控件的名称, 使用 [MouseGetPos](#)(See 23.5)。要获得一个窗口内所有控件的列表, 使用 [WinGet](#)(See 28.15)。

窗口标题和文本是 **case sensitive** (区分大小写)的。隐藏的窗口将不被探测，除非 [DetectHiddenWindows\(See 28.5\)](#) 已被打开。

相关命令

[ControlMove\(See 28.1.8\)](#), [WinGetPos\(See 28.19\)](#), [Control\(See 28.1.1\)](#), [ControlGet\(See 28.1.4\)](#), [ControlGetText\(See 28.1.7\)](#), [ControlSetText\(See 28.1.10\)](#), (See 28.1.1)[ControlClick\(See 23.2\)](#), [ControlFocus\(See 28.1.3\)](#), [ControlSend\(See 20.6\)](#)

示例

```
; 这个起作用的例子将不断地更新并显示当前鼠标光标下的控件的名称和位置:
```

```
Loop
{
    Sleep, 100
    MouseGetPos, , , WhichWindow, WhichControl
    ControlGetPos, x, y, w, h, %WhichControl%, ahk_id %WhichWindow%
    ToolTip, %WhichControl%`nX%X%`tY%Y%`nW%W%`t%H%
}
```

翻译: 坛友 baggio1987 修正: 天堂之门 menk33@163.com 2008年7月30日

28.1.7 ControlGetText

从一个控件获取文本。

ControlGetText, OutputVar [, Control, WinTitle, WinText, ExcludeTitle, ExcludeText]

参数

OutputVar	储存获得的文本的变量名称
Control	<p>可以是控件的名称/文本或 ClassNN (控件的 <code>classname</code>(类名)和序号)，它们都可以通过 <code>Window Spy</code> 来确定。当使用名称/文本时，<code>matching behavior</code>(匹配模式)通过 SetTitleMatchMode(See 28.8) 决定。如果此 <code>parameter</code>(参数)为空或省略，目标窗口的顶端控件将被使用。</p> <p>要对一个控件的 <code>HWND</code> (窗口 <code>handle</code>(句柄))进行操作，将 <code>Control</code> 参数留空并为 <code>WinTitle</code> 参数指定 <code>ahk_id %ControlHwnd%</code> (这样即使当 DetectHiddenWindows(See 28.5) 是 <code>Off</code> 状态，它也能对隐藏的控件起作用)。一个控件的 <code>HWND</code> 往往由 ControlGet Hwnd(See 28.1.4)、MouseGetPos(See 23.5) 或</p>

	DII Call (See 18.3) 取得。
WinTitle	目标窗口的标题或副标题(匹配模式由 SetTitleMatchMode (See 28.8) 决定)。如果此参数和后面的 3 个参数被省略, Last Found Window (See 30.2)(最近找到的窗口)将被使用。如果此参数是字母 A 并且后面的三个参数被省略, active(激活的)窗口将被使用。要使用一个窗口类, 指定 ahk_class 确切的类名(通过 Window Spy 显示)。要使用一个 process identifier (PID) (See 24.4)(进程标识符), 指定 ahk_pid % 包含 PID 的变量%。要使用一个 window group (See 28.2.2)(窗口组), 指定 ahk_group GroupName。要使用一个窗口的 unique ID number (See 28.15)(唯一标识符编号), 指定 ahk_id % 包含 ID 的变量%。通过指定 multiple criteria (See 30.2)(多个条件) 缩小搜索范围。例如: My File.txt ahk_class Notepad
WinText	如果用到, 此参数必须是目标窗口的一个单独 text element(文本对象)的 substring(子字符串) (像内置的 Window Spy 工具显示的一样)。如果 DetectHiddenText (See 28.4) 是 ON 的状态, 隐藏的文本对象将被探测。
ExcludeTitle	标题含有此参数值的窗口将不被考虑。
ExcludeText	文本含有此参数值的窗口将不被考虑。

ErrorLevel

如果有问题, [ErrorLevel](#)(See 30.8) 值将被设置为 1 , 否则为 0 。

注意

注意: 要从一个 ListView(列表视图)、ListBox(列表框)或者 ComboBox(组合框)获取文本, 使用 [ControlGet List](#)(See 28.1.4) 来代替此命令。

如果获得的文本显得被缩短过(不完整), 试试在用 [ControlGetText](#) 命令之前先用 [VarSetCapacity](#)(OutputVar, 55)(See 18.17) [用一个与缩短的文本相比而显得非常地长的大小来替换 55]。这是很有必要的, 因为有些应用程序对于 WM_GETTEXTLENGTH 消息的反馈并不准确, 这就导致了 AutoHotkey 的输出变量过小, 以至于不能装下整个文本。

返回的文本数量受到一个变量的最大容量的限制(容量可以通过 [#MaxMem](#)(See 29.15) 指令来改变)。因此, 如果目标控件包含大量的文本(例如: 用编辑器打开一个很大的文档), 这个命令可能会使用大量的 RAM(随机存取存储器)。不过, 一个变量的存储容量能在指定其为空后就可以被释放。例如 OutputVar =

从大部分控件类型取得的文本使用 carriage return(回车)和换行(`r`n)而不是一个单独的换行(`n)来表示每一行的结束。

没有必要执行 [SetTitleMatchMode Slow](#) , 因为 [ControlGetText](#) 总是用缓慢方法来获得文本(由于它工作在一个更大范围的控件类型)。

要获得一个窗口内所有控件的列表, 使用 [WinGet ControlList](#)(See 28.15) 。

窗口标题和文本是 **case sensitive** (区分大小写)的。隐藏的窗口将不被探测，除非 [DetectHiddenWindows\(See 28.5\)](#) 已被打开。

相关命令

[ControlSetText\(See 28.1.10\)](#), [WinGetText\(See 28.20\)](#), [Control\(See 28.1.1\)](#), [ControlGet\(See 28.1.4\)](#), [ControlMove\(See 28.1.8\)](#), [ControlFocus\(See 28.1.3\)](#), [ControlClick\(See 23.2\)](#), [ControlSend\(See 20.6\)](#), [#MaxMem\(See 29.15\)](#)

示例

```
ControlGetText, OutputVar, Edit1, 无标题 -
```

翻译：坛友 baggio1987 修正：天堂之门 menk33@163.com 2008年8月4日

28.1.8 ControlMove

移动或者重新调整一个控件的大小。

ControlMove, Control, X, Y, Width, Height [, WinTitle, WinText, ExcludeTitle, ExcludeText]

参数

Control	可以是控件的名称/文本或 ClassNN (控件的 <code>classname</code> (类名)和序号)，它们都可以通过 <code>Window Spy</code> 来确定。当使用名称/文本时， <code>matching behavior</code> (匹配模式)通过 SetTitleMatchMode(See 28.8) 决定。如果此 <code>parameter</code> (参数)为空，目标窗口的顶端控件将被使用。
X, Y	<code>Control</code> 左上角的 X 和 Y 方向的新坐标(用像素单位)，它们可以是 expressions(See 9.) (表达式)。如果任何一个坐标参数为空，那么那种尺寸的 <code>Control</code> 的位置就不会改变。坐标是相对于 <code>Control</code> 的父窗口的左上角而言的； ControlGetPos(See 28.1.6) 或者 <code>Window Spy</code> 能被用来确定它们。
Width, Height	<code>Control</code> 新的高度和宽度(用像素单位)，也可以是 expressions(See 9.) (表达式)。如果任何一个参数为空或省略，那种尺寸的 <code>Control</code> 的大小将不被改变。
WinTitle	目标窗口的标题或副标题(匹配模式由 SetTitleMatchMode(See 28.8) 决定)。如果此参数和后面的 3 个参数被省略， Last Found Window(See 30.2) (最近找到的窗口)将被使用。如果此参数是字母 A 并且后面的三个参数被省略， <code>active</code> (激活的)窗口将被使用。要使用一个窗口类，指定 <code>ahk_class</code> 确切的类名(通过 <code>Window Spy</code> 显示)。要使用一个

	<p>process identifier (PID)(See 24.4)(进程标识符)，指定 <code>ahk_pid</code> %包含 PID 的变量% 。要使用一个</p> <p>window group(See 28.2.2)(窗口组)，指定 <code>ahk_group GroupName</code> 。要使用一个窗口的</p> <p>unique ID number(See 28.15)(唯一标识符编号)，指定 <code>ahk_id</code> %包含 ID 的变量% 。通过指定</p> <p>multiple criteria(See 30.2)(多个条件)缩小搜索范围。例如: <i>My File.txt ahk_class Notepad</i></p>
WinText	如果用到，此参数必须是目标窗口的一个单独文本对象的 <code>substring</code> (子字符串)(像内置的 Window Spy 工具显示的一样)。如果 DetectHiddenText (See 28.4) 是 ON 的状态，隐藏的文本对象将被探测。
ExcludeTitle	标题含有此参数值的窗口将不被考虑。
ExcludeText	文本含有此参数值的窗口将不被考虑。

ErrorLevel

若有错误发生则 **ErrorLevel**(See 30.8) 的值将被设为 1 ，否则为 0 。

注意

为了改善可靠性，每次该命令执行后自动会有一个延迟。这个延迟可以通过 **SetControlDelay**(See 28.1.13) 来 改变。

窗口标题和文本是 `case sensitive`(区分大小写)的。隐藏的窗口将不被探测，除非 **DetectHiddenWindows**(See 28.5) 已被打开。

相关命令

[ControlGetPos](#)(See 28.1.6), [WinMove](#)(See 28.27), [SetControlDelay](#)(See 28.1.13), [Control](#)(See 28.1.1), [ControlGet](#)(See 28.1.4), [ControlGetText](#)(See 28.1.7), [ControlSetText](#)(See 28.1.10), [ControlClick](#)(See 23.2), [ControlFocus](#)(See 28.1.3), [ControlSend](#)(See 20.6)

示例

```
SetTimer, ControlMoveTimer
InputBox, OutputVar, My Input Box
return

ControlMoveTimer:
IfWinNotExist, My Input Box
return
; 否则上面的窗口将为我们设为 the "last found" window(最近找到的窗口):
SetTimer, ControlMoveTimer, off
WinActivate
```

```
ControlMove, OK, 10, , 200; 将 OK 按钮移至左侧并增加其宽度。
```

```
return
```

翻译: 坛友 baggio1987 修正: 天堂之门 menk33@163.com 2008年8月4日

28.1.9 ControlSend/ControlSendRaw

Sends simulated keystrokes to a window or control.

ControlSend [, *Control*, *Keys*, *WinTitle*, *WinText*, *ExcludeTitle*, *ExcludeText*]

ControlSendRaw: Same parameters as above.

参数

Control	<p>Can be either ClassNN (the classname and instance number of the control) or the name/text of the control, both of which can be determined via Window Spy. When using name/text, the matching behavior is determined by SetTitleMatchMode(See 28.8). If this parameter is blank or omitted, the target window's topmost control will be used. If this parameter is ahk_parent, the keystrokes will be sent directly to the control's parent window (see Automating Winamp(See 30.20) for an example).</p> <p>To operate upon a control's HWND (window handle), leave the <i>Control</i> parameter blank and specify <i>ahk_id %ControlHwnd%</i> for the <i>WinTitle</i> parameter (this also works on hidden controls even when DetectHiddenWindows(See 28.5) is Off) . The HWND of a control is typically retrieved via ControlGet Hwnd(See 28.1.4), MouseGetPos(See 23.5), or DII Call(See 18.3).</p>
Keys	<p>The sequence of keys to send (see the Send(See 20.12) command for details). To send a literal comma, escape(See 29.5) it (` ,). The rate at which characters are sent is determined by SetKeyDelay(See 20.14).</p> <p>Unlike the Send(See 20.12) command, mouse clicks cannot be sent by ControlSend. Use ControlClick(See 23.2) for that.</p>
WinTitle	<p>The title or partial title of the target window (the matching behavior is determined by SetTitleMatchMode(See 28.8)). If this and the next 3 parameters are omitted, the Last Found Window(See 30.2) will be used. If this is the letter A and the next 3 parameters are omitted, the active window will be used. To use a window class, specify <i>ahk_class ExactClassName</i> (shown by Window Spy). To use a process identifier (PID)(See 24.4), specify <i>ahk_pid %VarContainingPID%</i>. To use a window group(See 28.2.2), specify <i>ahk_group GroupName</i>. To use a window's unique ID number(See 28.15), specify <i>ahk_id %VarContainingID%</i>. The search can be narrowed by specifying multiple criteria(See 30.2). For example: <i>My File.txt ahk_class Notepad</i></p>

WinText	If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if DetectHiddenText (See 28.4) is ON.
ExcludeTitle	Windows whose titles include this value will not be considered.
ExcludeText	Windows whose text include this value will not be considered.

ErrorLevel

[ErrorLevel](#)(See 30.8) is set to 1 if there was a problem or 0 otherwise.

注意

`ControlSendRaw` sends the keystrokes in the `Keys` parameter exactly as they appear rather than translating `{Enter}` to an ENTER keystroke, `^c` to Control-C, etc.

If the `Control` parameter is omitted, this command will attempt to send directly to the target window by sending to its topmost control (which is often the correct one) or the window itself if there are no controls. This is useful if a window does not appear to have any controls at all, or just for the convenience of not having to worry about which control to send to.

By default, modifier keystrokes (Control, Alt, Shift, and Win) are sent as they normally would be by the `Send` command. This allows command prompt and other console windows to properly detect uppercase letters, control characters, etc. It may also improve reliability in other ways.

However, in some cases these modifier events may interfere with the active window, especially if the user is actively typing during a `ControlSend` or if the Alt key is being sent (since Alt activates the active window's menu bar). This can be avoided by explicitly sending modifier up and down events as in this example:

```
ControlSend, Edit1, {Alt down}f{Alt up}, Untitled - Notepad
```

The method above also allows the sending of modifier keystrokes (Control/Alt/Shift/Win) while the workstation is locked (protected by logon prompt).

[BlockInput](#)(See 20.5) should be avoided when using `ControlSend` against a console window such as command prompt. This is because it might prevent capitalization and modifier keys such as Control from working properly.

The value of [SetKeyDelay](#)(See 20.14) determines the speed at which keys are sent. If the target window does not receive the keystrokes reliably, try increasing the press duration via the second parameter of [SetKeyDelay](#)(See 20.14) as in these examples:

```
SetKeyDelay, 10, 10
SetKeyDelay, 0, 10
SetKeyDelay, -1, 0
```

If the target control is an Edit control (or something similar), the following are usually more reliable and faster than `ControlSend`:

[Control](#)(See 28.1.1), `EditPaste`, This text will be inserted at the caret position., `ControlName`,

WinTitle

[ControlSetText](#)(See 28.1.10), ControlName, This text will entirely replace any current text., WinTitle

ControlSend is generally not capable of manipulating a window's menu bar. To work around this, use [WinMenuSelectItem](#)(See 28.1.14). If that is not possible due to the nature of the menu bar, you could try to discover the message that corresponds to the desired menu item by following the [SendMessage Tutorial](#)(See 30.16).

Window titles and text are case sensitive. Hidden windows are not detected unless [DetectHiddenWindows](#)(See 28.5) has been turned on.

相关命令

[SetKeyDelay](#)(See 20.14), [Escape sequences \(e.g. `%\)](#) (See 29.5), [Control](#)(See 28.1.1), [ControlGet](#)(See 28.1.4), [ControlGetText](#)(See 28.1.7), [ControlMove](#)(See 28.1.8), [ControlGetPos](#)(See 28.1.6), [ControlClick](#)(See 23.2), [ControlSetText](#)(See 28.1.10), [ControlFocus](#)(See 28.1.3), [Send](#), ([See 20.12](#))[Automating Winamp](#)(See 30.20)

示例

```
ControlSend, Edit1, This is a line of text in the notepad window., Untitled
```

```
SetTitleMatchMode, 2
```

```
ControlSend, , abc, cmd.exe ; Send directly to a command prompt window.
```

28. 1. 10 ControlSetText

Changes the text of a control.

`ControlSetText [, Control, NewText, WinTitle, WinText, ExcludeTitle, ExcludeText]`

参数

Control	<p>Can be either ClassNN (the classname and instance number of the control) or the name/text of the control, both of which can be determined via Window Spy. When using name/text, the matching behavior is determined by SetTitleMatchMode(See 28.8). If this parameter is blank, the target window's topmost control will be used.</p> <p>To operate upon a control's HWND (window handle), leave the <i>Control</i> parameter blank and specify <i>ahk_id %ControlHwnd%</i> for the <i>WinTitle</i> parameter (this also works on hidden controls even when DetectHiddenWindows(See 28.5) is Off) . The HWND of a control is typically retrieved via ControlGet Hwnd(See 28.1.4), MouseGetPos(See 23.5), or DII Call(See 18.3).</p>
---------	--

NewText	The new text to set into the control. If blank or omitted, the control is made blank.
WinTitle	The title or partial title of the target window (the matching behavior is determined by SetTitleMatchMode (See 28.8)). If this and the next 3 parameters are omitted, the Last Found Window (See 30.2) will be used. If this is the letter A and the next 3 parameters are omitted, the active window will be used. To use a window class, specify ahk_class ExactClassName (shown by Window Spy). To use a process identifier (PID) (See 24.4), specify ahk_pid %VarContainingPID%. To use a window group (See 28.2.2), specify ahk_group GroupName. To use a window's unique ID number (See 28.15), specify ahk_id %VarContainingID%. The search can be narrowed by specifying multiple criteria (See 30.2). For example: <i>My File.txt ahk_class Notepad</i>
WinText	If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if DetectHiddenText (See 28.4) is ON.
ExcludeTitle	Windows whose titles include this value will not be considered.
ExcludeText	Windows whose text include this value will not be considered.

ErrorLevel

[ErrorLevel](#)(See 30.8) is set to 1 if there was a problem or 0 otherwise.

注意

Most control types use carriage return and linefeed (`r`n) rather than a solitary linefeed (`n) to mark the end of each line. To translate a block of text containing `n characters, follow this example: [StringReplace](#)(See 27.20), MyVar, MyVar, `n, `r`n, All

To improve reliability, a delay is done automatically after every use of this command. That delay can be changed via [SetControlDelay](#)(See 28.1.13).

Window titles and text are case sensitive. Hidden windows are not detected unless [DetectHiddenWindows](#)(See 28.5) has been turned on.

相关命令

[SetControlDelay](#)(See 28.1.13), (See 28.1.5)[ControlGetText](#)(See 28.1.7), [ControlGet](#)(See 28.1.4), [Control](#)(See 28.1.1), [ControlMove](#)(See 28.1.8), [ControlGetPos](#)(See 28.1.6), [ControlClick](#)(See 23.2), [ControlFocus](#)(See 28.1.3), [ControlSend](#)(See 20.6)

示例

```
ControlSetText, Edit1, New Text Here, Untitled -
```

28.1.11 Menu

创建、删除、修改以及显示菜单和菜单项。更改托盘图标和它的提示。控制是否允许打开[已编译脚本](#)(See 8.)的主窗口。

Menu, MenuName, Cmd [, P3, P4, P5]

参数

MenuName	它可以是 TRAY 或者任何自定义菜单的名称。自定义菜单的名称在第一次和 <i>Add</i> 命令一起使用时就会自动地创建这个自定义菜单。例如: <code>Menu, MyMenu, Add, Item1</code> 一旦被创建, 就可以用 <i>Show</i> 命令来显示自定义菜单。通过 <i>Add</i> 命令, 它也可以附着为一个或多个其它菜单的子菜单。
Cmd, P3, P4, P5	这 4 个参数相互依赖。可用的组合如下所示。

Add [, MenuItemName, Label-or-Submenu, Pn]: 此命令有多个用途, 如添加菜单项、为其更新一个新的子菜单或标签、或者将普通菜单项转换成子菜单(或者相反)。如果 *MenuItemName* 还不存在, 它将被添加到菜单。否则, *MenuItemName* 将会被最新指定的 *Label-or-Submenu* 所更新。

要添加一个菜单分割线, 只需省略所有三个参数。

当用户选择菜单项时, 将以一个新的[线程](#)(See 30.19)运行标签子程序(类似 [Gosub](#)(See 17.8) 和[热键子程序](#)(See 4.))。如果省略了 *Label-or-Submenu*, 那么 *MenuItemName* 将同时作为标签和菜单项的名称使用。

要让 *MenuItemName* 成为子菜单(就是一个菜单项在它被选择时会打开一个新的菜单), 那么给 *Label-or-Submenu* 指定一个冒号后跟一个已存在的自定义菜单 *MenuName* 即可。例如:

```
Menu, MySubmenu, add, Item1
Menu, tray, add, This Menu Item Is A Submenu, :MySubmenu
```

最后一个参数可以包含字母 P 后跟一个菜单的[线程优先级](#)(See 30.19), 比如 P1。如果在添加菜单项时省略此参数, 优先级将是标准的默认值 0。如果在更新菜单项时省略, 菜单项的优先级将不做改变。要仅仅改变现有的菜单项的优先级, 可以省略参数 *Label-or-Submenu*。优先级使用十进制数字(不是十六进制)。

Delete [, MenuItemName]: 从菜单删除 *MenuItemName*。标准菜单项比如 *Exit* (见下文)不能被单独地删除。如果 *default* 菜单项被删除, 其效果如同使用了 *NoDefault* 选项。如果省略 *MenuItemName*, 整个 *MenuName* 菜单将被删除, 同时删除在其它菜单中使用 *MenuName* 作为子菜单的任何菜单项。

DeleteAll: 从菜单中删除所有自定义的菜单项, 使菜单留空, 除非它包含了 *standard* 菜单项(见下文)。不同于 *Delete* 命令删除整个菜单(如上文), 空菜单仍然存在, 因此将它用作子菜单的任何其它菜单仍将保留这些子菜单。

Rename, MenuItemName [, NewName]: 将 *MenuItemName* 变更为 *NewName* (如果 *NewName* 为空, *MenuItemName* 将转换为分割线)。*NewName* 不能和任何已存在的自定义菜单项同名。菜单项当前的目标标签和子菜单不变。

Check, MenuItemName: 在菜单的 *MenuItemName* 前添加一个可见的对钩标识(如果还没有的话)。

Uncheck, MenuItemName: 从 *MenuItemName* 移除对钩标识(如果有的话)。

ToggleCheck, MenuItemName: 如果没有对钩标识则添加一个; 否则, 将它移除。

Enable, MenuItemName: 允许用户再次选择 *MenuItemName*, 如果它先前被禁用(变灰)的话。

Disable, MenuItemName: 使 *MenuItemName* 变成灰色以表示用户不能选择它。

ToggleEnable, MenuItemName: 如果 *MenuItemName* 之前可用, 则禁用它; 否则, 将启用它。

Default [, MenuItemName]: 将 *MenuItemName* 设为菜单的默认项并加粗它的字体(目前在 TRAY 以外的菜单中设置默认项, 仅是改变外观而已)。当用户双击托盘图标时, 将开启它的默认菜单项。如果没有默认项, 双击将没有任何效果。如果省略 *MenuItemName*, 将和下文的 *NoDefault* 作用相同。

NoDefault: 对于托盘菜单: 将菜单改回到它标准菜单的默认项, 对未编译脚本来说是 OPEN, 而已编译脚本(See 8.)则没有(除非当 MainWindow 选项生效时)。如果之前使用了下面的 NoStandard 命令而导致 OPEN 菜单项不存在的话, 那么将没有默认项, 因此双击托盘图标也将没有任何效果。对于 TRAY 以外的菜单: 任何现有的默认项将变回非加粗字体。

Standard: 在菜单底部插入标准菜单项(如果它们还不存在的话)。此命令可用于托盘菜单或者任何其它菜单。

NoStandard: 从托盘菜单移除所有标准的(非自定义的)菜单项(如果它们存在的话)。

Icon [, FileName, IconNumber, 1]: 将脚本的图标更改为 *FileName* 内图标中的一个。支持下列类型的文件: ICO, CUR, ANI, EXE, DLL, CPL, SCR 以及其它包含图标资源的类型。要使用文件内首个以外的图标组, 将它的编号指定给 *IconNumber* (如果省略, 那么它默认为 1)。例如, 2 将会从第二个图标组加载默认图标。指定星号(*)给 *FileName* 可以将脚本恢复到它的默认图标。

最后一个参数: 要冻结图标, 可将末尾参数设为 1, 要解冻则设为 0 (或留空从而保持冻结/解冻状态不做改变)。当图标已被冻结时, Pause(See 17.18) 和 Suspend(See 17.23) 将不能改变图标。注意: 要冻结当前的图标, 可以像后面例子一样使用 1 或 0 : *Menu, Tray, Icon,,, 1*

修改托盘图标也会改变 InputBox(See 19.10), Progress(See 19.12) 以及随后创建的 GUI(See 19.3) 窗口所显示的图标。已编译脚本(See 8.)即使在编译时已经指定过自定义的图标也会受其影响。注意: 如果先前使用例如 #NoTrayIcon(See 22.1) 隐藏了托盘图标, 那么改变图标是不会反隐藏它的; 要这样做的话, 可以使用 *Menu, Tray, Icon* (不带参数)。

从除了 .ICO 外的其他文件类型加载托盘图标时, 可能会产生轻微的变形。特别是 16x16 的图标。要避免这种情况, 将想要的托盘图标保存在 .ICO 文件里。

操作系统的 DLL 和 CPL 文件包含的一些图标可能会很有用。例如: *Menu, Tray, Icon, Shell32.dll, 174*; 省略 Dll 的路径以便它也能用在 Windows 9x 上。

内置变量 **A_IconNumber** 和 **A_IconFile** 包含了当前图标(如果是默认图标, 两者都为空)的序号和名称(带完整的路径)。

Icon (不带参数): 如果不存在托盘图标则创建它。如果脚本里也存在 [#NoTrayIcon](#)(See 22.1) 指令的话, 此命令将废除它。

NoIcon: 如果存在托盘图标则移除它。如果在脚本的最顶部使用此命令, 托盘图标可能在装载脚本时暂时地可见。要避免这种情况, 可以使用 [#NoTrayIcon](#)(See 22.1) 代替。如果当前托盘图标被隐藏, 内置变量 **A_IconHidden** 为 1, 否则为 0。

Tip [, Text]: 改变托盘图标的提示, 其在鼠标悬停于图标上面时会显示。要创建一个多行的提示, 在每一行中间使用换行符(`n), 例如: Line1`nLine2。将只显示 *Text* 的前 127 个字符。如果省略 *Text*, 提示将被恢复为默认文本。内置变量 **A_IconTip** 包含了当前提示的文本(如果是默认文本则为空)。

Show [, X, Y]: 显示 *MenuName*, 允许用户通过方向键、菜单快捷键(加下划线的字母)或者鼠标来选择菜单项。可以显示包括托盘菜单在内的任何菜单, 除了 [GUI](#)(See 19.3) 菜单栏外。如果省略 X 和 Y, 菜单会在当前的鼠标光标位置显示。如果只省略了它们中的一个, 那么会用鼠标光标的位置来代替它。X 和 Y 是相对于激活窗口的。事先指定 "[CoordMode](#)(See 22.6), Menu" 可以使它们相对于整个屏幕。

Color, ColorValue [, Single]: 将菜单的背景色改成 *ColorValue*, 它是 16 种 HTML 基础颜色之一或者是 6 位的 RGB 颜色值(请看[颜色表](#)(See 19.12))。将 *ColorValue* 留空(或指定为单词 Default)可以将菜单恢复为它的默认颜色。如果没有在下个参数中指定单词 Single, 那么附在这个菜单上的任何子菜单也将被改变颜色。此命令在 Windows 95/NT 上无效。

Click, ClickCount: 将 *ClickCount* 指定为 1 来允许单击激活托盘菜单的默认菜单项。将它指定为 2 可以返回到默认行为(双击)。例如: Menu, Tray, Click, 1

MainWindow: 此命令只影响[已编译脚本](#)(See 8.)。它允许通过托盘图标来打开脚本的主窗口, 反之则打不开。它也能启用主窗口的 View 菜单项, 比如 "Lines most recently executed" 就可以查看脚本的源代码和其他信息。*MenuName* 必须是 TRAY。

NoMainWindow (默认): 此命令只影响[已编译脚本](#)(See 8.)。它使脚本恢复为它的默认行为, 也就是说它会防止主窗口被打开。即使此选项生效, 但在脚本运行时遇上 [ListLines](#)(See 22.11), [ListVars](#)(See 22.12), [ListHotkeys](#)(See 20.1.11) 和 [KeyHistory](#)(See 20.9) 命令时, 它们仍可以显示主窗口。*MenuName* 必须是 TRAY。

UseErrorLevel [, off]: 如果在脚本中从未使用过此命令, 那么它默认为 OFF。每当 Menu 命令遇到错误时, OFF 设置会显示一个对话框并终止[当前线程](#)(See 30.19)。指定 *Menu, Tray, UseErrorLevel* 可以阻止对话框显示和线程终止; 取代它们的是, 如果遇到问题 [ErrorLevel](#)(See 30.8) 将被设为 1, 否则为 0。要将此选项调回 off, 可以把下个参数设为 OFF。此设置是全局性的, 这意味着它会影响所有的菜单, 而不仅仅是菜单 *MenuName*。

要给菜单项的某个字母加下划线, 那么在那个字母前加一个 & 符号。当菜单显示时, 可以按键盘上相应的按键来选择此菜单项。要显示一个原义的 & 符号, 只需指定两个连续的 & 符号, 例如: Save && Exit

菜单和菜单项的名称长度最多可达 260 个字符。

当引用已有的菜单或菜单项时，名称不区分大小写，但是必须包含 & 符号。例如：&Open

可以使用 *Menu, tray, add* (即省略其他所有的参数)来给菜单添加分割线。不过，当前还不能单独删除分割线。要绕弯解决这种情况，可使用 *Menu, tray, DeleteAll*，然后重新添加你的自定义菜单项。

新的菜单项总是添加到菜单的底部。对于托盘菜单：要将你的菜单项放在标准菜单项的上面(在添加完你的菜单项后)，可以运行 *Menu, tray, NoStandard* 后跟 *Menu, tray, Standard*。

不能使用任何菜单子命令来单独地操作标准菜单项，比如 "Pause Script" 和 "Suspend Hotkeys"。

如果菜单已完全变空，比如使用了 *Menu, MyMenu, DeleteAll* 命令，则它不会被显示。如果托盘菜单变空，那么右击和双击托盘图标将不起任何作用(这种情况下，通常使用 [#NoTrayIcon](#)(See 22.1) 会更好)。

如果菜单项的子程序运行时，用户再次选择了同个菜单项，那么将创建一个新的线程(See 30.19)来运行那个相同的子程序，并打断之前的线程。如果想缓存并推迟执行的话，将 [Critical](#)(See 22.7) 作为子程序的首行(不过，这么做同样会缓冲/推迟其它的线程，比如按了一个热键)。

每当通过菜单项启动一个子程序时，都将会刷新设置比如 [SendMode](#)(See 20.13) 的默认值。可以在[自动执行部分](#)(See 8.)修改这些默认值。

内置变量 [**A_ThisMenuItem**](#)(See 9.) 和 [**A_ThisMenuItemPos**](#)(See 9.) 分别包含了最近用户选择的自定义菜单项的名称和位置(如果没有则为空)。类似地，[**A_ThisMenu**](#) 是选择的 [**A_ThisMenuItem**](#) 的菜单名称。当创建的菜单内容不会一直相同时，这些变量就会很有用。在这种情况下，通常最好给这些菜单项指定相同的标签，然后把那些标签关联到上面的变量，从而确定要执行的动作。

要让一个没有热键、无 [GUI](#)(See 19.3) 的脚本持续运行，比如仅包含了自定义菜单或菜单项的脚本，可以使用 [#Persistent](#)(See 29.21)。

[GUI](#)(See 19.3), [Threads](#)(See 30.19), [Thread](#)(See 22.22), [Critical](#)(See 22.7), [#NoTrayIcon](#)(See 22.1), [Gosub](#)(See 17.8), [Return](#)(See 17.19), [SetTimer](#)(See 17.21), [#Persistent](#)(See 29.21)

;示例 #1：该脚本在托盘图标菜单底部添加了一个新的菜单项。

```
#Persistent ;保持脚本运行，直到用户退出它。
Menu, tray, add ;创建一条分割线。
Menu, tray, add, Item1, MenuHandler ;创建一个新的菜单项。
return

MenuHandler:
```

```
MsgBox 你从 %A_Menu% 菜单中选择了 %A_MenuItem% 菜单项。
```

```
return
```

;示例 #2: 该脚本创建了一个弹出菜单, 当用户按热键 Win-Z 将显示它。

;通过添加菜单项来创建弹出菜单。

```
Menu, MyMenu, Add, Item1, MenuHandler
```

```
Menu, MyMenu, Add, Item2, MenuHandler
```

```
Menu, MyMenu, Add ;添加分割线。
```

;为上述菜单创建子菜单。

```
Menu, Submenu1, Add, Item1, MenuHandler
```

```
Menu, Submenu1, Add, Item2, MenuHandler
```

;为第一个菜单创建子菜单(右箭头指示符)。当用户选择它时将显示第二个菜单。

```
Menu, MyMenu, Add, My Submenu, :Submenu1
```

Menu, MyMenu, Add ;在子菜单下添加分割线。

```
Menu, MyMenu, Add, Item3, MenuHandler ;在子菜单下添加另一个菜单项。
```

```
return ;脚本自动执行段的结尾。
```

MenuHandler:

```
MsgBox 你从 %A_Menu% 菜单中选择了 %A_MenuItem% 菜单项。
```

```
return
```

#z::Menu, MyMenu, Show ;即按下热键 Win-Z 显示菜单。

;示例 #3: 该脚本演示了多个菜单命令。

```
#Persistent  
  
#SingleInstance  
  
menu, tray, add ;分割线  
  
menu, tray, add, TestToggle&Check  
  
menu, tray, add, TestToggleEnable  
  
menu, tray, add, TestDefault  
  
menu, tray, add, TestStandard  
  
menu, tray, add, TestDelete  
  
menu, tray, add, TestDeleteAll  
  
menu, tray, add, TestRename  
  
menu, tray, add, Test  
  
return
```

```
;::::::::::::::::::;
```

TestToggle&Check:

```
menu, tray, ToggleCheck, TestToggle&Check  
  
menu, tray, Enable, TestToggleEnable ;它也能启用下面那个测试，因为它不能撤销它自己的禁用  
状态。  
  
menu, tray, add, TestDelete ;与上一行相似。  
  
return
```

TestToggleEnable:

```
menu, tray, ToggleEnable, TestToggleEnable  
  
return
```

TestDefault:

```
if default = TestDefault  
{
```

```
menu, tray, NoDefault

default =

}

else

{

    menu, tray, Default, TestDefault

    default = TestDefault

}

return

TestStandard:

if standard <> n

{

    menu, tray, NoStandard

    standard = n

}

else

{

    menu, tray, Standard

    standard = y

}

return

TestDelete:

menu, tray, delete, TestDelete

return

TestDeleteAll:

menu, tray, DeleteAll
```

```

return

TestRename:
if NewName <> renamed
{
    OldName = TestRename
    NewName = renamed
}
else
{
    OldName = renamed
    NewName = TestRename
}
menu, tray, rename, %OldName%, %NewName%
return

Test:
MsgBox, 你在 "%A_ThisMenu%" 菜单选择了 "%A_ThisMenuItem%"。
return

```

翻译: lwjeee 修正: 天堂之门 2008 年 11 月 20 日

28.1.12 PostMessage/SendMessage

Sends a message to a window or control (SendMessage additionally waits for acknowledgement).

PostMessage, **Msg** [, wParam, lParam, Control, WinTitle, WinText, ExcludeTitle, ExcludeText]
SendMessage, **Msg** [, wParam, lParam, Control, WinTitle, WinText, ExcludeTitle, ExcludeText]

参数

Msg	The message number to send, which can be an expression (See 9.). See the message list (See 30.17) to determine the number.
wParam	The first component of the message, which can be an expression (See 9.). If

	blank or omitted, 0 will be sent.
IParam	The second component of the message, which can be an expression (See 9.). If blank or omitted, 0 will be sent.
Control	If this parameter is blank or omitted, the message will be sent directly to the target window rather than one of its controls. Otherwise, this parameter can be either ClassNN (the classname and instance number of the control) or the name/text of the control, both of which can be determined via Window Spy. When using name/text, the matching behavior is determined by SetTitleMatchMode (See 28.8). To operate upon a control's HWND (window handle), leave the <i>Control</i> parameter blank and specify <code>ahk_id %ControlHwnd%</code> for the <i>WinTitle</i> parameter (this also works on hidden controls even when DetectHiddenWindows (See 28.5) is Off). The HWND of a control is typically retrieved via ControlGet Hwnd (See 28.1.4), MouseGetPos (See 23.5), or DII Call (See 18.3).
WinTitle	The title or partial title of the target window (the matching behavior is determined by SetTitleMatchMode (See 28.8)). If this and the next 3 parameters are omitted, the Last Found Window (See 30.2) will be used. If this is the letter A and the next 3 parameters are omitted, the active window will be used. To use a window class, specify <code>ahk_class ExactClassName</code> (shown by Window Spy). To use a process identifier (PID) (See 24.4), specify <code>ahk_pid %VarContainingPID%</code> . To use a window group (See 28.2.2), specify <code>ahk_group GroupName</code> . To use a window's unique ID number (See 28.15), specify <code>ahk_id %VarContainingID%</code> (also accepts a control's HWND (See 28.1.4)). The search can be narrowed by specifying multiple criteria (See 30.2). For example: <code>My File.txt ahk_class Notepad</code>
WinText	If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if DetectHiddenText (See 28.4) is ON.
ExcludeTitle	Windows whose titles include this value will not be considered.
ExcludeText	Windows whose text include this value will not be considered.

ErrorLevel

PostMessage: [ErrorLevel](#)(See 30.8) is set to 1 if there was a problem such as the target window or control not existing. Otherwise, it is set to 0.

SendMessage: [ErrorLevel](#)(See 30.8) is set to the word FAIL if there was a problem. Otherwise, it is set to the numeric result of the message, which might sometimes be a "reply" depending on the nature of the message and its target window. This result is an integer between 0 and 4294967295. If the result is intended to be a signed integer, a negative number can be

revealed by following this example: `MsgReply := ErrorLevel > 0x7FFFFFFF ? -(~ErrorLevel) - 1 : ErrorLevel`

注意

These commands should be used with caution because sending a message to the wrong window (or sending an invalid message) might cause unexpected behavior or even crash the target application. This is because most applications are not designed to expect certain types of messages from external sources.

`PostMessage` places the message in the message queue associated with the target window. It does not wait for acknowledgement or reply. By contrast, `SendMessage` waits up to 5 seconds for the target window to process the message. If the message is not processed within this time, the command finishes and sets `ErrorLevel` to the word FAIL.

The parameters `Msg`, `wParam`, and `lParam` should all be integers between -2147483648 and 4294967295 (0xFFFFFFFF). As with all integer values in AutoHotkey, a prefix of 0x indicates a hex value. For example, 0xFF is equivalent to 255.

A string may be sent via `wParam` or `lParam` by specifying the address of a variable. The following example uses the [address operator \(&\)](#)(See 9.) to do this:

```
SendMessage, 0xC, 0, &MyVar, ClassNN, WinTitle ; 0xC is WM_SETTEXT
```

In v1.0.43.06+, a string put into `MyVar` by the receiver of the message is properly recognized without the need for extra steps. However, this works only if the parameter's first character is an ampersand (&); for example, `5+&MyVar` would not work but `&MyVar` or `&MyVar+5` would work.

A quoted/literal string may also be sent as in the following working example (the & operator should not be used in this case):

```
Run Notepad  
WinWait Untitled - Notepad  
SendMessage, 0xC, 0, "New Notepad Title" ; 0xC is WM_SETTEXT
```

To send a message to all windows in the system, including those that are hidden or disabled, specify `ahk_id 0xFFFF` for `WinTitle` (0xFFFF is `HWND_BROADCAST`). This technique should be used only for messages intended to be broadcast, such as the following example:

```
SendMessage, 0x1A,,, ahk_id 0xFFFF ; 0x1A is WM_SETTINGCHANGE
```

To have a script receive a message, use [OnMessage\(\)](#)(See 18.11).

See the [Message Tutorial](#)(See 30.16) for an introduction to using these commands.

Window titles and text are case sensitive. Hidden windows are not detected unless [DetectHiddenWindows](#)(See 28.5) has been turned on.

相关命令

[Message List](#)(See 30.17), [Message Tutorial](#)(See 30.16), [OnMessage\(\)](#)(See 18.11), [Automating Winamp](#)(See 30.20), [DII Call](#)(See 18.3), [ControlSend](#)(See 20.6), [WinMenuSelectItem](#)(See 28.1.14)

示例

```
#o:: ; Win+O hotkey that turns off the monitor.

Sleep 1000 ; Give user a chance to release keys (in case their release would wake up the monitor again).

; Turn Monitor Off:

SendMessage, 0x112, 0xF170, 2,, Program Manager ; 0x112 is WM_SYSCOMMAND,
0xF170 is SC_MONITORPOWER.

; Note for the above: Use -1 in place of 2 to turn the monitor on.

; Use 1 in place of 2 to activate the monitor's low-power mode.

return

; Start the user's chosen screen saver:

SendMessage, 0x112, 0xF140, 0,, Program Manager ; 0x112 is WM_SYSCOMMAND, and
0xF140 is SC_SCREENSAVE.

; Scroll up by one line (for a control that has a vertical scroll bar):

ControlGetFocus, control, A

SendMessage, 0x115, 0, 0, %control%, A

; Scroll down by one line:

ControlGetFocus, control, A

SendMessage, 0x115, 1, 0, %control%, A

; Switch the active window's keyboard layout/language to English:

PostMessage, 0x50, 0, 0x4090409,, A ; 0x50 is WM_INPUTLANGCHANGEREQUEST.
```

```

; This example asks Winamp which track number is currently active:

SetTitleMatchMode, 2

SendMessage, 1024, 0, 120, - Winamp

if ErrorLevel <> FAIL

{

    ErrorLevel++ ; Winamp's count starts at "0", so adjust by 1.

    MsgBox, Track #%ErrorLevel% is active or playing.

}

; See Automating Winamp(See 30.20) for more information.

;

; To find the process ID of an AHK script (an alternative to "WinGet PID(See 28.15)"):

SetTitleMatchMode, 2

DetectHiddenWindows, on

SendMessage, 0x44, 0x405, 0, , SomeOtherScript.ahk - AutoHotkey v

MsgBox %ErrorLevel% is the process id.

```

28. 1. 13 SetControlDelay

设置两次控件操作类命令之间的延时。

`SetControlDelay, Delay`

参数

Delay	延时时间，单位毫秒。可以是一个 expression/表达式(See 9.) 。使用 <code>-1</code> 表示无延时，使用 <code>0</code> 表示最小延时。如果没有设置，默认延时 <code>20</code> 。
-------	---

注意

脚本在执行了每个控件操作类命令之后，会有一个自动的延时(休眠)。这些命令包括：[Control\(See 28.1.1\)](#)，[ControlMove\(See 28.1.8\)](#)，[ControlClick\(See 23.2\)](#)，[ControlFocus\(See 28.1.3\)](#) 以及 [ControlSetText\(See 28.1.10\)](#)（[ControlSend\(See 20.6\)](#) 例外，它的延时使用 [SetKeyDelay\(See 20.14\)](#) 命令进行设置）。这样做的目的是提高脚本的可靠性，因为一个控件一般情况下在两次操作之间都需要一个短暂的“休息”来进行刷新，好对下一个可能的命令进行响应。

虽然允许使用 `-1`（完全无延时），但是推荐最少只设置到 `0`，这样可以增加脚本正确执行的几率。

设置延时为 `0` 的话相当于执行了命令 `Sleep(0)`，它会将当前脚本的剩余时间片分配给有需要的进程。如果没有进程需要，延时 `0` 就相当于完全没有延时。

cpu 速度比较慢，或者 cpu 正忙，或者开启了窗口动画的时候，也许需要设置比较大的延时。

内置变量 **A_ControlDelay** 保存了当前的设置。

每一个新运行的 [Thread/线程\(See 30.19\)](#) (例如一个 [hotkey/热键\(See 4.\)](#), [custom menu item/自定义菜单\(See 22.13\)](#), 或 [timed/定时器\(See 17.21\)](#) 事件) 会将该命令的设置重置为默认值。要更改该命令的默认值，可以将该命令放在脚本的自动执行区域（脚本的顶部）。

相关命令

[Control\(See 28.1.1\)](#), [ControlMove\(See 28.1.8\)](#), [ControlClick\(See 23.2\)](#), [ControlFocus\(See 28.1.3\)](#), [ControlSetText\(See 28.1.10\)](#), [SetWinDelay\(See 28.9\)](#), [SetKeyDelay\(See 20.14\)](#), [SetMouseDelay\(See 23.8\)](#), [SetBatchLines\(See 17.20\)](#)

示例

```
SetControlDelay, 0
```

28.1.14 WinMenuItemSelect

调用匹配指定条件的窗口的菜单项。

`WinMenuItemSelect, WinTitle, WinText, Menu [, SubMenu1, SubMenu2, SubMenu3, SubMenu4, SubMenu5, SubMenu6, ExcludeTitle, ExcludeText]`

参数

WinTitle	目标窗口的标题或标题中的部分文字（匹配模式由 SetTitleMatchMode(See 28.8) 决定）。如果省略其它 3 个窗口参数，默认目标是 上一次匹配窗口(See 30.2) 。如果这个参数使用字母 A，同时省略其它 3 个窗口参数，则以当前激活的窗口作为目标。要用窗口的 class 名进行匹配，使用 <code>ahk_class</code> 精确 class 名 (Window Spy 中可以显示 class 名)。要用窗口的 进程标识符(PID)(See 24.4) 进行匹配，使用 <code>ahk_pid %PID 变量%</code> 。要用 窗口组(See 28.2.2) ，使用 <code>ahk_group</code> 组名。要用窗口的 唯一 ID(See 28.15) 进行匹配，使用 <code>ahk_id %ID 变量%</code> 。要减小检测范围，使用 多重条件(See 30.2) ，例如：My File.txt ahk_class Notepad
WinText	如果使用这个参数，则它应该是目标窗口中某个文本元素的子字符串 (在 Window Spy 中会显示出窗口中的文本元素)。隐藏文本只有在 DetectHiddenText(See 28.4) 设置为 ON 的时候才能检测到。
Menu	顶级菜单的名称，例如 File , Edit , View 。也可以使用 1& 表示第一个菜单项，2& 表示第二个菜单项，等等。
SubMenu1	需要选择的菜单的名称或位置 (参见 <code>Menu</code>)。
SubMenu2	如果 <code>SubMenu1</code> 本身还有子菜单，那么这个参数就是子菜单项的名称或位置。
SubMenu3	同上。
SubMenu4	同上。

SubMenu5	同上。
SubMenu6	同上。
ExcludeTitle	标题中包含该参数指定的文字的窗口将被除外。
ExcludeText	文本元素中包含该参数指定的文字的窗口将被除外。

ErrorLevel

如果出现错误, [ErrorLevel/错误等级\(See 30.8\)](#) 设置为 1 , 否则为 0 。

注意

使用这个命令的时候, 目标窗口不需要处于激活状态。但是, 某些窗口也许要求处于 [非最小化\(See 28.28\)](#) 状态。

这个命令对于使用非标准菜单栏的程序 **无效** 。例如 Microsoft Outlook 和 Outlook Express , 它们使用工具栏来伪装成菜单栏。在这种情况下, 推荐使用 [ControlSend\(See 20.6\)](#) 或 [PostMessage\(See 28.1.12\)](#) 命令。

菜单的名称是不区分大小写的 (例如, File->Save 等同于 file->save) , 同时也不需要使用“&”来表示菜单名称中的下划线字母 (例如, &File 等同于 File) 。

在菜单名称参数中同样可以指定菜单位置, 这样做是为了支持那些没有文本的菜单 (例如仅包含图片而没有文字的菜单项) 。1& 表示第一个菜单 (例如 File) , 2& 表示第二个菜单 (例如 Edit) , 以此类推。菜单中的分割线也算一个菜单项, 在参数中使用菜单位置的时候要注意。

窗口的标题和窗口中的文字是大小写敏感的。要检测隐藏窗口, 必须打开 [DetectHiddenWindows\(See 28.5\)](#) 。

相关命令

[ControlSend\(See 20.6\)](#), [PostMessage\(See 28.1.12\)](#)

示例

; 选择记事本中的“打开”菜单:

```
WinMenuSelectItem, Untitled - Notepad, , File, Open
```

; 和上面例子一样, 不过使用的是菜单位置:

```
WinMenuSelectItem, Untitled - Notepad, , 1&, 2&
```

28.2 Window Groups

28.2.1 GroupActivate

Activates the next window in a window group that was defined with [GroupAdd](#)(See 28.2.2).

`GroupActivate, GroupName [, R]`

参数

GroupName	The name of the group to activate, as originally defined by GroupAdd (See 28.2.2).
R	This determines whether the oldest or the newest window is activated whenever no members of the group are currently active. If omitted, the oldest window is always activated. If it's the letter R, the newest window (the one most recently active) is activated, but only if no members of the group are active when the command is given. "R" is useful in cases where you temporarily switch to working on an unrelated task. When you return to the group via GroupActivate (See 28.2.1), GroupDeactivate (See 28.2.4), or GroupClose (See 28.2.3), the window you were most recently working with is activated rather than the oldest window.

注意

This command causes the first window that matches any of the group's window specifications to be activated. Using it a second time will activate the next window in the series and so on. Normally, it is assigned to a hotkey so that this window-traversal behavior is automated by pressing that key.

When a window is activated immediately after another window was activated, task bar buttons may start flashing on some systems (depending on OS and settings). To prevent this, use [#WinActivateForce](#)(See 28.3).

See [GroupAdd](#)(See 28.2.2) for more details about window groups.

相关命令

[GroupAdd](#)(See 28.2.2), [GroupDeactivate](#)(See 28.2.4), [GroupClose](#)(See 28.2.3),
[#WinActivateForce](#)(See 28.3)

示例

```
GroupActivate, MyGroup, R
```

28.2.2 GroupAdd

Adds a window specification to a window group, creating the group if necessary.

GroupAdd, GroupName [, WinTitle, WinText, Label, ExcludeTitle, ExcludeText]

参数

GroupName	The name of the group to which to add this window specification. If the group doesn't exist, it will be created. Group names are not case sensitive.
WinTitle	<p>The title or partial title of the target window(s). It can be blank. Note: Although SetTitleMatchMode(See 28.8) and DetectHiddenWindows(See 28.5) do not directly affect the behavior of this command, they do affect the other group commands such as GroupActivate(See 28.2.1) and GroupClose(See 28.2.3). They also affect the use of ahk_group in any other command's <i>WinTitle</i>.</p> <p>To use a window class, specify ahk_class ExactClassName (shown by Window Spy). To use a process identifier (PID)(See 24.4), specify ahk_pid %VarContainingPID%.</p> <p>To use a window's unique ID number(See 28.15), specify ahk_id %VarContainingID%. To use a window group, specify ahk_group GroupName (i.e. groups may contain other groups).</p> <p>The search can be narrowed by specifying multiple criteria(See 30.2). For example: <i>My File.txt ahk_class Notepad</i></p>
WinText	If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if DetectHiddenText (See 28.4) is ON at the time that GroupActivate (See 28.2.1), GroupDeactivate (See 28.2.4), and GroupClose (See 28.2.3) are used.
Label	The label of a subroutine to run if no windows matching this specification exist when the GroupActivate (See 28.2.1) command is used. The label is jumped to as though a Gosub (See 17.8) had been used. Omit or leave blank for none.
ExcludeTitle	Windows whose titles include this value will not be considered.
ExcludeText	Windows whose text include this value will not be considered.

注意

Each use of this command adds a new rule to a group. In other words, a group consists of a set of criteria rather than a fixed list of windows. Later, when a group is used by a command such as [GroupActivate](#)(See 28.2.1), each window on the desktop is checked against each of these criteria. If a window matches one of the criteria in the group, it is considered a match.

A window group is typically used to bind together a collection of related windows, which is useful for tasks that involve many related windows, or an application that owns many subwindows. For example, if you frequently work with many instances of a graphics program or

text editor, you can use [GroupActivate](#)(See 28.2.1) on a hotkey to visit each instance of that program, one at a time, without having to use alt-tab or task bar buttons to locate them.

Since the entries in each group need to be added only once, this command is typically used in the auto-execute section (top part of the script). Attempts to add duplicate entries to a group are ignored.

To include all windows in a group (except the special Program Manager window), use this example:

GroupAdd, AllWindows

All windowing commands can operate upon a window group by specifying *ahk_group MyGroupName* for the *WinTitle* parameter. The commands [WinMinimize](#)(See 28.25), [WinMaximize](#)(See 28.24), [WinRestore](#)(See 28.28), [WinHide](#)(See 28.22), [WinShow](#)(See 28.31), [WinClose](#)(See 28.14), and [WinKill](#)(See 28.23) will act upon **all** the group's windows. To instead act upon only the topmost window, follow this example:

```
WinHide % "ahk_id " . WinExist("ahk_group MyGroup")
```

By contrast, the other window commands such as [WinActivate](#)(See 28.12) and [IfWinExist](#)(See 28.7) will operate only upon the topmost window of the group.

相关命令

[GroupActivate](#)(See 28.2.1), [GroupDeactivate](#)(See 28.2.4), [GroupClose](#)(See 28.2.3)

示例

```
; In the autoexecute section at the top of the script:
GroupAdd, MSIE, ahk_class IFrame ; Add only Internet Explorer windows to this group.
return ; End of autoexecute section.

; Assign a hotkey to activate this group, which traverses
; through all open MSIE windows, one at a time (i.e. each
; press of the hotkey).

Numpad1::GroupActivate, MSIE, r

; Here's a more complex group for MS Outlook 2002.
; In the autoexecute section at the top of the script:
SetTitleMatchMode, 2
```

```
GroupAdd, mail, Message - Microsoft Word ; This is for mails currently being composed
```

```
GroupAdd, mail, - Message ( ; This is for already opened items
```

```
; Need extra text to avoid activation of a phantom window:
```

```
GroupAdd, mail, Advanced Find, Sear&ch for the word(s)
```

```
GroupAdd, mail, , Recurrence:
```

```
GroupAdd, mail, Reminder
```

```
GroupAdd, mail, - Microsoft Outlook
```

```
return ; End of autoexecute section.
```

```
Numpad5::GroupActivate, mail ; Assign a hotkey to visit each Outlook window, one at a time.
```

28.2.3 GroupClose

Closes the active window if it was just activated by [GroupActivate](#)(See 28.2.1) or [GroupDeactivate](#)(See 28.2.4). It then activates the next window in the series. It can also close all windows in a group.

`GroupClose, GroupName [, A|R]`

参数

GroupName	The name of the group as originally defined by GroupAdd (See 28.2.2).
A R	If it's the letter A, all members of the group will be closed. This is the same effect as WinClose (See 28.14) <code>ahk_group GroupName</code> . Otherwise: If the command closes the active window, it will then activate the next window in the series. This parameter determines whether the oldest or the newest window is activated. If omitted, the oldest window is always activated. If it's the letter R, the newest window (the one most recently active) is activated, but only if no members of the group are active when the command is given. "R" is useful in cases where you temporarily switch to working on an unrelated task. When you return to the group via GroupActivate (See 28.2.1), GroupDeactivate (See 28.2.4), or GroupClose (See 28.2.3), the window you were most recently working with is activated rather than the oldest window.

注意

When the *A|R* parameter is not "A", the behavior of this command is determined by whether the previous action on *GroupName* was [GroupActivate](#)(See 28.2.1) or [GroupDeactivate](#)(See 28.2.4). If it was [GroupDeactivate](#)(See 28.2.4), this command will close the active window only if it is **not** a member of the group (otherwise it will do nothing). If it was [GroupActivate](#)(See 28.2.1) or nothing, this command will close the active window only if it **is** a member of the group (otherwise it will do nothing). This behavior allows [GroupClose](#) to be assigned to a hotkey as a companion to *GroupName*'s [GroupActivate](#)(See 28.2.1) or [GroupDeactivate](#)(See 28.2.4) hotkey.

See [GroupAdd](#)(See 28.2.2) for more details about window groups.

相关命令

[GroupAdd](#)(See 28.2.2), [GroupActivate](#)(See 28.2.1), [GroupDeactivate](#)(See 28.2.4)

示例

```
GroupClose, MyGroup, R
```

28.2.4 GroupDeactivate

Similar to [GroupActivate](#)(See 28.2.1) except activates the next window **not** in the group.

`GroupDeactivate, GroupName [, R]`

参数

GroupName	The name of the target group, as originally defined by GroupAdd (See 28.2.2).
R	This determines whether the oldest or the newest non-member window is activated whenever a member of the group is currently active. If omitted, the oldest non-member window is always activated. If it's the letter R, the newest non-member window (the one most recently active) is activated, but only if a member of the group is active when the command is given. "R" is useful in cases where you temporarily switch to working on an unrelated task. When you return to the group via GroupActivate (See 28.2.1), GroupDeactivate (See 28.2.4), or GroupClose (See 28.2.3), the window you were most recently working with is activated rather than the oldest window.

注意

[GroupDeactivate](#) causes the first window that does **not** match any of the group's window specifications to be activated. Using [GroupDeactivate](#) a second time will activate the next window in the series and so on. Normally, [GroupDeactivate](#) is assigned to a hotkey so that this window-traversal behavior is automated by pressing that key.

This command is useful in cases where you have a collection of favorite windows that are almost always running. By adding these windows to a group, you can use GroupDeactivate to visit each window that isn't one of your favorites and decide whether to close it. This allows you to clean up your desktop much more quickly than doing it manually.

See [GroupAdd](#)(See 28.2.2) for more details about window groups.

相关命令

[GroupAdd](#)(See 28.2.2), [GroupActivate](#)(See 28.2.1), [GroupClose](#)(See 28.2.3)

示例

```
GroupDeactivate, MyFavoriteWindows ; Visit non-favorite windows to clean up desktop.
```

28.3 #WinActivateForce

跳过温和的方法激活窗口而直接使用强硬的方法。

#WinActivateForce

在脚本的任意位置指定此指令将促使激活窗口的命令 -- 例如 [WinActivate](#)(See 28.12), [WinActivateBottom](#)(See 28.13), 和 [GroupActivate](#)(See 28.2.1) -- 跳过 "温和的" 方法激活窗口而直接使用更强硬的方法。

虽然这个指令通常不会改善如何快速地或可靠地激活窗口，但它可以防止任务栏按钮在不同窗口接连快速地激活时的闪烁。

Windows 95 和 NT 将可能永不需要此设置，因为它们在准许窗口被激活方面更自由。

相关命令

[WinActivate](#)(See 28.12), [WinActivateBottom](#)(See 28.13), [GroupActivate](#)(See 28.2.1)

示例

```
#WinActivateForce
```

翻译：天堂之门 menk33@163.com 2008 年 9 月 5 日

28.4 DetectHiddenText

决定窗口中隐藏的文本是否在探测窗口时“可见”。这会影响像 IfWinExist 和 WinActivate 这样的命令。

DetectHiddenText, On|Off

参数

On Off	On: 非 AutoIt v2 脚本中的默认值 (例如 .ahk, .ini)。将探测隐藏的文本。 Off: AutoIt v2 脚本中的默认值。不探测隐藏的文本。
--------	---

注意

"Hidden text"是一个术语，和窗口中那些不可见的控件有关。它们的文本因而被认为是“隐藏的”。当你要探测多标签对话框或多窗格窗口不同的窗格之间的差异的时候，将 `DetectHiddenText` 设为 Off 是很有用的。用 `Window Spy` 来测定当前活动窗口中哪些文本是隐藏的。所有接受 `WinText` 参数的命令都受到该设置影响，包括 [WinActivate\(See 28.12\)](#), [IfWinActive\(See 28.6\)](#), [WinWait\(See 28.32\)](#) 和 [IfWinExist\(See 28.7\)](#)。

内置变量 **A_DetectHiddenText** 包含当前的设置 (On 或 Off)。

每一个新运行的 [线程\(See 30.19\)](#) (例如一个 [热键\(See 4.\)](#), [自定义菜单项\(See 22.13\)](#), 或者 [定时的\(See 17.21\)](#) 子程序) 都会以本命令的默认设置开始。这一默认值可以通过将本命令放置于自动运行区段 (位于脚本的顶部)来改变。

相关命令

[DetectHiddenWindows\(See 28.5\)](#)

示例

```
DetectHiddenText, off
```

翻译: tcgbp 修正: 天堂之门 menk33@163.com 2008 年 9 月 18 日

28.5 DetectHiddenWindows

决定不可见的窗口是否被脚本“看见”。

`DetectHiddenWindows, On|Off`

参数

On Off	On: 探测隐藏的窗口。 Off: 这参数是默认的。不探测隐藏的窗口，除了通过 WinShow(See 28.31) 命令。
--------	---

注意

在一些情况下开启 `DetectHiddenWindows` 能使脚本运行更艰难，因为有些隐藏的系统窗口可能偶然地匹配你正在试着去一同工作的另一个窗口的标题或文本。因此大多数脚本应该让这个设置关闭。不过假如你希望直接地和隐藏窗口一同工作而不是先使用 [WinShow\(See 28.31\)](#) 命令来反隐藏它们，那么开启它也许会很有用。

所有窗口命令除了 [WinShow](#)(See 28.31) 都被此设置影响，包括 [WinActivate](#)(See 28.12)、[IfWinActive](#)(See 28.6)、[WinWait](#)(See 28.32)、[IfWinExist](#)(See 28.7)。相比之下，[WinShow](#)(See 28.31) 将总是能反隐藏一个隐藏的窗口，即使隐藏的窗口没有被探测到。

当通过 [ahk_id](#) 方法(See 30.2) 或像 [last-found-window](#)(See 30.2)(最后找到的窗口)来访问一个子窗口或控件时，没有必要开启 [DetectHiddenWindows](#)。当通过 [Gui +LastFound](#)(See 19.3) 来访问 GUI 窗口时它也是没必要的。

内置变量 **A_DetectHiddenWindows** 包含当前的设置(On 或者 Off)。

对于此命令来说，每个最近启动的 [thread](#)(See 30.19) (例如一个 [hotkey](#)(See 4.)、[custom menu item](#)(See 22.13) 或 [timed](#)(See 17.21) 子程序) 以新的默认设置开始。这个默认设置可以通过在自动执行部分(脚本的顶部)使用此命令来改变。

相关命令

[DetectHiddenText](#)(See 28.4)

示例

```
DetectHiddenWindows, on
```

翻译：天堂之门 menk33@163.com 2008 年 8 月 12 日

28.6 IfWinActive/IfWinNotActive

检查指定的窗口是否存在和当前是否激活(在最前面)。

```
IfWinActive [, WinTitle, WinText, ExcludeTitle, ExcludeText]
IfWinNotActive [, WinTitle, WinText, ExcludeTitle, ExcludeText]
UniqueID(See 28.15) := WinActive("WinTitle", "WinText", "ExcludeTitle", "ExcludeText")
```

参数

WinTitle	目标窗口的标题或部分标题(匹配模式由 SetTitleMatchMode (See 28.8) 来决定)。如果省略此参数和其他 3 个参数， Last Found Window (See 30.2)(最后找到的窗口) 将被使用。要使用一个窗口类，指定 ahk_class 确切的类名 (通过 Window Spy 显示)。要使用一个 process identifier (PID) (See 24.4)(进程标识符)，指定 ahk_pid % 包含 PID 的变量%。要使用一个 window group (See 28.2.2)(窗口组)，指定 ahk_group GroupName。要使用一个窗口的 unique ID number (See 28.15)(唯一标识符编号)，指定 ahk_id % 包含 ID 的变量%。通过指定 multiple criteria (See 30.2)(多个条件) 缩小搜索范围。例如：My File.txt ahk_class Notepad
WinText	如果用到，此参数必须是目标窗口的一个单独文本对象的子串(像内置的 Window Spy 工具显示的一样)。如果 DetectHiddenText (See 28.4) 是 ON 的状态，隐藏的文本对象将被探测。
ExcludeTitle	标题含有此参数值的窗口将不被考虑。注意：由于反向兼容 .aut 脚本 (See 11.)，如果此

	参数刚好符合一个命令的名称，它会被解释为一个命令。要绕弯解决这种情况，使用 WinActive() 函数 (See 10.) 来代替此命令。
ExcludeText	文本含有此参数值的窗口将不被考虑。

注意

如果省略所有参数，[Last Found Window](#)(See 30.2) 将被使用。

如果这两个命令中任何一个确定激活的窗口是一个合格的匹配，[Last Found Window](#)(See 30.2) 将更新为激活的窗口。换句话说，如果 *IfWinActive* 赋值为“真”或者 *IfWinNotActive* 赋值为“假”，[Last Found Window](#)(See 30.2) 将被更新。

WinActive() 函数返回激活窗口的 [Unique ID \(HWND\)](#)(See 28.15)，如果窗口匹配指定的条件。假如不匹配，函数返回 0。由于所有非零数字被视为“真”，在 *WinTitle* 是激活的时候 *if WinActive("WinTitle")* 语句为真。

[SetWinDelay](#)(See 28.9) 不适用于 *IfWinExist/IfWinActive*。

窗口标题和文本是区分大小写的。隐藏的窗口将不被探测，除非 [DetectHiddenWindows](#)(See 28.5) 已被打开。

相关命令

[IfWinExist](#)(See 28.7), [SetTitleMatchMode](#)(See 28.8), [DetectHiddenWindows](#)(See 28.5), [Last Found Window](#)(See 30.2), [WinActivate](#)(See 28.12), [WinWaitActive](#)(See 28.33), [WinWait](#)(See 28.32), [WinWaitClose](#)(See 28.34), [#IfWinActive/Exist](#)(See 20.1.4)

示例

```
IfWinActive, 无标题 - 记事本
{
    WinMaximize ; 最大化由上面的 IfWinActive 找到的记事本窗口。
    Send, Some text.{Enter}
    return
}

if WinActive("ahk_class Notepad") or WinActive("ahk_class" . ClassName) ; "ahk_class"
后面不需要带空格。
WinClose ; 使用 last found window(See 30.2)。
```

翻译：天堂之门 menk33@163.com 2008 年 8 月 12 日

28.7 IfWinExist/IfWinNotExist

检查指定的窗口是否存在。

`IfWinExist [, WinTitle, WinText, ExcludeTitle, ExcludeText]`

`IfWinNotExist [, WinTitle, WinText, ExcludeTitle, ExcludeText]`

`UniqueID`(See 28.15) := `WinExist("WinTitle", "WinText", "ExcludeTitle", "ExcludeText")`

参数

WinTitle	目标窗口的标题或部分标题(匹配模式由 SetTitleMatchMode (See 28.8) 来决定)。如果省略此参数和其他 3 个参数, Last Found Window (See 30.2)(最后找到的窗口) 将被使用(即检查指定窗口看它是否还存在)。要使用一个窗口类, 指定 <code>ahk_class</code> 确切的类名(通过 Window Spy 显示)。要使用一个 process identifier (PID) (See 24.4)(进程标识符), 指定 <code>ahk_pid</code> %包含 PID 的变量%。要使用一个 window group (See 28.2.2)(窗口组), 指定 <code>ahk_group</code> <code>GroupName</code> 。要使用一个窗口的 unique ID number (See 28.15)(唯一标识符编号), 指定 <code>ahk_id</code> %包含 ID 的变量%。通过指定 multiple criteria (See 30.2)(多个条件)缩小搜索范围。例如: <code>My File.txt ahk_class Notepad</code>
WinText	如果用到, 此参数必须是目标窗口的一个单独文本对象的子字符串(像内置的 Window Spy 工具显示的一样)。如果 DetectHiddenText (See 28.4) 是 ON 的状态, 隐藏的文本对象将被探测。
ExcludeTitle	标题含有此参数值的窗口将不被考虑。注意: 由于反向兼容 .aut 脚本 (See 11.) , 如果此参数刚好符合一个命令的名称, 它会被解释为一个命令。要绕弯解决这种情况, 使用 WinExist() 函数 (See 10.) 来代替此命令。
ExcludeText	文本含有此参数值的窗口将不被考虑。

注意

如果省略所有参数, [Last Found Window](#)(See 30.2) 将被检查看它是否还存在(或者在使用 `IfWinNotExist` 时看它是否不存在)。

如果这两个命令中任何一个确定有一个合格的窗口存在, [Last Found Window](#)(See 30.2) 将更新为那个窗口。换句话说, 如果 `IfWinExist` 赋值为“真”或者 `IfWinNotExist` 赋值为“假”, 最后找到的窗口将被更新。

`WinExist()` 函数返回首个匹配窗口的 [Unique ID \(HWND\)](#)(See 28.15) (如果无匹配将为 0)。由于所有非零数字被视为“真”, 在 `WinTitle` 存在时 `if WinExist("WinTitle")` 语句为真。

要探索一个控件的 HWND (为了和 [Post/SendMessage](#)(See 28.1.12) 或 [DIICall](#)(See 18.3) 一起使用), 请用 [ControlGet Hwnd](#)(See 28.1.4) 或者 [MouseGetPos](#)(See 23.5) 。

[SetWinDelay](#)(See 28.9) 不适用于 `IfWinExist/IfWinActive` 。

窗口标题和文本是区分大小写的。隐藏的窗口将不被探测, 除非 [DetectHiddenWindows](#)(See 28.5) 已被打开。

相关命令

[IfWinActive\(See 28.6\)](#), [SetTitleMatchMode\(See 28.8\)](#), [DetectHiddenWindows\(See 28.5\)](#), [Last Found Window\(See 30.2\)](#), [Process\(See 24.4\)](#), [WinActivate\(See 28.12\)](#), [WinWaitActive\(See 28.33\)](#), [WinWait\(See 28.32\)](#), [WinWaitClose\(See 28.34\)](#), [#IfWinActive/Exist\(See 20.1.4\)](#)

示例

```
IfWinExist, 无标题 - 记事本
{
    WinActivate ; 自动地使用上面找到的窗口。
    WinMaximize ; 同上
    Send, Some text.{Enter}
    return
}

IfWinNotExist, 计算器
{
    return
}
else
{
    WinActivate ; 上面的 "IfWinNotExist" 也为设置了 "last found" window(最后找到的窗口)。
    WinMove, 40, 40 ; 将它移动到一个新位置。
    return
}

if WinExist("ahk_class Notepad") or WinExist("ahk_class" . ClassName)
{
    WinActivate ; 使用 last found window(See 30.2) 。
}

MsgBox % "激活窗口的 ID 是 " . WinExist("A")
```

翻译：天堂之门 menk33@163.com 2008 年 8 月 12 日

28.8 SetTitleMatchMode

设置在例如 [WinWait](#)(See 28.32) 这样的命令中，参数 `WinTitle` 的匹配模式。

`SetTitleMatchMode`, `MatchMode`

`SetTitleMatchMode`, `Fast|Slow`

参数

MatchMode	<p>使用下面所示的一个数字或单词 <code>RegEx</code> :</p> <p>1 : 匹配由指定的 <code>WinTitle</code> 开始的窗口标题。</p> <p>2 : 匹配包含指定的 <code>WinTitle</code> 的窗口标题。</p> <p>3 : 精确匹配。窗口的标题必须完全和 <code>WinTitle</code> 一样。</p> <p>RegEx(v1.0.45+) : 切换参数 <code>WinTitle</code> , <code>WinText</code>, <code>ExcludeTitle</code> 以及 <code>ExcludeText</code> 为 正则表达式(See 30.14) 匹配模式。当使用正则表达式的时候，不用在正则表达式的外面加引号。例如: WinActivate Untitled.*Notepad(See 28.12) 。<code>RegEx</code> 同样适用于 <code>ahk_group</code>(See 30.2) 和 <code>ahk_class</code>(See 30.2) 。例如 <code>ahk_class IFrame</code> 匹配一个类名中任意位置包含 <code>IFrame</code> 的窗口（这是因为默认情况下，正则表达式会在目标字符串中的 任意位置 搜索匹配值）。注意：对 <code>WinText</code> 参数来说，正则表达式会独立的匹配每个文本元素（例如每个控件的文本），因此，正则表达式的匹配范围不能横跨多个文本元素。</p> <p>上面的模式设置同样影响 <code>ExcludeTitle</code> 参数。例如，模式 3 要求被排除的窗口的标题精确匹配 <code>ExcludeTitle</code> 中指定的名称。</p>
Fast Slow	<p>Fast: 快速模式，这个是默认值。使用 <code>Fast</code> 的性能比使用 <code>Slow</code> 要好得多，但是对于窗口操作类命令中的 <code>WinText</code> 参数，某些类型的窗口中的文本元素可能不能被检测到。</p> <p>Slow: 慢速模式，速度会比较慢，但是 <code>WinText</code> 可以匹配到窗口中所有可能的文本元素。Window Spy 中显示在“slow Hidden Text”部分的文本都要求使用慢速模式来匹配。</p>

注意

这个命令影响所有的窗口操作类命令，例如 [IfWinExist](#)(See 28.7) 和 [WinActivate](#)(See 28.12) 。

如果未指定，默认使用 1 和 `fast` 。

一般来说，只有在用窗口标题或者快速模式的 `WinText` 参数搜索不到窗口的时候才使用慢速模式。因为当任一个窗口正忙或者无响应的时候慢速模式会变得相当的慢。

AutoHotkey 自带的 Window Spy 将需要使用慢速模式检测的文本元素放在一个独立的区域中显示，这样就可以很容易的决定是否使用慢速模式。

如果你要同时修改两个参数，可以使用这个命令两次：

`SetTitleMatchMode`, 2

`SetTitleMatchMode`, `slow`

内置变量 **A_TitleMatchMode** 和 **A_TitleMatchModeSpeed** 存储了当前的设置。

与当前的 **TitleMatchMode** 设置无关, **WinTitle**, **WinText**, **ExcludeTitle** 以及 **ExcludeText** 都是大小写敏感的。但是有一个例外, 就是正则表达式中 [大小写敏感设置](#)(See 18.12), 例如: *i)untitled - notepad*

每一个新运行的 [Thread/线程](#)(See 30.19) (例如一个 [hotkey/热键](#)(See 4.), [custom menu item/自定义菜单](#)(See 22.13), 或 [timed/定时器](#)(See 17.21) 事件) 会将该命令的设置重置为默认值。要更改该命令的默认值, 可以将命令放在脚本的自动执行区域 (脚本的顶部)。

相关命令

[SetWinDelay](#)(See 28.9), [IfWinExist](#)(See 28.7), [WinActivate](#)(See 28.12), [RegExMatch\(\)](#)(See 18.12)

示例

```
SetTitleMatchMode 2
```

; 或者:

```
SetTitleMatchMode RegEx
```

SetTitleMatchMode Slow ; Slow/Fast 可以在其它的模式之外单独进行设置。

28.9 SetWinDelay

设置两个窗口操作类命令间的延时, 例如 [WinActivate](#)(See 28.12)。

`SetWinDelay, Delay`

参数

Delay	延时时间, 单位毫秒。可以是一个 expression/表达式 (See 9.)。使用 <code>-1</code> 表示无延时, 使用 <code>0</code> 表示最小延时。如果没有设置, 默认延时 <code>100</code> 。
-------	---

注意

脚本在执行了每个窗口操作类命令之后, 会有一个自动的延时(休眠), 除了 [IfWinActive](#)(See 28.6) 和 [IfWinExist](#)(See 28.7)。这样做的目的是提高脚本的可靠性, 因为一个窗口在创建、激活、最小化或其它操作之后有时候需要一个简短的“休息”来进行刷新, 好对下一个可能的命令进行响应。

虽然允许使用 `-1` (完全无延时), 但是推荐最少只设置到 `0`, 这样可以增加脚本正确执行的几率。

设置延时为 `0` 的话相当于执行了命令 `Sleep(0)`, 它会将当前脚本的剩余时间片分配给有需要的进程。如果没有进程需要, 延时 `0` 就相当于完全没有延时。

cpu 速度比较慢, 或者 cpu 正忙, 或者开启了窗口动画的时候, 也许需要设置比较大的延时。

内置变量 **A_WinDelay** 保存了当前的设置。

每一个新运行的 [Thread/线程\(See 30.19\)](#) (例如一个 [hotkey/热键\(See 4.\)](#), [custom menu item/自定义菜单\(See 22.13\)](#), 或 [timed/定时器\(See 17.21\)](#) 事件) 会将该命令的设置重置为默认值。要更改该命令的默认值, 可以将该命令放在脚本的自动执行区域 (脚本的顶部)。

相关命令

[SetControlDelay\(See 28.1.13\)](#), [SetKeyDelay\(See 20.14\)](#), [SetMouseDelay\(See 23.8\)](#),
[SetBatchLines\(See 17.20\)](#), [SendMode\(See 20.13\)](#)

示例

```
SetWinDelay, 10
```

28.10 StatusBarGetText

Retrieves the text from a standard status bar control.

`StatusBarGetText, OutputVar [, Part#, WinTitle, WinText, ExcludeTitle, ExcludeText]`

参数

OutputVar	The name of the variable in which to store the retrieved text.
Part#	Which part number of the bar to retrieve, which can be an expression(See 9.) . Default 1, which is usually the part that contains the text of interest.
WinTitle	The title or partial title of the target window (the matching behavior is determined by SetTitleMatchMode(See 28.8)). If this and the other 3 window parameters are blank or omitted, the Last Found Window(See 30.2) will be used. If this is the letter A and the other 3 window parameters are blank or omitted, the active window will be used. To use a window class, specify <code>ahk_class</code> <code>ExactClassName</code> (shown by Window Spy). To use a process identifier (PID)(See 24.4) , specify <code>ahk_pid %VarContainingPID%</code> . To use a window group(See 28.2.2) , specify <code>ahk_group</code> <code>GroupName</code> . To use a window's unique ID number(See 28.15) , specify <code>ahk_id %VarContainingID%</code> . The search can be narrowed by specifying multiple criteria(See 30.2) . For example: <i>My File.txt ahk_class Notepad</i>
WinText	If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if DetectHiddenText(See 28.4) is ON.
ExcludeTitle	Windows whose titles include this value will not be considered.
ExcludeText	Windows whose text include this value will not be considered.

ErrorLevel

[ErrorLevel](#)(See 30.8) is set to 1 if there is a problem or 0 otherwise. If there was a problem, [OutputVar](#) is also made blank.

注意

This command attempts to read the first *standard* status bar on a window (Microsoft common control: msctls_statusbar32). Some programs use their own status bars or special versions of the MS common control, in which case the text cannot be retrieved.

Rather than using this command in a loop, it is usually more efficient to use [StatusBarWait](#), (See 28.11)which contains optimizations that avoid the overhead of repeated calls to [StatusBarGetText](#).

Window titles and text are case sensitive. Hidden windows are not detected unless [DetectHiddenWindows](#)(See 28.5) has been turned on.

相关命令

[StatusBarWait](#)(See 28.11), [WinGetTitle](#)(See 28.21), [WinGetText](#)(See 28.20),
[ControlGetText](#)(See 28.1.7)

示例

```
StatusBarGetText, RetrievedText, 1, Search Results
```

```
IfInString, RetrievedText, found, MsgBox, Search results have been found.
```

28.11 StatusBarWait

Waits until a window's status bar contains the specified string.

```
StatusBarWait [, BarText, Seconds, Part#, WinTitle, WinText, Interval, ExcludeTitle, ExcludeText]
```

参数

BarText	The text or partial text for the which the command will wait to appear. Default is blank (empty), which means to wait for the status bar to become blank. The text is case sensitive and the matching behavior is determined by SetTitleMatchMode (See 28.8), similar to <i>WinTitle</i> below. To instead wait for the bar's text to <i>change</i> , either use StatusBarGetText (See 28.10) in a loop, or use the RegEx example at the bottom of this page.
Seconds	The number of seconds (can contain a decimal point or be an expression (See 9.)) to wait before timing out, in which case ErrorLevel (See 30.8) will be set to 1.

	Default is blank, which means wait indefinitely. Specifying 0 is the same as specifying 0.5.
Part#	Which part number of the bar to retrieve, which can be an expression (See 9.). Default 1, which is usually the part that contains the text of interest.
WinTitle	The title or partial title of the target window (the matching behavior is determined by SetTitleMatchMode (See 28.8)). If this and the other 3 window parameters are blank or omitted, the Last Found Window (See 30.2) will be used. If this is the letter A and the other 3 window parameters are blank or omitted, the active window will be used. To use a window class, specify ahk_class ExactClassName (shown by Window Spy). To use a process identifier (PID) (See 24.4), specify ahk_pid %VarContainingPID%. To use a window group (See 28.2.2), specify ahk_group GroupName. To use a window's unique ID number (See 28.15), specify ahk_id %VarContainingID%. The search can be narrowed by specifying multiple criteria (See 30.2). For example: <i>My File.txt ahk_class Notepad</i>
WinText	If present, this parameter must be a substring from a single text element of the target window (as revealed by the included Window Spy utility). Hidden text elements are detected if DetectHiddenText (See 28.4) is ON.
Interval	How often the status bar should be checked while the command is waiting (in milliseconds), which can be an expression (See 9.). Default is 50.
ExcludeTitle	Windows whose titles include this value will not be considered.
ExcludeText	Windows whose text include this value will not be considered.

ErrorLevel

[ErrorLevel](#)(See 30.8) is set to 1 if the command times out before a match could be found in the status bar. It is set to 2 if the status bar could not be accessed. It is set to 0 if a match is found.

注意

StatusBarWait attempts to read the first *standard* status bar on a window (class msctls_statusbar32). Some programs use their own status bars or special versions of the MS common control. Such bars are not supported.

Rather than using [StatusBarGetText](#)(See 28.10) in a loop, it is usually more efficient to use StatusBarWait because it contains optimizations that avoid the overhead that repeated calls to [StatusBarGetText](#)(See 28.10) would incur.

StatusBarWait determines its target window before it begins waiting for a match. If that target window is closed, the command will stop waiting even if there is another window matching the specified WinTitle and WinText.

While the command is in a waiting state, new [threads](#)(See 30.19) can be launched via [hotkey](#)(See 4.), [custom menu item](#)(See 22.13), or [timer](#)(See 17.21).

Window titles and text are case sensitive. Hidden windows are not detected unless [DetectHiddenWindows](#)(See 28.5) has been turned on.

相关命令

[StatusBarGetText](#)(See 28.10), [WinGetTitle](#)(See 28.21), [WinGetText](#)(See 28.20), [ControlGetText](#)(See 28.1.7)

示例

```
; The following example enters a new search pattern into an existing Explorer/Search window.

IfWinExist, Search Results ; Sets the Last Found window to simplify the below.

{
    WinActivate

    Send, {tab 2}!o*.txt{enter} ; In the Search window, enter the pattern to search for.

    Sleep, 400 ; Give the status bar time to change to "Searching".

StatusbarWait, found, 30

    if ErrorLevel

        MsgBox, The command timed out or there was a problem.

    else

        MsgBox, The search successfully completed.

}
```

```
; The following example waits for the status bar of the active window to change. This example requires v1.0.46.06+.

SetTitleMatchMode RegEx(See 28.8)

IfWinExist A ; Set the last-found window to be the active window (for use below).

{
    StatusBarGetText, OrigText

    StatusbarWait, ^(?!\^Q%OrigText%\E$) ; This regular expression waits for any change to the text.
```

{}

28.12 WinActivate

激活匹配指定条件的窗口（将它置于最前端）。

WinActivate [, WinTitle, WinText, ExcludeTitle, ExcludeText]

参数

WinTitle	目标窗口的标题或标题中的部分文字（匹配模式由 SetTitleMatchMode(See 28.8) 决定）。如果省略所有的参数，默认目标是 上一次匹配窗口(See 30.2) 。要用窗口的 class 名进行匹配，使用 <i>ahk_class</i> 精确 class 名（Window Spy 中可以显示 class 名）。要用窗口的 进程标识符 (PID) (See 24.4) 进行匹配，使用 <i>ahk_pid %PID%</i> 变量%。要用 窗口组(See 28.2.2) ，使用 <i>ahk_group</i> 组名。要用窗口的 唯一 ID(See 28.15) 进行匹配，使用 <i>ahk_id %ID%</i> 变量%。要减小检测范围，使用 多重条件(See 30.2) ，例如： <i>My File.txt ahk_class Notepad</i>
WinText	如果使用这个参数，则它应该是目标窗口中某个文本元素的子字符串（在 Window Spy 中会显示出窗口中的文本元素）。隐藏文本只有在 DetectHiddenText(See 28.4) 设置为 ON 的时候才能检测到。
ExcludeTitle	标题中包含该参数指定的文字的窗口将被除外。
ExcludeText	文本元素中包含该参数指定的文字的窗口将被除外。

注意

如果窗口处于最小化状态，它首先会还原到原来的状态，再被激活。

在 60ms 中，脚本会对目标窗口的激活进行 6 次尝试。因此，一般情况下在 WinActivate 命令之后不需要使用 [WinWaitActive\(See 28.33\)](#) 命令。

如果匹配的窗口已经处于激活状态，那么它会继续保持激活状态，脚本不会再去激活其它的匹配窗口。一般情况下，如果匹配的窗口不止一个，处于最前端（最近使用）的窗口将被激活。也可以使用 [WinActivateBottom\(See 28.13\)](#) 激活最底端（最久一次使用）的窗口。

如果一个窗口紧接着另一个窗口的激活事件之后被激活，任务栏可能会闪烁（取决于系统和设置）。要防止这种现象，使用 [#WinActivateForce\(See 28.3\)](#)。

窗口的标题和窗口中的文字是大小写敏感的。要检测隐藏窗口，必须打开 [DetectHiddenWindows\(See 28.5\)](#)。

相关命令

[WinActivateBottom\(See 28.13\)](#), [#WinActivateForce\(See 28.3\)](#), [SetTitleMatchMode\(See 28.8\)](#), [DetectHiddenWindows\(See 28.5\)](#), [Last Found Window\(See 30.2\)](#), [IfWinExist\(See 28.7\)](#), [IfWinActive\(See 28.6\)](#), [WinWaitActive\(See 28.33\)](#), [WinWait\(See 28.32\)](#), [WinWaitClose\(See 28.34\)](#), [WinClose\(See 28.14\)](#), [GroupActivate\(See 28.2.1\)](#), [WinSet\(See 28.29\)](#)

示例

```
IfWinExist, Untitled - Notepad
    WinActivate ; 使用上一句找到的窗口
else
    WinActivate, Calculator
```

28.13 WinActivateBottom

功能和 [WinActivate\(See 28.12\)](#) 一样，只是这个是激活最底端的（至少最近激活的）窗口，而不是最顶端的。

`WinActivateBottom [, WinTitle, WinText, ExcludeTitle, ExcludeText]`

参数

WinTitle	目标窗口的标题或标题中的部分文字（匹配模式由 SetTitleMatchMode(See 28.8) 决定）。要用窗口的 <code>class</code> 名进行匹配，使用 <code>ahk_class</code> 精确 <code>class</code> 名（Window Spy 中可以显示 <code>class</code> 名）。要用窗口的 进程标识符 (PID) (See 24.4) 进行匹配，使用 <code>ahk_pid %PID%</code> 变量%。要用 窗口组(See 28.2.2) ，使用 <code>ahk_group</code> 组名。要用窗口的 唯一 ID(See 28.15) 进行匹配，使用 <code>ahk_id %ID%</code> 变量%。要减小检测范围，使用 多重条件(See 30.2) ，例如：My File.txt ahk_class Notepad
WinText	如果使用这个参数，则它应该是目标窗口中某个文本元素的子字符串（在 Window Spy 中会显示出窗口中的文本元素）。隐藏文本只有在 DetectHiddenText(See 28.4) 设置为 ON 的时候才能检测到。
ExcludeTitle	标题中包含该参数指定的文字的窗口将被除外。
ExcludeText	文本元素中包含该参数指定的文字的窗口将被除外。

注意

如果只有一个匹配窗口，`WinActivateBottom` 完全和 [WinActivate\(See 28.12\)](#) 一样。

[Window groups\(See 28.2.2\)](#) 更加高级和方便，为了提高脚本的功能性和适应性，应该尽量使用它。

如果窗口处于最小化状态，它首先会还原，再被激活。

在 `60ms` 中，脚本会对目标窗口的激活进行 6 次尝试。因此，一般情况下在 `WinActivate` 命令之后不需要使用 [WinWaitActive\(See 28.33\)](#) 命令。

和 [WinActivate\(See 28.12\)](#) 不同，这个命令中不能使用 [上一次匹配窗口\(See 30.2\)](#)，因为它可能不是最底端的窗口。因此，至少要有一个参数不为空。

如果一个窗口紧接着另一个窗口的激活事件之后被激活，任务栏可能会闪烁（取决于系统和设置）。要防止这种现象，使用 [#WinActivateForce\(See 28.3\)](#)。

窗口的标题和窗口中的文字是大小写敏感的。要检测隐藏窗口，必须打开 [DetectHiddenWindows\(See 28.5\)](#)。

相关命令

[WinActivate\(See 28.12\)](#), [#WinActivateForce\(See 28.3\)](#), [SetTitleMatchMode\(See 28.8\)](#),
[DetectHiddenWindows\(See 28.5\)](#), [IfWinExist\(See 28.7\)](#), [IfWinActive\(See 28.6\)](#),
[WinWaitActive\(See 28.33\)](#), [WinWait\(See 28.32\)](#), [WinWaitClose\(See 28.34\)](#),
[GroupActivate\(See 28.2.1\)](#)

示例

```
; 这个快捷键让你以从最老到最新的顺序访问所有打开的浏览器窗口
```

```
#i::  
SetTitleMatchMode, 2  
WinActivateBottom, - Microsoft Internet Explorer  
return
```

28.14 WinClose

关闭匹配指定条件的窗口。

`WinClose [, WinTitle, WinText, SecondsToWait, ExcludeTitle, ExcludeText]`

参数

WinTitle	目标窗口的标题或标题中的部分文字（匹配模式由 SetTitleMatchMode(See 28.8) 决定）。如果省略其它 3 个窗口参数，默认目标是 上一次匹配窗口(See 30.2) 。如果这个参数使用字母 A，同时省略其它 3 个窗口参数，则以当前激活的窗口作为目标。要用窗口的 class 名进行匹配，使用 <code>ahk_class</code> 精确 class 名（Window Spy 中可以显示 class 名）。要用窗口的 进程标识符 (PID) (See 24.4) 进行匹配，使用 <code>ahk_pid %PID%</code> 变量%。要用 窗口组(See 28.2.2) ，使用 <code>ahk_group</code> 组名（此时 <code>WinText</code> ， <code>ExcludeTitle</code> ，以及 <code>ExcludeText</code> 三个变量要省略）。要用窗口的 唯一 ID(See 28.15) 进行匹配，使用 <code>ahk_id %ID%</code> 变量%。要减小检测范围，使用 多重条件(See 30.2) ，例如：My File.txt ahk_class Notepad
WinText	如果使用这个参数，则它应该是目标窗口中某个文本元素的子字符串（在 Window Spy 中会显示出窗口中的文本元素）。隐藏文本只有在 DetectHiddenText(See 28.4) 设置为 ON 的时候才能检测到。
SecondsToWait	如果省略或留空，命令完全不会等待。如果使用 0，会等待 500ms。否则，它会等待指定的时间（可以包含小数点或者是一个 表达式(See 9.) ）来让窗口彻底关闭。如果窗口在指定的时间内并未关闭，脚本会继续执行，但是该命令 不会 设置 <code>ErrorLevel/错误等级</code> 。所以当需要检测一个窗口是否已被关闭的时候，使用 IfWinExist(See 28.7) 或

	WinWaitClose (See 28.34)。当该命令处于等待状态的时候，仍然可以通过 快捷键 (See 4.)， 自定义菜单 (See 22.13)，或 定时器 (See 17.21) 来启动新的 线程 (See 30.19)。
ExcludeTitle	标题中包含该参数指定的文字的窗口将被除外。
ExcludeText	文本元素中包含该参数指定的文字的窗口将被除外。

注意

这个命令传送一个关闭信息给窗口，结果取决于窗口（可能会询问是否保存数据，或其它）。

如果匹配的窗口中有一个处于激活状态，那么它会被关闭，脚本不会再去关闭其它的匹配窗口。一般情况下，如果匹配的窗口不止一个，处于最前端（最近使用）的窗口将被关闭。

这个命令只对匹配窗口中处于最前端的进行操作，除了在 [WinTitle](#) 参数中使用 [ahk_group](#) 组名(See 28.2.2) 的时候，这时候会影响窗口组中的所有窗口。

[WinClose](#) 通过发送一个 [WM_CLOSE](#) 消息给目标窗口来关闭窗口。这个关闭的方法稍微有一点强迫性。另外一个关闭窗口的方法是发送下面那样的消息给窗口。这个方法和 [WinClose](#) 有点不同，它更接近按 [Alt + F4](#) 或者点击标题栏上关闭按钮的效果：

```
PostMessage, 0x112, 0xF060,,, WinTitle, WinText ; 0x112 = WM_SYSCOMMAND,  
0xF060 = SC_CLOSE
```

用 [WinClose](#) 不能关闭的窗口，可以使用 [WinKill](#)(See 28.23) 强制关闭。

窗口的标题和窗口中的文字是大小写敏感的。要检测隐藏窗口，必须打开 [DetectHiddenWindows](#)(See 28.5)。

相关命令

[WinKill](#)(See 28.23), [WinWaitClose](#)(See 28.34), [Process](#)(See 24.4), [WinActivate](#)(See 28.12), [SetTitleMatchMode](#)(See 28.8), [DetectHiddenWindows](#)(See 28.5), [Last Found Window](#)(See 30.2), [IfWinExist](#)(See 28.7), [IfWinActive](#)(See 28.6), [WinWaitActive](#)(See 28.33), [WinWait](#)(See 28.32), [GroupActivate](#)(See 28.2.1)

示例

```
IfWinExist, Untitled - Notepad  
    WinClose ; 使用上面找到的窗口  
else  
    WinClose, Calculator
```

28.15 WinGet

返回符合指定条件的窗口的 `uID`, 进程 ID, 进程名称, 或控件列表。它也可以返回一个列表, 包含所有符合指定条件的窗口。

`WinGet, OutputVar [, Cmd, WinTitle, WinText, ExcludeTitle, ExcludeText]`

参数

<code>OutputVar</code>	输出变量, 存储由 <code>Cmd</code> 指定的结果。
<code>Cmd</code>	查看下面的说明。
<code>WinTitle</code>	目标窗口的标题或标题中的部分文字 (匹配模式由 SetTitleMatchMode(See 28.8) 决定)。如果省略下面 3 个参数, 默认目标是 上一次匹配的窗口(See 30.2) 。要用窗口的 <code>class</code> 名进行匹配, 使用 <code>ahk_class</code> 精确 <code>class</code> 名 (<code>Window Spy</code> 中可以显示 <code>class</code> 名)。要用窗口的 进程标识符(PID)(See 24.4) 进行匹配, 使用 <code>ahk_pid %PID 变量%</code> 。要用窗口的 唯一 ID(See 28.15) 进行匹配, 使用 <code>ahk_id %ID 变量%</code> 。要减小检测范围, 使用 多重条件(See 30.2) , 例如: <code>My File.txt ahk_pid %ID 变量%</code>
<code>WinText</code>	如果使用这个参数, 则它应该是目标窗口中某个文本元素的子字符串 (在 <code>Window Spy</code> 中会显示出窗口中的文本元素)。隐藏文本只有在 DetectHiddenText(See 28.4) 设置为 <code>ON</code> 的时候才能检测到。
<code>ExcludeTitle</code>	标题中包含该参数指定的文字的窗口将被除外。
<code>ExcludeText</code>	文本元素中包含该参数指定的文字的窗口将被除外。

`Cmd` 指定了返回值的内容, 如果留空, 默认是 `ID`。它可以是下面单词的其中之一:

ID: 返回窗口的 `uID`(`HWND`/句柄)。如果没有匹配窗口, `OutputVar` 为空。函数 [WinExist\(\)\(See 28.7\)](#) 和 [WinActive\(\)\(See 28.6\)](#) 也可以用来获取窗口的 `uID`, 例如, `WinExist("A")` 能快速获取当前激活窗口的 `uID`。要获取一个控件的 `HWND` (供 [Post/SendMessage\(See 28.1.12\)](#) 或 [DIICall\(See 18.3\)](#) 使用), 使用 [ControlGet Hwnd\(See 28.1.4\)](#) 或 [MouseGetPos\(See 23.5\)](#)。

IDLast: 和上面的功能一样, 不过当匹配的窗口超过一个的时候, 它返回的是 最底端/最后使用 的窗口的 `ID`。如果只有一个匹配窗口, 它和 `ID` 的效果一样。类似的概念也适用于 [WinActivateBottom\(See 28.13\)](#)。

PID: 返回窗口的 [进程 ID \(PID\) \(See 24.4\)](#)。

ProcessName: 返回窗口所属进程的进程名称 (例如 `notepad.exe`)。如果没有匹配窗口, `OutputVar` 为空。

Count: 返回符合 `WinTitle` , `WinText` , `ExcludeTitle` 以及 `ExcludeText` 所指定的条件的窗口数量 (如果没有匹配窗口则返回 `0`)。要统计当前总窗口数, 省略所有和 标题/文本 有关的参数。默认不统计隐藏窗口, 除非打开了 [DetectHiddenWindows\(See 28.5\)](#)。

List: 返回所有符合 `WinTitle` , `WinText` , `ExcludeTitle` 以及 `ExcludeText` 所指定的条件的窗口的 `uID` (要返回当前所有窗口的 `uID`, 省略所有和 标题/文本 有关的参数)。每个 `ID` 由一个 [数组元素\(See 30.5\)](#) 存储, 数组元素名从 `OutputVar` 开始, 而 `OutputVar` 本身则存储了数组的长度 (如果没有匹配窗口则为 `0`)。例如, `OutputVar` 使用 `MyArray` , 有 2 个匹配窗口, 则变量 `MyArray1` 存储了第一个窗口的 `ID` , 变量 `MyArray2` 存储了第二个窗口的 `ID` , 而变量 `MyArray` 存储了数组长度 `2` 。窗口 `ID` 的返回顺序是从最顶端的窗口到最底端的窗口 (取决于它们在桌面上的堆放方式)。默认不返回隐藏窗

口的 ID，除非打开了 [DetectHiddenWindows\(See 28.5\)](#)。在一个 [函数\(See 10.\)](#) 中，要创建一个全局数组，在使用这个命令之前 [声明\(See 10.\)](#) MyArray 为全局变量（在使用了 [assume-global\(See 10.\)](#) 的函数中则是相反的情况）。

MinMax: 返回窗口的 最小化/最大化 状态。如果没有匹配窗口，*OutputVar* 为空，否则，返回下列数值之一：

- 1: 窗口处于最小化状态（[WinRestore\(See 28.28\)](#) 可以还原它）。
- 1: 窗口处于最大化状态（[WinRestore\(See 28.28\)](#) 可以还原它）。
- 0: 窗口既不处于最大化也不处于最小化状态。

ControlList: 返回一个列表，包含窗口中所有控件的名称。如果没有匹配窗口或窗口中没有控件，*OutputVar* 为空。否则，每个控件的名称由它的 class 名和序号 (ClassNN) 组成，同 Window Spy 中显示的一样。

除了最后一个控件的名称之外，每个名称后都跟着一个换行符 (`n)。要单独遍历所有的控件名称，使用 [parsing loop\(See 17.14\)](#)，可以参考最下面示例区中的例子。

控件名称的存储顺序取决于它们的 Z 轴顺序，如果窗口支持 Tab 导航，这个顺序一般和 Tab 顺序一样。

要返回鼠标当前所指控件的信息，使用 [MouseGetPos\(See 23.5\)](#)。

ControlListHwnd [v1.0.43.06+]: 功能同上，不同的是它返回的是控件的 窗口句柄 (HWND) (See 28.1.4)。

Transparent: 返回窗口的透明度（用 [WinSet\(See 28.29\)](#) 设置窗口透明）。*OutputVar* 在以下情况时为空：1、系统比 Windows XP 老；2、没有匹配窗口；3、窗口没有设置透明；4、其它情况（由系统行为引起），例如窗口透明后被最小化，还原，以及/或者调整大小。否则，*OutputVar* 存储一个 0 到 255 之间的值，0 表示完全透明，255 表示不透明。例如：

```
MouseGetPos,,, MouseWin
WinGet, Transparent, Transparent, ahk_id %MouseWin% ; 获取鼠标当前所指窗口的透明度。
```

TransColor: 返回窗口的透明色（用 [WinSet\(See 28.29\)](#) 设置透明色）。*OutputVar* 在以下情况时为空：1、系统比 Windows XP 老；2、没有匹配窗口；3、窗口没有设置透明色；4、其它情况（由系统行为引起），例如窗口透明后被最小化，还原，以及/或者调整大小。否则，*OutputVar* 存储一个 6 位的十六进制数，表示 RGB 颜色，比如 0x00CC99。例如：

```
MouseGetPos,,, MouseWin
WinGet, TransColor, TransColor, ahk_id %MouseWin% ; 获取鼠标当前所指窗口的透明色。
```

Style 或 ExStyle: 返回一个 8 位的十六进制数，表示窗口的样式或扩展样式。如果没有匹配窗口，*OutputVar* 为空。下面这个例子检测一个窗口是否具有 WS_DISABLED 样式：

```
WinGet, Style, Style, My Window Title
if (Style & 0x8000000) ; 0x8000000 表示 WS_DISABLED 。
... 检测到窗口为不可用状态，进行适当的操作。
```

下面的例子检测一个窗口是否具有 WS_EX_TOPMOST 样式（置顶）：

```
WinGet, ExStyle, ExStyle, My Window Title
if (ExStyle & 0x8) ; 0x8 表示 WS_EX_TOPMOST.
... 检测到窗口处于置顶状态，进行适当的操作。
```

更多样式，查看 [样式列表\(See 30.18\)](#)。

注意

窗口的 ID 只在窗口的生命周期中有效。也就是说，一个程序重启之后，它所有的窗口都将使用新的 ID。

这个命令获取的 ID 仅是一个数字（不包括前缀“ahk_id”），并且以十六进制存储，与 [SetFormat\(See 21.10\)](#) 的设置无关。

要获取鼠标当前所指窗口的 ID，使用 [MouseGetPos\(See 23.5\)](#)。

虽然目前获取的 ID 是 32 位的无符号整数，但是将来的版本中可能会变成 64 位的。因此，对这些值进行数学操作，例如相加，是不安全的。因为这些操作要求操作数是有正负号的整数而不是无符号的整数。

窗口的标题和窗口中的文字是大小写敏感的。要检测隐藏窗口，必须打开 [DetectHiddenWindows\(See 28.5\)](#)。

相关命令

[WinGetClass\(See 28.18\)](#), [Process\(See 24.4\)](#), [WinGetTitle\(See 28.21\)](#), [MouseGetPos\(See 23.5\)](#),
[ControlGet\(See 28.1.4\)](#), [ControlFocus\(See 28.1.3\)](#), [GroupAdd\(See 28.2.2\)](#)

示例

```
; 示例 #1: 最大化当前激活窗口并且显示它的 uID。
WinGet, active_id, ID, A
WinMaximize, ahk_id %active_id%
MsgBox, The active window's ID is "%active_id%".

; 示例 #2: 访问当前所有窗口并且分别显示它们的信息。
WinGet, id, list,,, Program Manager
Loop, %id%
{
    this_id := id%A_Index%
    WinActivate, ahk_id %this_id%
```

```

WinGetClass, this_class, ahk_id %this_id%
WinGetTitle, this_title, ahk_id %this_id%
MsgBox, 4,, Visiting All Windows`n%a_index%
of %id%`nahk_id %this_id%`nahk_class %this_class%`n%this_title%`n`nContinue?
IfMsgBox, NO, break
}

; 示例 #3: 从控件列表中分离出每个单独的控件名称。
WinGet, ActiveControlList, ControlList, A
Loop, Parse, ActiveControlList, `n
{
    MsgBox, 4,, Control #%a_index% is "%A_LoopField%". Continue?
    IfMsgBox, No
        break
}

; 示例 #4: 实时显示当前激活窗口的控件列表。
#Persistent
SetTimer, WatchActiveWindow, 200
return
WatchActiveWindow:
WinGet, ControlList, ControlList, A
ToolTip, %ControlList%
return

```

28.16 WinGetActiveStats

将 [WinGetActiveTitle](#)(See 28.17) 和 [WinGetPos](#)(See 28.19) 合并为一条命令。

`WinGetActiveStats, Title, Width, Height, X, Y`

参数

Title	存储当前激活窗口的标题的变量。
Width/Height	存储当前激活窗口的宽或高的变量。
X, Y	存储当前激活窗口左上角的横坐标和纵坐标的变量。

注意

如果没有找到匹配的窗口，输出变量为空。

这个命令等同于顺序执行下面两个命令：

[WinGetTitle\(See 28.21\)](#), Title, A

[WinGetPos\(See 28.19\)](#), X, Y, Width, Height, A

如果当前激活的窗口是一个隐藏窗口并且 [DetectHiddenWindows\(See 28.5\)](#) 是关闭的（默认），则所有的命令，除了 [WinShow\(See 28.31\)](#)，都不能检测到该窗口。如果因为这个原因或者其它的原因不能检测到当前激活的窗口，则该命令所有的输出变量都为空。

相关命令

[WinGetPos\(See 28.19\)](#), [WinGetActiveTitle\(See 28.17\)](#), [WinGetTitle\(See 28.21\)](#),

[WinGetClass\(See 28.18\)](#), [WinGetText\(See 28.20\)](#), [ControlGetText\(See 28.1.7\)](#)

示例

```
WinGetActiveStats, Title, Width, Height, X, Y
```

```
MsgBox, The active window "%Title%" is %Width% wide` , %Height% tall` , and positioned  
at %X%` ,%Y%.
```

28.17 WinGetActiveTitle

获取当前激活窗口的标题。

`WinGetActiveTitle, OutputVar`

参数

OutputVar	输出变量，存储当前激活窗口的标题。
-----------	-------------------

注意

这个命令等同于：[WinGetTitle\(See 28.21\)](#), OutputVar, A

相关命令

[WinGetPos\(See 28.19\)](#), [WinGetActiveStats\(See 28.16\)](#), [WinGetTitle\(See 28.21\)](#),

[WinGetClass\(See 28.18\)](#), [WinGetText\(See 28.20\)](#), [ControlGetText\(See 28.1.7\)](#)

示例

```
WinGetActiveTitle, Title
MsgBox, The active window is "%Title%".
```

28.18 WinGetClass

返回匹配指定条件的窗口的 `class` 名。

`WinGetClass, OutputVar [, WinTitle, WinText, ExcludeTitle, ExcludeText]`

参数

<code>OutputVar</code>	输出变量，存储 <code>class</code> 名。
<code>WinTitle</code>	目标窗口的标题或标题中的部分文字（匹配模式由 SetTitleMatchMode(See 28.8) 决定）。如果省略所有的参数，默认目标是 上一次匹配窗口(See 30.2) 。如果这个参数使用字母 A，同时省略其它 3 个参数，则以当前激活的窗口作为目标。要用窗口的 进程标识符 (PID) (See 24.4) 进行匹配，使用 <code>ahk_pid %PID%</code> 变量%。要用 窗口组(See 28.2.2) ，使用 <code>ahk_group</code> 组名。要用窗口的 唯一 ID(See 28.15) 进行匹配，使用 <code>ahk_id %ID%</code> 变量%。要减小检测范围，使用 多重条件(See 30.2) ，例如：My File.txt ahk_class Notepad
<code>WinText</code>	如果使用这个参数，则它应该是目标窗口中某个文本元素的子字符串（在 Window Spy 中会显示出窗口中的文本元素）。隐藏文本只有在 DetectHiddenText(See 28.4) 设置为 ON 的时候才能检测到。
<code>ExcludeTitle</code>	标题中包含该参数指定的文字的窗口将被除外。
<code>ExcludeText</code>	文本元素中包含该参数指定的文字的窗口将被除外。

注意

只返回 `class` 名（前缀 “ahk_class” 不会存入 `OutputVar` 中）。

窗口的标题和窗口中的文字是大小写敏感的。要检测隐藏窗口，必须打开 [DetectHiddenWindows\(See 28.5\)](#)。

相关命令

[WinGet\(See 28.15\)](#), [WinGetTitle\(See 28.21\)](#)

示例

```
WinGetClass, class, A
MsgBox, The active window's class is "%class%".
```

28.19 WinGetPos

返回匹配指定条件的窗口的位置和大小。

`WinGetPos [, X, Y, Width, Height, WinTitle, WinText, ExcludeTitle, ExcludeText]`

参数

X, Y	存储目标窗口左上角的横坐标和纵坐标的变量。如果省略，则不能获取到相应的数据。
Width/Height	存储目标窗口的宽或高的变量。如果省略，则不能获取到相应的数据。
WinTitle	目标窗口的标题或标题中的部分文字（匹配模式由 SetTitleMatchMode(See 28.8) 决定）。如果省略后面的 3 个参数，默认目标是 上一次匹配窗口(See 30.2) 。如果这个参数使用字母 A，同时省略后面的 3 个参数，则以当前激活的窗口作为目标。要用窗口的 class 名进行匹配，使用 <code>ahk_class</code> 精确 class 名（Window Spy 中可以显示 class 名）。要用窗口的 进程标识符(PID)(See 24.4) 进行匹配，使用 <code>ahk_pid %PID 变量%</code> 。要用 窗口组(See 28.2.2) ，使用 <code>ahk_group</code> 组名。要用窗口的 唯一 ID(See 28.15) 进行匹配，使用 <code>ahk_id %ID 变量%</code> 。要减小检测范围，使用 多重条件(See 30.2) ，例如： <code>My File.txt ahk_class Notepad</code>
WinText	如果使用这个参数，则它应该是目标窗口中某个文本元素的子字符串（在 Window Spy 中会显示出窗口中的文本元素）。隐藏文本只有在 DetectHiddenText(See 28.4) 设置为 ON 的时候才能检测到。
ExcludeTitle	标题中包含该参数指定的文字的窗口将被除外。
ExcludeText	文本元素中包含该参数指定的文字的窗口将被除外。

注意

如果没有匹配窗口，输出变量为空。

如果参数 `WinTitle` 使用“Program Manager”，该命令会返回桌面的大小，一般和当前屏幕分辨率相同。

最小化的窗口同样有位置和大小，具体取决于操作系统以及系统设置。

要获取当前鼠标所指窗口和控件的名称，使用 [MouseGetPos\(See 23.5\)](#)。

窗口的标题和窗口中的文字是大小写敏感的。要检测隐藏窗口，必须打开 [DetectHiddenWindows\(See 28.5\)](#)。

相关命令

[WinMove\(See 28.27\)](#), [ControlGetPos\(See 28.1.6\)](#), [WinGetActiveStats\(See 28.16\)](#),
[WinGetActiveTitle\(See 28.17\)](#), [WinGetTitle\(See 28.21\)](#), [WinGetText\(See 28.20\)](#),
[ControlGetText\(See 28.1.7\)](#)

示例

```
WinGetPos, X, Y, Width, Height, Calculator
```

```
MsgBox, Calculator is at %X%,%Y%
```

WinGetPos, X, Y, , , A ; “A”表示使用当前激活窗口为目标窗口

```
 MsgBox, The active window is at %X%,%Y%
```

```
IfWinExist, Untitled - Notepad
```

```
{
```

```
 WinGetPos, Xpos, Ypos ; 使用上面找到的窗口
```

```
 MsgBox, Notepad is at %Xpos%,%Ypos%
```

```
}
```

28.20 WinGetText

返回匹配指定条件的窗口中的文本。

```
WinGetText, OutputVar [, WinTitle, WinText, ExcludeTitle, ExcludeText]
```

参数

OutputVar	输出变量，存储获取的文本。
WinTitle	目标窗口的标题或标题中的部分文字（匹配模式由 SetTitleMatchMode(See 28.8) 决定）。如果省略其它 3 个参数，默认目标是 上一次匹配窗口(See 30.2) 。如果这个参数使用字母 A，同时省略其它 3 个参数，则以当前激活的窗口作为目标。要用窗口的 class 名进行匹配，使用 <code>ahk_class</code> 精确 class 名（Window Spy 中可以显示 class 名）。要用窗口的 进程标识符 (PID) (See 24.4) 进行匹配，使用 <code>ahk_pid %PID 变量%</code> 。要用 窗口组(See 28.2.2) ，使用 <code>ahk_group</code> 组名。要用窗口的 唯一 ID(See 28.15) 进行匹配，使用 <code>ahk_id %ID 变量%</code> 。要减小检测范围，使用 多重条件(See 30.2) ，例如：My File.txt ahk_class Notepad
WinText	如果使用这个参数，则它应该是目标窗口中某个文本元素的子字符串（在 Window Spy 中会显示出窗口中的文本元素）。隐藏文本只有在 DetectHiddenText(See 28.4) 设置为 ON 的时候才能检测到。
ExcludeTitle	标题中包含该参数指定的文字的窗口将被除外。
ExcludeText	文本元素中包含该参数指定的文字的窗口将被除外。

ErrorLevel

如果出现错误，[ErrorLevel/错误级别\(See 30.8\)](#) 设置为 1，否则为 0。

注意

一般情况下，返回的文本和 Window Spy 中显示的一样。但是，如果 [DetectHiddenText\(See 28.4\)](#) 为关闭状态，则不能获取隐藏文本。

每个获取的文本元素之后都跟着一个回车和一个换行符（CR+LF），在脚本中用 `r`n 表示。要将返回值分解为单行文本或从中提取子字符串，使用 [StringGetPos\(See 27.14\)](#) 和 [StringMid\(See 27.19\)](#) 这样的命令。也可以使用 [parsing loop\(See 17.14\)](#) 来单独检查每行文本或每个单词。

如果获取的文本不完整，尝试一下在 `WinGetText` 之前使用 [VarSetCapacity\(OutputVar, 55\)\(See 18.17\)](#)（将 55 替换为大于窗口文本的大小）。有时候这是很有必要的，因为有些应用程序的窗口不会正确响应 `WM_GETTEXTLENGTH` 消息，导致 `AutoHotkey` 将输出变量设置过小，无法存储所有的窗口文本。

能够获取的文本数量取决于变量的最大容量（可以使用 [#MaxMem\(See 29.15\)](#) 设置）。因此，当一个窗口包含大量文本（例如打开了一个大文档的记事本）的时候，这个命令可能会消耗大量的内存。要避免这种情况，使用 [ControlGetText\(See 28.1.7\)](#) 来获取窗口的部分文本。任何时候，一个变量可以通过指向空值来释放它所占用的内存，例如 `OutputVar =`

`Windows 95/98/ME` 中可能会限制窗口中的每个文本元素大小不超过 64 KB。

要获取窗口中的控件列表，使用：[WinGet\(See 28.15\)](#), `OutputVar`, `ControlList`, `WinTitle`

窗口的标题和窗口中的文字是大小写敏感的。要检测隐藏窗口，必须打开 [DetectHiddenWindows\(See 28.5\)](#)。

相关命令

[ControlGetText\(See 28.1.7\)](#), [WinGetActiveStats\(See 28.16\)](#), [WinGetActiveTitle\(See 28.17\)](#),
[WinGetTitle\(See 28.21\)](#), [WinGetPos\(See 28.19\)](#), [#MaxMem\(See 29.15\)](#)

示例

```
Run, Calc.exe
WinWait, Calculator
WinGetText, text ; 使用上面找到的窗口
MsgBox, The text is: `n%text%
```

28.21 WinGetTitle

返回匹配指定条件的窗口的标题。

`WinGetTitle, OutputVar [, WinTitle, WinText, ExcludeTitle, ExcludeText]`

参数

OutputVar	输出变量，存储标题。
WinTitle	目标窗口的标题或标题中的部分文字（匹配模式由 SetTitleMatchMode(See 28.8) 决定）。如果省略其它 3 个参数，默认目标是 上一次匹配窗口(See 30.2) 。如果这个参数使用字母 A，同时省略其它 3 个参数，则以当前激活的窗口作为目标。要用窗口的 class 名进行匹配，使用 <i>ahk_class</i> 精确 class 名（Window Spy 中可以显示 class 名）。要用窗口的 进程标识符 (PID) (See 24.4) 进行匹配，使用 <i>ahk_pid %PID%</i> 变量%。要用 窗口组(See 28.2.2) ，使用 <i>ahk_group</i> 组名。要用窗口的 唯一 ID(See 28.15) 进行匹配，使用 <i>ahk_id %ID%</i> 变量%。要减小检测范围，使用 多重条件(See 30.2) ，例如：My File.txt ahk_class Notepad
WinText	如果使用这个参数，则它应该是目标窗口中某个文本元素的子字符串（在 Window Spy 中会显示出窗口中的文本元素）。隐藏文本只有在 DetectHiddenText(See 28.4) 设置为 ON 的时候才能检测到。
ExcludeTitle	标题中包含该参数指定的文字的窗口将被除外。
ExcludeText	文本元素中包含该参数指定的文字的窗口将被除外。

注意

要获取当前鼠标所指的窗口标题，使用 [MouseGetPos\(See 23.5\)](#)。

窗口的标题和窗口中的文字是大小写敏感的。要检测隐藏窗口，必须打开 [DetectHiddenWindows\(See 28.5\)](#)。

相关命令

[WinGetActiveStats\(See 28.16\)](#), [WinGetActiveTitle\(See 28.17\)](#), [WinGetClass\(See 28.18\)](#),
[WinGet\(See 28.15\)](#), [WinGetText\(See 28.20\)](#), [ControlGetText\(See 28.1.7\)](#), [WinGetPos\(See 28.19\)](#)

示例

```
WingetTitle, Title, A
MsgBox, The active window is "%Title%".
```

28.22 WinHide

隐藏匹配指定条件的窗口。

`WinHide [, WinTitle, WinText, ExcludeTitle, ExcludeText]`

参数

WinTitle	目标窗口的标题或标题中的部分文字（匹配模式由 SetTitleMatchMode(See 28.8) 决定）。如果省略所有的参数，默认目标是 上一次匹配窗口(See 30.2) 。如果这个参数使用
----------	--

	字母 A , 同时省略其它 3 个参数, 则以当前激活的窗口作为目标。要用窗口的 class 名进行匹配, 使用 <code>ahk_class</code> 精确 class 名 (Window Spy 中可以显示 class 名)。要用窗口的 进程标识符 (PID) (See 24.4) 进行匹配, 使用 <code>ahk_pid %PID%</code> 变量%。要用 窗口组 (See 28.2.2), 使用 <code>ahk_group</code> 组名 (此时 <code>WinText</code> , <code>ExcludeTitle</code> , 以及 <code>ExcludeText</code> 三个变量要省略)。要用窗口的 唯一 ID (See 28.15) 进行匹配, 使用 <code>ahk_id %ID%</code> 变量%。要减小检测范围, 使用 多重条件 (See 30.2) , 例如: <code>My File.txt ahk_class Notepad</code>
WinText	如果使用这个参数, 则它应该是目标窗口中某个文本元素的子字符串 (在 Window Spy 中会显示出窗口中的文本元素)。隐藏文本只有在 DetectHiddenText (See 28.4) 设置为 ON 的时候才能检测到。
ExcludeTitle	标题中包含该参数指定的文字的窗口将被除外。
ExcludeText	文本元素中包含该参数指定的文字的窗口将被除外。

注意

使用 [WinShow](#)(See 28.31) 显示一个隐藏的窗口 (不受 [DetectHiddenWindows](#)(See 28.5) 设置的影响)。

这个命令只对匹配窗口中处于最前端的进行操作, 除了在 `WinTitle` 参数中使用 [ahk_group](#) 组名(See 28.2.2) 的时候, 这时候会影响窗口组中的所有窗口。

可以使用下面的脚本隐藏/显示任务栏:

```
WinHide ahk_class Shell_TrayWnd
WinShow ahk_class Shell_TrayWnd
```

相关命令

[WinShow](#)(See 28.31), [SetTitleMatchMode](#)(See 28.8), [DetectHiddenWindows](#)(See 28.5), [Last Found Window](#)(See 30.2), [WinSet](#)(See 28.29)

示例

```
Run, notepad.exe
WinWait, Untitled - Notepad
Sleep, 500
WinHide ; 使用上面匹配到的窗口
Sleep, 1000
WinShow
```

28.23 WinKill

强制关闭匹配指定条件的窗口。

WinKill [, WinTitle, WinText, SecondsToWait, ExcludeTitle, ExcludeText]

参数

WinTitle	目标窗口的标题或标题中的部分文字（匹配模式由 SetTitleMatchMode(See 28.8) 决定）。如果省略其它 3 个窗口参数，默认目标是 上一次匹配窗口(See 30.2) 。如果这个参数使用字母 A，同时省略其它 3 个窗口参数，则以当前激活的窗口作为目标。要用窗口的 class 名进行匹配，使用 <code>ahk_class</code> 精确 class 名（Window Spy 中可以显示 class 名）。要用窗口的 进程标识符 (PID) (See 24.4) 进行匹配，使用 <code>ahk_pid %PID%</code> 变量%。要用 窗口组(See 28.2.2) ，使用 <code>ahk_group</code> 组名（此时 <code>WinText</code> ， <code>ExcludeTitle</code> ，以及 <code>ExcludeText</code> 三个参数要省略）。要用窗口的 唯一 ID(See 28.15) 进行匹配，使用 <code>ahk_id %ID%</code> 变量%。要减小检测范围，使用 多重条件(See 30.2) ，例如：My File.txt ahk_class Notepad
WinText	如果使用这个参数，则它应该是目标窗口中某个文本元素的子字符串（在 Window Spy 中会显示出窗口中的文本元素）。隐藏文本只有在 DetectHiddenText(See 28.4) 设置为 ON 的时候才能检测到。
SecondsToWait	如果省略或留空，命令完全不会等待。如果使用 0，会等待 500ms。否则，它会等待指定的时间（可以包含小数点或者是一个 表达式(See 9.) ）来让窗口彻底关闭。如果窗口在指定的时间内并未关闭，脚本会继续执行，但是该命令 不会 设置 ErrorLevel/错误等级。所以当需要检测一个窗口是否已被关闭的时候，使用 IfWinExist(See 28.7) 或 WinWaitClose(See 28.34) 。
ExcludeTitle	标题中包含该参数指定的文字的窗口将被除外。
ExcludeText	文本元素中包含该参数指定的文字的窗口将被除外。

注意

这个命令首先会短暂的尝试正常关闭窗口，如果失败了，就通过结束窗口进程的方式来强制关闭它。

如果匹配的窗口中有一个处于激活状态，那么它会被关闭，脚本不会再关闭其它的匹配窗口。一般情况下，如果匹配的窗口不止一个，处于最前端（最近使用）的窗口将被关闭。

这个命令只对匹配窗口中处于最前端的进行操作，除了在 `WinTitle` 参数中使用 `ahk_group` 组名([See 28.2.2](#)) 的时候，这时候会影响窗口组中的所有窗口。

窗口的标题和窗口中的文字是大小写敏感的。要检测隐藏窗口，必须打开 [DetectHiddenWindows\(See 28.5\)](#)。

相关命令

[WinClose\(See 28.14\)](#), [WinWaitClose\(See 28.34\)](#), [Process\(See 24.4\)](#), [WinActivate\(See 28.12\)](#), [SetTitleMatchMode\(See 28.8\)](#), [DetectHiddenWindows\(See 28.5\)](#), [Last Found Window\(See 30.2\)](#), [IfWinExist\(See 28.7\)](#), [IfWinActive\(See 28.6\)](#), [WinWaitActive\(See 28.33\)](#), [WinWait\(See 28.32\)](#), [GroupActivate\(See 28.2.1\)](#)

示例

```
IfWinExist, Untitled - Notepad
    WinKill ; 使用上面找到的窗口
else
    WinKill, Calculator
```

28.24 WinMaximize

最大化匹配指定条件的窗口。

`WinMaximize [, WinTitle, WinText, ExcludeTitle, ExcludeText]`

参数

WinTitle	目标窗口的标题或标题中的部分文字（匹配模式由 SetTitleMatchMode(See 28.8) 决定）。如果省略其它 3 个参数，默认目标是 上一次匹配窗口(See 30.2) 。如果这个参数使用字母 A，同时省略其它 3 个参数，则以当前激活的窗口作为目标。要用窗口的 class 名进行匹配，使用 <code>ahk_class</code> 精确 class 名（Window Spy 中可以显示 class 名）。要用窗口的 进程标识符 (PID) (See 24.4) 进行匹配，使用 <code>ahk_pid %PID%</code> 。要用 窗口组(See 28.2.2) ，使用 <code>ahk_group</code> 组名（此时 <code>WinText</code> , <code>ExcludeTitle</code> ，以及 <code>ExcludeText</code> 三个参数要省略）。要用窗口的 唯一 ID(See 28.15) 进行匹配，使用 <code>ahk_id %ID%</code> 。要减小检测范围，使用 多重条件(See 30.2) ，例如：My File.txt <code>ahk_class Notepad</code>
WinText	如果使用这个参数，则它应该是目标窗口中某个文本元素的子字符串（在 Window Spy 中会显示出窗口中的文本元素）。隐藏文本只有在 DetectHiddenText(See 28.4) 设置为 ON 的时候才能检测到。
ExcludeTitle	标题中包含该参数指定的文字的窗口将被除外。
ExcludeText	文本元素中包含该参数指定的文字的窗口将被除外。

注意

使用 [WinRestore\(See 28.28\)](#) 还原窗口，使用 [WinMinimize\(See 28.25\)](#) 将窗口最小化。

如果某些窗口不能正确响应 `WinMaximize`，尝试使用下面的脚本：

```
PostMessage, (See 28.1.12)0x112, 0xF030,,, WinTitle, WinText ; 0x112 =
WM_SYSCOMMAND, 0xF030 = SC_MAXIMIZE
```

这个命令只对匹配窗口中处于最前端的进行操作，除了在 `WinTitle` 参数中使用 [ahk_group 组名\(See 28.2.2\)](#) 的时候，这时候会影响窗口组中的所有窗口。

窗口的标题和窗口中的文字是大小写敏感的。要检测隐藏窗口，必须打开 [DetectHiddenWindows\(See 28.5\)](#)。

相关命令

[WinRestore\(See 28.28\)](#), [WinMinimize\(See 28.25\)](#)

示例

```
Run, notepad.exe
```

```
WinWait, Untitled - Notepad
```

```
WinMaximize ; 使用上面找到的窗口
```

```
^Up::WinMaximize, A ; 设置一个快捷键来最大化当前窗口
```

28.25 WinMinimize

最小化匹配指定条件的窗口到任务栏。

`WinMinimize [, WinTitle, WinText, ExcludeTitle, ExcludeText]`

参数

WinTitle	目标窗口的标题或标题中的部分文字（匹配模式由 SetTitleMatchMode(See 28.8) 决定）。如果省略其它 3 个参数，默认目标是 上一次匹配窗口(See 30.2) 。如果这个参数使用字母 A，同时省略其它 3 个参数，则以当前激活的窗口作为目标。要用窗口的 class 名进行匹配，使用 <code>ahk_class</code> 精确 class 名（Window Spy 中可以显示 class 名）。要用窗口的 进程标识符 (PID) (See 24.4) 进行匹配，使用 <code>ahk_pid %PID%</code> 。要用 窗口组(See 28.2.2) ，使用 <code>ahk_group</code> 组名（此时 <code>WinText</code> ， <code>ExcludeTitle</code> ，以及 <code>ExcludeText</code> 三个参数要省略）。要用窗口的 唯一 ID(See 28.15) 进行匹配，使用 <code>ahk_id %ID%</code> 。要减小检测范围，使用 多重条件(See 30.2) ，例如：My File.txt ahk_class Notepad
WinText	如果使用这个参数，则它应该是目标窗口中某个文本元素的子字符串（在 Window Spy 中会显示出窗口中的文本元素）。隐藏文本只有在 DetectHiddenText(See 28.4) 设置为 ON 的时候才能检测到。
ExcludeTitle	标题中包含该参数指定的文字的窗口将被除外。
ExcludeText	文本元素中包含该参数指定的文字的窗口将被除外。

注意

使用 [WinRestore\(See 28.28\)](#) 或者 [WinMaximize\(See 28.24\)](#) 还原窗口。

如果某些窗口不能正确响应 `WinMinimize`，尝试使用下面的脚本：

```
PostMessage, (See 28.1.12)0x112, 0xF020,,, WinTitle, WinText ; 0x112 =
WM_SYSCOMMAND , 0xF020 = SC_MINIMIZE
```

这个命令只对匹配窗口中处于最前端的进行操作，除了在 `WinTitle` 参数中使用 `ahk_group` 组名(See 28.2.2) 的时候，这时候会影响窗口组中的所有窗口。

窗口的标题和窗口中的文字是大小写敏感的。要检测隐藏窗口，必须打开 `DetectHiddenWindows`(See 28.5)。

相关命令

[WinRestore](#)(See 28.28), [WinMaximize](#)(See 28.24), [WinMinimizeAll](#)(See 28.26)

示例

```
Run, notepad.exe
WinWait, Untitled - Notepad
WinMinimize ; 使用上面找到的窗口

^Down::WinMinimize, A ;设置一个快捷键来最小化当前窗口
```

28.26 WinMinimizeAll/WinMinimizeAllUndo

最小化或还原所有窗口

`WinMinimizeAll`
`WinMinimizeAllUndo`

在大多数系统中，这个命令等同于 `Explore` 的 `Win-M` 和 `Win-D` 快捷键。

相关命令

[WinMinimize](#)(See 28.25), [GroupAdd](#)(See 28.2.2)

示例

```
WinMinimizeAll
WinMinimizeAllUndo
```

28.27 WinMove

对匹配指定条件的窗口进行移动或更改大小的操作。

`WinMove, X, Y`
`WinMove, WinTitle, WinText, X, Y [, Width, Height, ExcludeTitle, ExcludeText]`

参数

X, Y	目标窗口新位置的 X 轴 和 Y 轴 坐标（单位是像素），以窗口左上角为基准点，可以是 表达式(See 9.)。屏幕左上角坐标为 0, 0 。 如果只指定了这 2 个参数，其它参数留空，默认目标是 上一次匹配窗口(See 30.2)。
WinTitle	目标窗口的标题或标题中的部分文字（匹配模式由 SetTitleMatchMode(See 28.8) 决定）。如果省略其它 3 个窗口参数，默认目标是 上一次匹配窗口(See 30.2)。如果这个参数使用字母 A，同时省略其它 3 个窗口参数，则以当前激活的窗口作为目标。要用窗口的 class 名进行匹配，使用 <i>ahk_class</i> 精确 class 名（Window Spy 中可以显示 class 名）。要用窗口的 进程标识符 (PID) (See 24.4) 进行匹配，使用 <i>ahk_pid %PID 变量%</i> 。要用 窗口组(See 28.2.2)，使用 <i>ahk_group</i> 组名。要用窗口的 唯一 ID(See 28.15) 进行匹配，使用 <i>ahk_id %ID 变量%</i> 。要减小检测范围，使用 多重条件(See 30.2)，例如：My File.txt ahk_class Notepad
WinText	如果使用这个参数，则它应该是目标窗口中某个文本元素的子字符串（在 Window Spy 中会显示出窗口中的文本元素）。隐藏文本只有在 DetectHiddenText(See 28.4) 设置为 ON 的时候才能检测到。
Width, Height	目标窗口新大小的宽度和高度（单位是像素），可以是 表达式(See 9.)。如果其中一个或两个一起留空或省略或使用单词 DEFAULT，则默认等于原始宽度/高度。
ExcludeTitle	标题中包含该参数指定的文字的窗口将被除外。
ExcludeText	文本元素中包含该参数指定的文字的窗口将被除外。

注意

如果 *Width* 和 *Height* 参数太小（或负数），大多数带标题栏的窗口最小不会小于 112 x 27 像素（有些类型的窗口可能大小有出入）。如果 *Width* 和 *Height* 参数太大，大多数窗口最大不会大于（桌面分辨率 + 12）像素。

坐标轴参数可以使用负数，这是为了支持多显示器系统，以及可以让窗口完全移出桌面。

虽然 WinMove 不能移动最小化的窗口，但是在 DetectHiddenWindows(See 28.5) 打开的情况下它可以移动隐藏窗口。

窗口移动的速度受 SetWinDelay(See 28.9) 影响。

窗口的标题和窗口中的文字是大小写敏感的。要检测隐藏窗口，必须打开 DetectHiddenWindows(See 28.5)。

相关命令

[ControlMove](#)(See 28.1.8), [WinGetPos](#)(See 28.19), [WinHide](#)(See 28.22), [WinMinimize](#)(See 28.25), [WinMaximize](#)(See 28.24), [WinSet](#)(See 28.29)

示例

```

Run, calc.exe
WinWait, Calculator
WinMove, 0, 0 ; 将 WinWait 找到的窗口移动到屏幕左上角。

SplashTextOn, 400, 300, Clipboard, The clipboard contains: `n%clipboard%
WinMove, Clipboard, , 0, 0 ; 将窗口移动到左上角
Msgbox, Press OK to dismiss the SplashText
SplashTextOff

; 下面的 函数(See 10.) 将指定的窗口居中。
CenterWindow(WinTitle)
{
    WinGetPos,,, Width, Height, %WinTitle%
    WinMove, %WinTitle%,, (A_ScreenWidth/2)-(Width/2),
(A_ScreenHeight/2)-(Height/2)
}

```

28.28 WinRestore

还原匹配指定条件的处于最大化或最小化状态中的窗口。

`WinRestore [, WinTitle, WinText, ExcludeTitle, ExcludeText]`

参数

WinTitle	目标窗口的标题或标题中的部分文字（匹配模式由 SetTitleMatchMode(See 28.8) 决定）。如果省略其它 3 个参数，默认目标是 上一次匹配窗口(See 30.2) 。如果这个参数使用字母 A，同时省略其它 3 个参数，则以当前激活的窗口作为目标。要用窗口的 class 名进行匹配，使用 <code>ahk_class</code> 精确 class 名（Window Spy 中可以显示 class 名）。要用窗口的 进程标识符 (PID) (See 24.4) 进行匹配，使用 <code>ahk_pid %PID 变量%</code> 。要用 窗口组(See 28.2.2) ，使用 <code>ahk_group</code> 组名（此时 <code>WinText</code> ， <code>ExcludeTitle</code> ，以及 <code>ExcludeText</code> 三个参数要省略）。要用窗口的 唯一 ID(See 28.15) 进行匹配，使用 <code>ahk_id %ID 变量%</code> 。要减小检测范围，使用 多重条件(See 30.2) ，例如：My File.txt ahk_class Notepad
WinText	如果使用这个参数，则它应该是目标窗口中某个文本元素的子字符串（在 Window Spy 中

	会显示出窗口中的文本元素)。隐藏文本只有在 DetectHiddenText(See 28.4) 设置为 ON 的时候才能检测到。
ExcludeTitle	标题中包含该参数指定的文字的窗口将被除外。
ExcludeText	文本元素中包含该参数指定的文字的窗口将被除外。

注意

如果某些窗口不能正确响应 `WinRestore`，尝试使用下面的脚本：

```
PostMessage, (See 28.1.12)0x112, 0xF120,,, WinTitle, WinText ; 0x112 =
WM_SYSCOMMAND, 0xF120 = SC_RESTORE
```

这个命令只对匹配窗口中处于最前端的进行操作，除了在 `WinTitle` 参数中使用 [ahk_group 组名\(See 28.2.2\)](#) 的时候，这时候会影响窗口组中的所有窗口。

窗口的标题和窗口中的文字是大小写敏感的。要检测隐藏窗口，必须打开 [DetectHiddenWindows\(See 28.5\)](#)。

相关命令

[WinMinimize\(See 28.25\)](#), [WinMaximize\(See 28.24\)](#)

示例

```
WinRestore, Untitled - Notepad
```

28.29 WinSet

对匹配指定条件的窗口进行一系列的设置，例如设置“总在最前”，以及透明度等。

`WinSet, Attribute, Value [, WinTitle, WinText, ExcludeTitle, ExcludeText]`

参数

Attribute, Value	查看下面的列表。
WinTitle	目标窗口的标题或标题中的部分文字（匹配模式由 SetTitleMatchMode(See 28.8) 决定）。如果省略后面 3 个参数，默认目标是 上一次匹配的窗口(See 30.2) 。如果这个参数使用字母 A，同时省略后面 3 个参数，则以当前激活的窗口作为目标。要用窗口的 class 名进行匹配，使用 <code>ahk_class</code> 精确 class 名（Window Spy 中可以显示 class 名）。要用窗口的 进程标识符 (PID) (See 24.4) 进行匹配，使用 <code>ahk_pid %PID 变量%</code> 。要用 窗口组(See 28.2.2) ，使用 <code>ahk_group</code> 组名。要用窗口的 唯一 ID(See 28.15) 进行匹配，使用 <code>ahk_id %ID 变量%</code> 。要减小检测范围，使用 多重条件(See 30.2) ，例如：My File.txt ahk_class Notepad
WinText	如果使用这个参数，则它应该是目标窗口中某个文本元素的子字符串（在 Window Spy 中

	会显示出窗口中的文本元素)。隐藏文本只有在 DetectHiddenText(See 28.4) 设置为 ON 的时候才能检测到。
ExcludeTitle	标题中包含该参数指定的文字的窗口将被除外。
ExcludeText	文本元素中包含该参数指定的文字的窗口将被除外。

Attribute, Value

AlwaysOnTop, [On|Off|Toggle]: 让一个窗口处于置顶状态。使用 ON 开启设置，使用 OFF 关闭设置，使用 TOGGLE 切换设置。如果省略，默认是 TOGGLE。也可以使用单词 Topmost 替换 AlwaysOnTop。

Bottom: 将一个窗口放到最底端，也就是所有窗口的下面。这个效果类似使用 Alt-Escape 快捷键。例如：WinSet, Bottom,, WinTitle

Top: 将一个窗口放到最前端而不进行 [激活\(See 28.12\)](#)。但是，大多数情况下系统都会自动激活它。另外，当系统禁止切换窗口焦点的时候该命令可能没有效果（这取决于当前窗口类型以及用户当前的操作）。要应付这种情况，可以让窗口暂时 [置顶](#)，再取消置顶。

Disable 或 Enable: 停用或启用一个窗口。当一个窗口处于停用状态的时候，用户不能移动它或利用它的控件进行交互。另外，停用的窗口也不会出现在 alt-tab 列表中。

Redraw: 通过通知系统某个窗口区域需要重绘，来刷新一个窗口的界面和内容。如果对某个窗口不起作用，尝试 [WinMove\(See 28.27\)](#)。如果还不起作用，尝试：

[WinHide\(See 28.22\)](#), WinTitle

[WinShow\(See 28.31\)](#), WinTitle

Style,N 或 ExStyle,N: 分别设置窗口的样式或扩展样式。如果 N 中的第一个字符是加号或减号，则窗口会增加或移除 N 中指定的样式（可同时指定多个样式，下同）；如果第一个字符是符号“`”，窗口将对 N 中指定的样式进行切换操作；如果第一个字符是数字，窗口原来的所有样式将被覆盖，彻底变成 N 中指定的样式。

如果样式设置失败，[ErrorLevel/错误等级\(See 30.8\)](#) 设置为 1，成功为 0。当没有匹配窗口或窗口不能应用指定的样式的时候（一般发生在 Windows 9x 中）会设置失败。

更改了一个窗口的样式之后，可能需要对窗口进行重绘，使用命令 [WinSet Redraw](#)（具体见下面）。最后，一些常用的样式代码可以参看 [样式列表\(See 30.18\)](#)。一些例子：

WinSet, Style, -0xC00000, A ; 移除当前激活窗口的标题栏 (WS_CAPTION)。

WinSet, ExStyle, ^0x80, WinTitle ; 切换窗口的 WS_EX_TOOLWINDOW 属性，可以将窗口添加到/移除于 alt-tab 列表中。

WinSet, Region [, Options, WinTitle, ...]

更改一个窗口的形状为指定的方形，椭圆形，或多边形。如果 Options 参数留空，窗口将恢复到原来的形状。否则，可以指定下面的一个或多个参数，用空格分隔：

Wn: 方形或椭圆形的宽度。例如: w200

Hn: 方形或椭圆形的高度。例如: h300

X-Y: X 轴 和 Y 轴 坐标。例如, 200-0 中的 200 表示 X 轴 坐标, 0 表示 Y 轴 坐标。

E: 将窗口形状设置为椭圆形, 而不是方形。这个参数只有在同时指定了 W 和 H 参数的时候才有效。

R[w-h]: 设置方形窗口的圆角。例如, R30-30 表示圆角大小为 30x30 。如果省略 w-h , 默认使用 30-30 。R 参数只有在同时指定了 W 和 H 参数的时候才有效。

方形或椭圆形: 如果同时指定了 W 和 H 参数, 窗口的形状将变成方形, 方形区域左上角的位置坐标为 X-Y 参数指定的坐标 (以窗口左上角为原点)。但是, 如果同时使用了 E 参数的话, 窗口的形状将变成椭圆形。

例如: WinSet, Region, 50-0 W200 H250 E, WinTitle

多边形: 如果同时指定了 W 和 H 参数, 并且在 X-Y 参数中指定了多组坐标, 窗口的形状将变成多边形 (每组坐标以窗口左上角为原点)。例如, 如果指定了 3 组坐标, 大多数情况下窗口的形状会变成三角形, 具体的形状取决于 3 组坐标的顺序。此外, 在 Options 参数中也可以加上单词 Wind , 这将会使用一种弯曲的方式来决定多边形的形状。

如果形状设置失败, ErrorLevel/错误等级(See 30.8) 设置为 1 , 成功为 0 。失败的原因有: 1) 没有匹配窗口; 2) Options 中有一个或多个参数无效; 3) 指定了超过 2000 组坐标; 4) 指定的形状无效或不适用与目标窗口。

更多示例请查看本页最下方的示例代码。

WinSet, Transparent, N, WinTitle

设置窗口为半透明状态。指定 N 为一个 0 到 255 的数字来表示透明度: 0 表示完全透明, 255 表示完全不透明。使用单词 OFF 彻底关闭窗口的透明效果。和设置透明度为 255 不同, 使用 OFF 会提升系统性能并降低系统资源的占用。

已知的关于透明度和 TransColor/透明色 的限制:

- 在 Windows 9x 和 NT4 上无效。
- 对于没有标题栏并且没有 总在最前 属性的窗口无效。对于 图形界面窗口(See 19.3), 可以在设置了窗口透明之后再移除标题栏。另外, 下面的属性设置也可以让窗口设置透明度: Gui -Caption +ToolWindow(See 19.3).
- 设置透明度为 OFF 之前最好先设置透明度为 255 , 这样可以避免窗口的重绘问题, 例如背景变黑。如果窗口仍然不能正确的重绘, 参看下面有关 Redraw/重绘 的说明。
- 要更改窗口当前的 TransColor/透明色 , 最好先关闭窗口透明。

小贴士: 要设置任务栏的透明度, 使用 WinSet, Transparent, 150, ahk_class Shell_TrayWnd 。类似的, 要设置开始菜单的透明度, 参照这个例子:

```
DetectHiddenWindows, on
```

```
WinSet, Transparent, 150, ahk_class BaseBar ; 要让开始菜单的子菜单也透明, 继续加入下面的代码。
```

要让当前选择的菜单或系统所有的菜单透明, 运行下面这个脚本。注意虽然这个脚本不能让它自己的菜单透明, 但是可以让其它脚本的菜单透明:

```
#Persistent

SetTimer, WatchForMenu, 5

return ; 结束自动运行区域。

WatchForMenu:

DetectHiddenWindows, on ; 以立即检测到菜单。

IfWinExist, ahk_class #32768

    WinSet, Transparent, 150 ; 使用上一行找到的窗口。

return
```

WinSet, TransColor, Color [N], WinTitle

让目标窗口中所有符合指定颜色的区域完全透明（在 Windows 9x 和 NT4 中无效）。如果用户点击窗口中的透明部分，点击事件会“穿透”该窗口传递给下面的窗口。*Color* 参数可以是一个颜色名称或一个 RGB 值（关于颜色详细请参看 [颜色图表\(See 19.12\)](#)，或者使用 [PixelGetColor\(See 22.15\)](#) 的 RGB 模式）。要让目标窗口中符合指定颜色的区域半透明，在颜色参数之后添加一个表示透明度的数字（0 - 255）。例如：*WinSet, TransColor, EEA99 150, WinTitle*

TransColor/透明色 常被用来制作屏幕菜单（OSD）或窗口特效。有一个关于屏幕菜单的例子在 [Gui 命令帮助页的最下面\(See 19.3\)](#)。

单词 **OFF** 用来完全关闭窗口的透明设置。下面两个脚本的功能是一样的：

```
WinSet, Transparent, Off, WinTitle
WinSet, TransColor, Off, WinTitle
```

已知的限制：参看 [上面的列表](#)。

注意

除了上面明确说明的地方外，[ErrorLevel/错误等级\(See 30.8\)](#) 不会被该命令修改。

虽然 Windows 2000/XP 及之后的系统都支持设置透明，但是只有 XP 之后的系统才能获取到窗口的透明度设置（通过 [WinGet\(See 28.15\)](#) 命令）。

可以用下面的脚本取消一个脚本的 [SplashText\(See 19.14\)](#) 窗口的置顶状态：

```
WinSet, AlwaysOnTop, Off, My Splash Window Title
```

窗口的标题和窗口中的文字是大小写敏感的。要检测隐藏窗口，必须打开 [DetectHiddenWindows\(See 28.5\)](#)。

相关命令

[WinGet](#)(See 28.15), [WinHide](#)(See 28.22), [WinSetTitle](#)(See 28.30), [WinMove](#)(See 28.27),
[WinActivate](#)(See 28.12), [Control](#)(See 28.1.1)

示例

```
WinSet, Transparent, 200, Untitled - Notepad ; 让窗口稍微有点透明效果  
WinSet, TransColor, White, Untitled - Notepad ; 让窗口中白色的部分透明  
WinSet, AlwaysOnTop, toggle, Calculator ; 切换计算器的“总在最前”状态  
  
; 长一点的例子：  
; 这里有一些快捷键用来演示“Transparent/透明度”和“TransColor/透明色”的效果  
; 注意：如果你按下快捷键的时候鼠标正指在 TransColor/透明色  
; 设置的窗口透明部分上，则快捷键影响的就是透明窗口下方的窗口。  
; 同样，快捷键 Win + G 只有在 Windows XP 上才有效果，  
; 因为 Windows 2000 不能获取窗口的透明设置  
  
#t:: ; 快捷键 Win+T 让窗口中符合鼠标所指颜色的部分透明  
MouseGetPos, MouseX, MouseY, MouseWin  
PixelGetColor, MouseRGB, %MouseX%, %MouseY%, RGB  
; 设置新的透明效果之前需要关闭窗口当前的透明效果  
WinSet, TransColor, Off, ahk_id %MouseWin%  
WinSet, TransColor, %MouseRGB% 220, ahk_id %MouseWin%  
return  
  
#o:: ; 快捷键 Win+O 关闭鼠标所指窗口的透明效果  
MouseGetPos,,, MouseWin  
WinSet, TransColor, Off, ahk_id %MouseWin%  
return  
  
#g:: ; 快捷键 Win+G 显示鼠标所指窗口当前的透明效果设置  
MouseGetPos,,, MouseWin
```

```

WinGet, Transparent, Transparent, ahk_id %MouseWin%
WinGet, TransColor, TransColor, ahk_id %MouseWin%
ToolTip Translucency: `t%Transparent%`nTransColor: `t%TransColor%
return

; "WinSet Region"的例子

WinSet, Region, 50-0 W200 H250, WinTitle ; 让窗口只显示指定的方形部分

WinSet, Region, 50-0 W200 H250 R40-40, WinTitle ; 和上面的一样，不过有 40 x 40 的圆角。

WinSet, Region, 50-0 W200 H250 E, WinTitle ; 椭圆形窗口

WinSet, Region, 50-0 250-0 150-250, WinTitle ; 倒三角形窗口

WinSet, Region,, WinTitle ; 将窗口恢复原状

; 更加复杂的 Region 例子，在窗口中创建了一个方形的“洞”。

; 这里指定了 2 个方形，一个内部的一个外部的。每个方形使用 5 组坐标

WinSet, Region, 0-0 300-0 300-300 0-300 0-0    100-100 200-100 200-200 100-200
100-100, WinTitle

```

28.30 WinSetTitle

改变匹配指定条件的窗口的标题。

`WinSetTitle, NewTitle`
`WinSetTitle, WinTitle, WinText, NewTitle [, ExcludeTitle, ExcludeText]`

参数

NewTitle	目标窗口的新标题。如果只指定这一个参数，其它参数省略，默认目标是 上一次匹配窗口 (See 30.2)。
WinTitle	目标窗口的标题或标题中的部分文字（匹配模式由 SetTitleMatchMode (See 28.8) 决定）。如果省略其它 3 个参数，默认目标是 上一次匹配窗口 (See 30.2)。如果这个参数使用字母 A，同时省略其它 3 个参数，则以当前激活的窗口作为目标。要用窗口的 class 名进行匹配，使用 <code>ahk_class</code> 精确 class 名 (Window Spy 中可以显示 class 名)。要用窗口的 进程标识符 (PID) (See 24.4) 进行匹配，使用 <code>ahk_pid %PID%</code> 变量%。要用 窗口组 (See 28.2.2)，使用 <code>ahk_group</code> 组名。要用窗口的 唯一 ID (See 28.15) 进行匹配，使用 <code>ahk_id %ID%</code> 变量%。要减小检测范围，使用 多重条件 (See 30.2)，例如：My File.txt ahk_class Notepad

WinText	如果使用这个参数，则它应该是目标窗口中某个文本元素的子字符串（在 Window Spy 中会显示出窗口中的文本元素）。隐藏文本只有在 DetectHiddenText(See 28.4) 设置为 ON 的时候才能检测到。
ExcludeTitle	标题中包含该参数指定的文字的窗口将被除外。
ExcludeText	文本元素中包含该参数指定的文字的窗口将被除外。

注意

如果窗口所在的程序本身经常修改窗口的标题，那么该命令对窗口标题的修改就只能是暂时的。

窗口的标题和窗口中的文字是大小写敏感的。要检测隐藏窗口，必须打开 [DetectHiddenWindows\(See 28.5\)](#)。

相关命令

[WinMove\(See 28.27\)](#), [WinGetActiveStats\(See 28.16\)](#), [WinGetActiveTitle\(See 28.17\)](#),
[WinGetText\(See 28.20\)](#), [ControlGetText\(See 28.1.7\)](#), [WinGetPos\(See 28.19\)](#), [WinSet\(See 28.29\)](#)

示例

```
WinSetTitle, Untitled - Notepad, , This is a new title

; 另一个修改记事本标题的例子:

Run, notepad.exe

WinWaitActive, Untitled - Notepad

WinSetTitle, This is a new title ; 使用上面 WinWaitActive 找到的窗口
```

28.31 WinShow

显示匹配指定条件的窗口。

`WinShow [, WinTitle, WinText, ExcludeTitle, ExcludeText]`

参数

WinTitle	目标窗口的标题或标题中的部分文字（匹配模式由 SetTitleMatchMode(See 28.8) 决定）。如果省略所有的参数，默认目标是 上一次匹配窗口(See 30.2) 。要用窗口的 class 名进行匹配，使用 <i>ahk_class</i> 精确 class 名（Window Spy 中可以显示 class 名）。要用窗口的 进程标识符(PID) (See 24.4) 进行匹配，使用 <i>ahk_pid %PID%</i> 变量%。要用 窗口组(See 28.2.2) ，使用 <i>ahk_group</i> 组名（此时 <i>WinText</i> ， <i>ExcludeTitle</i> ，以及 <i>ExcludeText</i> 三个变量要省略）。要用窗口的 唯一 ID(See 28.15) 进行匹配，使用
----------	---

	<i>ahk_id %ID% 变量%</i> 。要减小检测范围，使用 多重条件(See 30.2) ，例如： <i>My File.txt ahk_class Notepad</i>
WinText	如果使用这个参数，则它应该是目标窗口中某个文本元素的子字符串（在 Window Spy 中会显示出窗口中的文本元素）。隐藏文本只有在 DetectHiddenText(See 28.4) 设置为 ON 的时候才能检测到。
ExcludeTitle	标题中包含该参数指定的文字的窗口将被除外。
ExcludeText	文本元素中包含该参数指定的文字的窗口将被除外。

注意

默认情况下，`WinShow` 是唯一一个在任何时候都能检测到隐藏窗口的命令。其它的命令只有在开启 [DetectHiddenWindows\(See 28.5\)](#) 的时候才能检测到隐藏窗口。

这个命令只对匹配窗口中处于最前端的进行操作，除了在 `WinTitle` 参数中使用 [ahk_group 组名\(See 28.2.2\)](#) 的时候，这时候会影响窗口组中的所有窗口。

相关命令

[WinHide\(See 28.22\)](#), [SetTitleMatchMode\(See 28.8\)](#), [DetectHiddenWindows\(See 28.5\)](#), [Last Found Window\(See 30.2\)](#)

示例

```
Run, notepad.exe
WinWait, Untitled - Notepad
Sleep, 500
WinHide ; 因为它省略了所有参数，它使用上面找到的窗口。.
Sleep, 1000
WinShow
```

28.32 WinWait

等待，直到匹配指定条件的窗口出现。

`WinWait [, WinTitle, WinText, Seconds, ExcludeTitle, ExcludeText]`

参数

WinTitle	目标窗口的标题或标题中的部分文字（匹配模式由 SetTitleMatchMode(See 28.8) 决定）。要用窗口的 <code>class</code> 名进行匹配，使用 <code>ahk_class 精确 class 名</code> （Window Spy 中可以显示 <code>class</code> 名）。要用窗口的 <code>进程标识符 (PID)</code> (See 24.4) 进行匹配，使用
----------	--

	<i>ahk_pid %PID 变量%</i> 。要用 窗口组(See 28.2.2) ，使用 <i>ahk_group</i> 组名。要减小检测范围，使用 多重条件(See 30.2) ，例如：My File.txt ahk_class Notepad <i>WinTitle</i> 只有在 <i>WinText</i> ， <i>ExcludeTitle</i> ，以及 <i>ExcludeText</i> 不都为空的时候才能留空。
WinText	如果使用这个参数，则它应该是目标窗口中某个文本元素的子字符串（在 Window Spy 中会显示出窗口中的文本元素）。隐藏文本只有在 DetectHiddenText(See 28.4) 设置为 ON 的时候才能检测到。
Seconds	超时时间。超时后将不再等待并且设置 ErrorLevel/错误等级(See 30.8) 为 1。留空表示无限等待。指定 0 等同于 0.5。这个参数可以一个 表达式(See 9.) 。
ExcludeTitle	标题中包含该参数指定的文字的窗口将被除外。
ExcludeText	文本元素中包含该参数指定的文字的窗口将被除外。

ErrorLevel/错误等级

超时的话 [ErrorLevel/错误等级\(See 30.8\)](#) 为 1，否则为 0。

注意

匹配的窗口出现后，命令会立即设置 [ErrorLevel/错误等级\(See 30.8\)](#) 为 0，更新 [上一次匹配窗口\(See 30.2\)](#)，然后让脚本继续执行，而不必再等待 *Seconds* 参数中指定的时间。

当该命令处于等待状态的时候，仍然可以通过 [快捷键\(See 4.\)](#)，[自定义菜单\(See 22.13\)](#)，或 [定时器\(See 17.21\)](#) 来启动新的 [线程\(See 30.19\)](#)。

如果其它的 [线程\(See 30.19\)](#) 在该命令等待的时候改变了参数中所使用到的变量的值，该命令会无视这些改变 —— 它仍然会使用这些变量在开始等待的时候的值。

窗口的标题和窗口中的文字是大小写敏感的。要检测隐藏窗口，必须打开 [DetectHiddenWindows\(See 28.5\)](#)。

相关命令

[WinWaitActive\(See 28.33\)](#), [WinWaitClose\(See 28.34\)](#), [IfWinExist\(See 28.7\)](#), [IfWinActive\(See 28.6\)](#), [Process\(See 24.4\)](#), [SetTitleMatchMode\(See 28.8\)](#), [DetectHiddenWindows\(See 28.5\)](#)

示例

```
Run, notepad.exe
WinWait, Untitled - Notepad, , 3
if ErrorLevel
{
```

```

    MsgBox, WinWait timed out.

    return

}

else

    WinMinimize ; 最小化上面找到窗口。

```

28.33 WinWaitActive/WinWaitNotActive

等待匹配指定条件的窗口处于激活或未激活状态。

`WinWaitActive [, WinTitle, WinText, Seconds, ExcludeTitle, ExcludeText]`

`WinWaitNotActive [, WinTitle, WinText, Seconds, ExcludeTitle, ExcludeText]`

参数

WinTitle	目标窗口的标题或标题中的部分文字（匹配模式由 SetTitleMatchMode(See 28.8) 决定）。如果省略其它 3 个窗口参数，默认目标是 上一次匹配窗口(See 30.2) 。要用窗口的 <code>class</code> 名进行匹配，使用 <code>ahk_class</code> 精确 <code>class</code> 名（Window Spy 中可以显示 <code>class</code> 名）。要用窗口的 进程标识符(PID)(See 24.4) 进行匹配，使用 <code>ahk_pid %PID 变量%</code> 。要用 窗口组(See 28.2.2) ，使用 <code>ahk_group</code> 组名。要用窗口的 唯一 ID(See 28.15) 进行匹配，使用 <code>ahk_id %ID 变量%</code> 。要减小检测范围，使用 多重条件(See 30.2) ，例如： <code>My File.txt ahk_class Notepad</code>
WinText	如果使用这个参数，则它应该是目标窗口中某个文本元素的子字符串（在 Window Spy 中会显示出窗口中的文本元素）。隐藏文本只有在 DetectHiddenText(See 28.4) 设置为 ON 的时候才能检测到。
Seconds	超时时间。超时后将不再等待并且设置 ErrorLevel/错误等级(See 30.8) 为 1。留空表示无限等待。指定 0 等同于 0.5。这个参数可以一个 表达式(See 9.) 。
ExcludeTitle	标题中包含该参数指定的文字的窗口将被除外。
ExcludeText	文本元素中包含该参数指定的文字的窗口将被除外。

ErrorLevel

超时以后 [ErrorLevel/错误等级\(See 30.8\)](#) 为 1，否则为 0。

注意

如果一个匹配窗口的状态符合命令要求，命令会立即设置 [ErrorLevel/错误等级\(See 30.8\)](#) 为 0，然后让脚本继续执行，而不必再等待 `Seconds` 参数中指定的时间。

如果命令开始执行的时候匹配窗口处于激活状态，`WinWaitActive` 和 `WinWaitNotActive` 都会更新 [上一次匹配的窗口\(See 30.2\)](#)。另外，匹配窗口在超时时间以内激活，`WinWaitActive` 也会更新上一次匹配窗口。

当该命令处于等待状态的时候，仍然可以通过 [快捷键\(See 4.\)](#)，[自定义菜单\(See 22.13\)](#)，或 [定时器\(See 17.21\)](#) 来启动新的 [线程\(See 30.19\)](#)。

如果其它的 [线程\(See 30.19\)](#) 在该命令等待的时候改变了参数中所使用到的变量的值，该命令会无视这些改变 —— 它仍然会使用这些变量在开始等待的时候的值。

窗口的标题和窗口中的文字是大小写敏感的。要检测隐藏窗口，必须打开 [DetectHiddenWindows\(See 28.5\)](#)。

相关命令

[WinWait\(See 28.32\)](#), [WinWaitClose\(See 28.34\)](#), [IfWinExist\(See 28.7\)](#), [IfWinActive\(See 28.6\)](#), [SetTitleMatchMode\(See 28.8\)](#), [DetectHiddenWindows\(See 28.5\)](#)

示例

```
Run, notepad.exe

WinWaitActive, Untitled - Notepad, , 2

if ErrorLevel

{

    MsgBox, WinWait timed out.

    return

}

else

    WinMinimize ; 最小化 WinWaitActive 找到的窗口。
```

28.34 WinWaitClose

等待匹配指定条件的窗口被关闭。

`WinWaitClose [, WinTitle, WinText, Seconds, ExcludeTitle, ExcludeText]`

参数

WinTitle	目标窗口的标题或标题中的部分文字（匹配模式由 SetTitleMatchMode(See 28.8) 决定）。如果省略其它 3 个窗口参数，默认目标是 上一次匹配窗口(See 30.2) 。要用窗口的 <code>class</code> 名进行匹配，使用 <code>ahk_class</code> 精确 <code>class</code> 名（Window Spy 中可以显示 <code>class</code> 名）。要用窗口的 进程标识符(PID)(See 24.4) 进行匹配，使用 <code>ahk_pid %PID 变量%</code> 。要用 窗口组(See 28.2.2) ，使用 <code>ahk_group</code> 组名。要用窗口的 唯一 ID(See 28.15) 进行匹配，使用 <code>ahk_id %ID 变量%</code> 。要减小检测范围，使用 多重条件(See 30.2) ，例如：My File.txt ahk_class Notepad
----------	---

WinText	如果使用这个参数，则它应该是目标窗口中某个文本元素的子字符串（在 Window Spy 中会显示出窗口中的文本元素）。隐藏文本只有在 DetectHiddenText(See 28.4) 设置为 ON 的时候才能检测到。
Seconds	超时时间。超时后将不再等待并且设置 ErrorLevel/错误等级(See 30.8) 为 1 。留空表示无限等待。指定 0 等同于 0.5 。这个参数可以一个 表达式(See 9.) 。
ExcludeTitle	标题中包含该参数指定的文字的窗口将被除外。
ExcludeText	文本元素中包含该参数指定的文字的窗口将被除外。

ErrorLevel

超时以后 [ErrorLevel/错误等级\(See 30.8\)](#) 为 1 ，否则为 0 。

注意

没有匹配窗口的时候，命令会立即设置 [ErrorLevel/错误等级\(See 30.8\)](#) 为 0 ，然后让脚本继续执行，而不必再等待 `Seconds` 参数中指定的时间。

当该命令处于等待状态的时候，仍然可以通过 [快捷键\(See 4.\)](#)，[自定义菜单\(See 22.13\)](#)，或 [定时器\(See 17.21\)](#) 来启动新的 [线程\(See 30.19\)](#) 。

如果其它的 [线程\(See 30.19\)](#) 在该命令等待的时候改变了参数中所使用到的变量的值，该命令会无视这些改变 —— 它仍然会使用这些变量在开始等待的时候的值。

窗口的标题和窗口中的文字是大小写敏感的。要检测隐藏窗口，必须打开 [DetectHiddenWindows\(See 28.5\)](#) 。

相关命令

[WinClose\(See 28.14\)](#), [WinWait\(See 28.32\)](#), [WinWaitActive\(See 28.33\)](#), [IfWinExist\(See 28.7\)](#),
[IfWinActive\(See 28.6\)](#), [Process\(See 24.4\)](#), [SetTitleMatchMode\(See 28.8\)](#),
[DetectHiddenWindows\(See 28.5\)](#)

示例

```
Run, notepad.exe
WinWait, Untitled - Notepad
WinWaitClose ; 等待 WinWait 找到的窗口被关闭。
MsgBox, Notepad is now closed.
```

29. #指令

29.1 #AllowSameLineComments

只适用于 AutoIt v2 (.aut) 脚本：允许注解出现在命令的同一行。

`#AllowSameLineComments`

在任何 AutoIt v2 (.aut) 脚本的顶部指定此指令将可以使用同行注解，通常为了兼容性的原因会禁用它。如果没有用在脚本的顶部，那么在该指令出现那行之前就不支持同行注解。

示例

```
#AllowSameLineComments
Sleep, 1 ;此注解就是一个同行注解。
```

翻译：yugi 修正：天堂之门 menk33@163.com 2008 年 10 月 20 日

29.2 #ClipboardTimeout

更改首次尝试访问剪贴板失败后需要间隔的时间。

`#ClipboardTimeout Milliseconds`

参数

Milliseconds	以毫秒为单位的间歇时间。指定为 -1 使其保持不停地访问剪贴板。指定为 0 使其只尝试一次。不包含该指令的脚本将在 1000 ms 后超时。
--------------	--

注意

一些应用程序使剪贴板长时间打开，可能是要读写大量的数据。这种情况下，增加此设置的值会使脚本在放弃并显示出错信息前长时间处于等待状态。

该设置应用于所有的剪贴板(See 30.6)操作，最简单的比如下面这个例子：

```
Var = %Clipboard%
Clipboard = 新文本
```

每当脚本处于等待剪贴板的内容成为变量时，新的线程(See 30.19)不能被启动并且定时器(See 17.21)也不会运行。不过，如果用户按了热键(See 4.)，选择了一个自定义菜单项(See 22.13)或者执行了 GUI 操作(See 19.3)比如点击按钮，那么这些事件会先被缓存起来留到之后处理；换句话说，子进程被推迟到剪贴板内容成为变量后才执行。

该指令也会导致首次访问失败后重新尝试读取剪贴板数据（在先前的版本中，只有打开的剪贴板才会重新尝试访问）。

相关命令

[Clipboard](#)(See 30.6), [Thread](#)(See 22.22)

示例

```
#ClipboardTimeout 2000
```

翻译: yugi 修正: 天堂之门 menk33@163.com 2008 年 10 月 26 日

29.3 #CommentFlag

把脚本的注解符号从分号改为其他字符串。

```
#CommentFlag NewString
```

参数

NewString	一个或多个字符可以被用作新的注解标识。最多可指定 15 个字符。
-----------	----------------------------------

注意

默认的注解标识是分号 (;)。

注解标识用来表示紧随其后的文本不应该被脚本执行 (当脚本启动时注解不被加载, 因此它们不影响执行)。

注解标识出现在命令的同一行上不被视为表示注解, 除非它的左边至少有一个空格或 tab。例如:

```
MsgBox, Test1 ; 这是一个注解。
```

```
 MsgBox, Test2; 这不是一个注解并且将被 MsgBox 命令显示。
```

相关命令

[#EscapeChar](#)(See 29.5)

示例

```
#CommentFlag // ; 改为 C++ 注解样式。
```

翻译: 天堂之门 menk33@163.com 2008 年 9 月 3 日

29.4 #ErrorStdOut

让脚本启动时将任何语法错误发送到标准输出而不是显示错误对话框。

```
#ErrorStdOut
```

它允许像 Textpad, SciTE, Crimson 和 EditPlus 这样的优秀编辑器在遇上语法错误时跳转到出错行。由于 `#ErrorStdOut` 指令必须加进每个脚本, 通常设置你的编辑器从而在启动任何 AutoHotkey 脚本时使用[命令行选项](#)(See 8.) /`ErrorStdOut` 会更好(详见更下面的设置说明)。

虽然语法错误被发送到标准输出(stdout)，但不会直接显示在命令提示符中。然而，这些输出可以通过管道或者重定向来获得。例如：

```
"C:\Program Files\AutoHotkey\AutoHotkey.exe" /ErrorStdOut "My Script.ahk" |more
"C:\Program Files\AutoHotkey\AutoHotkey.exe" /ErrorStdOut "My
Script.ahk" >"Syntax-Error Log.txt"
```

你也可以直接传送输出信息到剪切板，通过下载 [cb.zip](#) (4 KB)，然后按照下例操作：

```
"C:\Program Files\AutoHotkey\AutoHotkey.exe" /ErrorStdOut "My Script.ahk"
|cb.exe
```

特定编辑器的设置：

EditPlus:

从菜单栏中选择 Tools > Configure User Tools。

点击按钮：Add Tool > Program

Menu Text：你自己定

Command: C:\Program Files\AutoHotkey\AutoHotkey.exe

Argument: /ErrorStdOut "\$(FilePath)"

Initial directory: \$(FileDir)

Capture output: Yes

TextPad:

从菜单栏中选择 Configure > Preferences。

展开 Tools 选项。

点击 Add 按钮并选择 "Program"。

复制然后粘贴（调整为你的路径）: C:\Windows\System32\cmd.exe -- 然后点击 OK。

在列表框中快速点击三次新添加的项目 (cmd.exe)，重命名为你定的名称(例如 Launch Script)。

点击应用。

在左边的目录树选择新项目并键入如下信息：

Command (应该已经填过): cmd.exe (或者它的完整路径)

Parameters (如果有必要的话调整为你的路径): /c ""C:\Program

Files\AutoHotkey\AutoHotkey.exe" /ErrorStdOut "\$File""

Initial folder: \$FileDir

勾上如下选框：1) Run minimized; 2) Capture output。

点击 OK。新添加的项目现在应该已经在 Tools 菜单中了。

相关命令

[FileAppend](#)(See 16.4) (同样可以发送文本到标准输出)

示例

```
#ErrorStdOut
```

翻译：yugi 修正：天堂之门 menk33@163.com 2008年10月21日

29.5 #EscapeChar

改变脚本的转义符号（例如重音符与反斜线）。

```
#EscapeChar NewChar
```

参数

NewChar	指定单个字符。
---------	---------

注意

转义字符用来表明其后紧跟的字符应该被解释地与它通常的含义不同。

AutoIt v2 (.aut) 脚本默认的转义符号是反斜线 (\)。对所有其它的文件扩展名，包括所有编译的脚本，默认的转义符号是重音符/反引号 (`)。当一个 .aut 脚本 [自动转换\(See 11.\)](#) 为一个 .ahk 脚本时，反斜线转义符号将被全部替换为重音符。

转义顺序（当重音符是转义符号时）

输入这些	来得到这些
,	, (原来的逗号)。 注意： 在命令最后一个参数里出现的逗号不需要被转义，因为程序知道将它们视作原义。同样也适用于 MsgBox(See 19.11) 命令的所有参数，因为它有智能的逗号处理功能。
`%	% (原来的百分号)
``	` (原来的重音符；也就是两个连续的转义字符结果为单个原义的字符)
`;	; (原来的分号)。 注意： 这只有在分号的左边有一个空格或 tab 时才有必要。如果没有，它会被正确地识别而不用转义。
`::	:: (一对原义的冒号)。在 v1.0.40+ 版本之后，它不再需要被转义。
`n	新起一行 (换行/LF)
`r	回车 (CR)
`b	退格
`t	tab (很典型的水平的移位)
`v	垂直的 tab -- 相当于 Ascii 编码的值 11。它也可以在一些应用程序中通过键入 Control+K 来显示。
`a	alert (警告/bell) -- 相当于 Ascii 编码的值 7。它也可以在一些应用程序中通过键入 Control+G 来显示。
`f	formfeed(进纸) -- 相当于 Ascii 编码的值 12。它也可以在一些应用程序中通过键入 Control+L 来显示。
Send	当 Send 命令(See 20.12) 或 热字符串(See 5.) 以它们默认的（非 raw）模式使用时，像 {}^!+# 这样的字符有特殊的含义。因此，在这些情况下要使用它们的原义就得将它们围

	在大括号内。例如: Send {^}{!}{`}
""	在 表达式(See 9.) 内, 一个原义的字符串内的两个连续的引号解析为单个原义的引号。例如: Var := "The color ""red"" was found."

相关命令

同样存在下面这些极其少用的指令; 它们的用法在这些例子里表明:

```
#DerefChar # ; 将引用符号一般默认的 % 改为 # 。  
#Delimiter / ; 将分隔符号一般默认的逗号改为 / 。
```

示例

```
#EscapeChar \ ;将其改变为反斜杠来代替默认的重音符 (`)。
```

翻译: 天堂之门 menk33@163.com 2008年9月7日

29.6 #HotkeyInterval

随同 [#MaxHotkeysPerInterval\(See 20.1.5\)](#) 一起, 指定 热键(See 4.) 激活的速率, 当超过这一速率时, 将会显示一个警告对话框。

```
#HotkeyInterval Milliseconds
```

参数

Milliseconds	间隔的长度, 以毫秒为单位。
--------------	----------------

注意

如果在脚本里没有指定该指令, 它会像设为 2000 毫秒一样运转。

更多细节和注意事项, 请参考: [#MaxHotkeysPerInterval\(See 20.1.5\)](#)。

相关命令

[#MaxHotkeysPerInterval\(See 20.1.5\)](#)

示例

```
#HotkeyInterval 2000 ; 这是默认值(毫秒)。
```

```
#MaxHotkeysPerInterval 200
```

翻译: 坛友 lwjeee 修正: 天堂之门 menk33@163.com 2008年8月3日

29.7 #HotkeyModifierTimeout

Affects the behavior of [hotkey](#)(See 4.) modifiers: CTRL, ALT, WIN, and SHIFT.

#HotkeyModifierTimeout Milliseconds

参数

Milliseconds	The length of the interval in milliseconds. The value can be -1 so that it never times out (modifier keys are always pushed back down after the Send), or 0 so that it always times out (modifier keys are never pushed back down).
--------------	---

注意

This directive **need not** be used when:

- Hotkeys send their keystrokes via the [SendInput](#)(See 20.12) or [SendPlay](#)(See 20.12) methods. This is because those methods postpone the user's physical pressing and releasing of keys until after the Send completes.
- The script has the keyboard hook installed (you can see if your script uses the hook via the "View->Key history" menu item in the main window, or via the [KeyHistory](#)(See 20.9) command). This is because the hook can keep track of which modifier keys (ALT/CTRL/WIN/SHIFT) the user is physically holding down and doesn't need to use the timeout.

To illustrate the effect of this directive, consider this example:

```
^!a::Send, abc
```

When the [Send](#)(See 20.12) command executes, the first thing it does is release the CTRL and ALT keys so that the characters get sent properly. After sending all the keys, the command doesn't know whether it can safely push back down CTRL and ALT (to match whether the user is still holding them down). But if less than the specified number of milliseconds have elapsed, it will assume that the user hasn't had a chance to release the keys yet and it will thus push them back down to match their physical state. Otherwise, the modifier keys will not be pushed back down and the user will have to release and press them again to get them to modify the same or another key.

The timeout should be set to a value less than the amount of time that the user typically holds down a hotkey's modifiers before releasing them. Otherwise, the modifiers may be restored to the down position (get stuck down) even when the user isn't physically holding them down.

You can reduce or eliminate the need for this directive with one of the following:

- Install the keyboard hook by adding the line [#InstallKeybdHook](#)(See 20.2) anywhere in the script (however, the hook is not supported on Win9x).
- Use the [SendInput](#)(See 20.12) or [SendPlay](#)(See 20.12) methods rather than the traditional [SendEvent](#)(See 20.12) method.

- When using the traditional `SendEvent`(See 20.12) method, reduce `SetKeyDelay`(See 20.14) to 0 or -1, which should help because it sends the keystrokes more quickly.

If this is directive is unspecified in a script, it behaves as though set to 50.

相关命令

[GetKeyState](#)(See 20.7)

示例

```
#HotkeyModifierTimeout 100
```

29.8 #Hotstring

改变 [热字符串](#)(See 5.) 的选项或结尾字符。

```
#Hotstring NoMouse
#Hotstring EndChars NewChars
#Hotstring NewOptions
```

参数

NoMouse	防止鼠标点击会像 这里 (See 5.) 描述的那样重设热字符串识别器。作为副作用，这也阻止了热字符串需要的 鼠标钩子 (See 20.3) (然而如果脚本因为其他目的需要它时，它仍会被安装，例如鼠标热键)。 <code>#Hotstring NoMouse</code> 出现在脚本的任何位置都将影响所有的热字符串，不仅仅是那些物理上在它下面的热字符串。
EndChars NewChars	<p>指定 <code>EndChars</code> 单词，紧跟单独空格然后是新的结尾字符。例如：</p> <pre>#Hotstring EndChars -()[]{}';"/\,.?!`n `t</pre> <p>由于新的结尾字符对整个脚本是全局生效的 -- 不管 <code>EndChars</code> 指令出现在哪里 -- 故没有必要多次指定 <code>EndChars</code>。</p> <p>结尾字符的最大数量是 100 个。超出此长度的字符将被忽略。</p> <p>要让 <code>tab</code> 成为一个结尾字符，在列表里包括 <code>'t</code>。要让空格成为一个结尾字符，在列表的其他两个字符中间包含它 (或在列表开头，如果列表仅包含一个其他字符或没有其他字符)。</p>
NewOptions	<p>指定新选项给 热字符串选项(See 5.) 里描述的选项。例如： <code>#Hotstring r s k0 c0</code></p> <p>与上面的 <code>EndChars</code> 不同，<code>#Hotstring</code> 指令按这种方法使用时是位置相关的。换句话说，整个热字符串部分可以有不同的默认选项，像在这个例子中一样：</p> <pre>::btw::by the way</pre> <pre>#Hotstring r c ; 下面所有的热字符串将使用 "send raw" 并且默认为区分大</pre>

	<p>小写。</p> <pre>::al::airline ::CEO::Chief Executive Officer</pre> <p>#Hotstring c0 ; 让这下面的所有热字符串不区分大小写。</p>
--	--

相关命令

[Hotstrings\(See 5.\)](#)

翻译: 天堂之门 menk33@163.com 2008 年 9 月 4 日

29.9 #IfWinActive/Exist

创建上下文相关的 [热键\(See 4.\)](#) 和 [热字符串\(See 5.\)](#)。这样的热键视某类窗口的激活或存在而执行一个不同的动作 (或根本不执行)。

```
#IfWinActive [, WinTitle, WinText]
#IfWinExist [, WinTitle, WinText]
#IfWinNotActive [, WinTitle, WinText]
#IfWinNotExist [, WinTitle, WinText]
```

参数

WinTitle	目标窗口的标题或部分标题 (匹配模式由设置在 自动执行部分(See 8.) 里的 SetTitleMatchMode(See 28.8) 决定(See 8.)。要使用一个窗口类, 指定 <code>ahk_class</code> 确切的 <code>ClassName</code> (通过 <code>Window Spy</code> 显示的)。要使用一个 窗口组(See 28.2.2) , 指定 <code>ahk_group</code> <code>GroupName</code> 。也支持 <code>ahk_pid</code> 和 <code>ahk_id</code> (See 30.2) 关键词, 但对 <code>#IfWin</code> 来说更常见的是通过 GroupAdd(See 28.2.2) 间接地使用它们(或者, 使用 " Hotkey IfWin "(See 20.1.10))。最后, 通过指定 多个条件(See 30.2) 可以缩小搜索范围。例如: <code>My File.txt ahk_class Notepad</code>
WinText	如果有的话, 此参数必须是目标窗口的单独文本对象中的子字符串 (像内含的 <code>Window Spy</code> 工具显示的那样)。如果 DetectHiddenText(See 28.4) 在自动执行部分(脚本的顶部)已被开启, 隐藏的文本对象会被探测到。
ExcludeTitle ExcludeText	虽然这些不被支持, 但它们能间接地通过指定 <code>ahk_group MyGroup</code> 给 <code>WinTitle</code> 来使用 (在那通过 GroupAdd(See 28.2.2) 创建 <code>MyGroup</code> , 它支持 <code>ExcludeTitle/Text</code>)。

基本的操作

`#IfWin` 指令能简单地创建上下文相关的 [hotkeys\(See 4.\)](#) 和 [hotstrings\(See 5.\)](#)。例如:

```
#IfWinActive ahk_class Notepad
#space::MsgBox 你在记事本中按了 Win+空格。
```

#IfWin 指令是位置相关的：它们影响脚本中在它们下面的所有热键和热字符串。它们也是互斥的；也就是说，只有最近的一个才会生效。

要关闭上下文制约，指定任意 #IfWin 指令但省略它所有的参数。例如：

```
#IfWinActive
```

当 #IfWin 关闭时（或没用在脚本中），所有的 [热键\(See 4.\)](#) 和 [热字符串\(See 5.\)](#) 对所有的窗口都有效（除了被 [Suspend\(See 17.23\)](#) 或 [Hotkey 命令\(See 20.1.10\)](#) 禁用）。

当一个鼠标或键盘热键被 #IfWin 条件限制时，它执行它本来的功能；也就是说，它传递到激活的窗口时就仿佛没有那么一个热键。有两个例外： 1) Windows 95/98/Me：按一个 IfWin-禁用的 热键没有任何效果（甚至没有它本来的功能）； 2) 操纵杆热键：虽然 #IfWin 可用，但它不能阻止其他程序探测按钮的按动。

#IfWin 也能用来改变一个普通的按键像 Enter 或 空格 的表现。这在一个特殊的窗口忽略那个按键或执行一些你发觉不想要的动作时很有用。例如：

```
#IfWinActive Reminders ahk_class #32770 ; 在 Outlook 里的 "reminders" 。
```

Enter::Send !o ; 让 "Enter" 键击打开选择的 reminder 而不是让它睡觉。

```
#IfWinActive
```

变体（重复的）热键

一个特定的 [热键\(See 4.\)](#) 或 [热字符串\(See 5.\)](#) 能在脚本中定义多次，假如每个定义都有不同的 #IfWin 条件。这些被称作 *hotkey variants*（热键变体）。例如：

```
#IfWinActive ahk_class Notepad
^!c::MsgBox 你在记事本中按了 Control+Alt+C 。
#IfWinActive ahk_class WordPadClass
^!c::MsgBox 你在写字板中按了 Control+Alt+C 。
#IfWinActive
^!c::MsgBox 你在不是记事本/写字板的窗口中按了 Control+Alt+C 。
```

如果有多个（[译注：即条件不同但热键相同](#)）变体适合被激发，那么只有最接近脚本顶部的那个会被激发。除此之外是全局的变体（没有 #IfWin 条件）：它总是最低的优先级；因此，它只在没有其他合适的变体的时候才会激发（这个例外不适用于 [热字符串\(See 5.\)](#)）。

在创建重复的热键时，[修饰键符号\(See 4.\)](#) 例如 ^!+# 的顺序没有关系。例如，^!c 和 !^c 是一样的。不过按键必须拼写一致。例如，为了这种目的， Esc 和 Escape 是不一样的（然而大小写没关系）。同样，任何带 [通配符前缀 \(*\)](#)（See 4.）的热键完全和一个没有通配符的有区别；例如，*F1 和 F1 每个都将会有自己的变体。

要让同个热键的子程序通过多个变体执行，最简单的方法就是创建一堆相同的热键，每个热键都有不同的 #IfWin 指令在它上面。例如：

```
#IfWinActive ahk_class Notepad
#z:::
```

```
#IfWinActive ahk_class WordPadClass
#z::
MsgBox 你在记事本或写字板中按了 Win+Z 。
return
```

或者，通过 `#IfWinActive ahk_group MyGroup` 使用一个 [窗口组](#)(See 28.2.2)。

要动态地创建热键变种 (当脚本在运行时)，请看 "[Hotkey IfWin](#)"(See 20.1.10)。

一般说明

`#IfWin` 也能在前缀按键被占用时还原它们本来的功能 (在一个热键像 "a & b" 中 [前缀按键](#)(See 4.) 是 "a" 键)。这用在没有给一个特定的前缀指定可用的热键时。

当 `Gosub` 或 `Goto` 被用来跳转到一个热键或热字符串标识时，它跳转到最接近脚本顶部的那个变体。

当一个热键当前被 `#IfWin` 限制时，它的按键或鼠标按钮会用一个 "#" 符号出现在 [KeyHistory](#) 的(See 20.9) "Type" 列里。这能帮助调试一个脚本。

当前不支持例如 `%Var%` 的变量引用。因此，百分号必须通过 `%` 转义(See 29.5) 来允许将来支持它们。同样，原义的逗号必须被转义 (通过 `,`) 来允许附加的变量在将来被添加。如果你需要绕弯解决这种限制，使用 [GroupAdd](#)(See 28.2.2) 和 [ahk_group](#)(See 30.2)。

[Hotkey 命令](#)(See 20.1.10) 指定热键的 `label` 不直接被 `#IfWin` 影响。取而代之的是，在最接近脚本底部使用的 `#IfWin` (如果有的话) 将影响所有由 `Hotkey` 命令创建的热键 (除非使用 "[Hotkey IfWin](#)"(See 20.1.10) 来改变)。

[Alt-tab 热键](#)(See 4.) 不受 `#IfWin` 影响：它们在所有的窗口有效。

[最后找到的窗口](#)(See 30.2) 可被 `#IfWinActive/Exist` 设置(但是不被 `#IfWinNotActive/NotExist` 设置)。例如：

```
#IfWinExist ahk_class Notepad
#n::WinActivate ; 激活被 #IfWin 找到的窗口。
```

如果在 `#IfWin` 的某个变量里需要开头或结尾空格，[转义顺序](#)(See 29.5) `s 和 `t 可以被使用。

由于性能的原因，`#IfWin` 不会持续监视指定窗口的激活或存在。取而代之的是，它只在你输入一个热键或热字串时才去核查匹配窗口。如果匹配的窗口不存在，你的键击或鼠标点击会被允许传递给没变的激活窗口 (除了在 Windows 95/98/Me 上)。

Windows 95/98/Me: 如果热键的首个变体有一个 \$ 前缀，所有的变体将被允许 "发送它们本身"。这为一个热键去执行它原来的功能提供了一种方法而不是完全什么也不做。例如：

```
$^a::Send ^a ; 首个变体必须有一个 $ 前缀来允许它在 Windows 9x 上 "发送自身"。
#IfWinActive ahk_class Notepad
^a::MsgBox 你在记事本激活时按了 Control-A 。
```

窗口标题和文本是区别大小写的。隐藏的窗口不被探测，除非 [DetectHiddenWindows](#)(See 28.5) 在自动执行部分 (脚本顶部) 被开启。

相关命令

[Hotkey command\(See 20.1.10\)](#), [Hotkeys\(See 4.\)](#), [Hotstrings\(See 5.\)](#), [Suspend\(See 17.23\)](#),
[IfWinActive\(See 28.6\)](#), [IfWinExist\(See 28.7\)](#), [SetTitleMatchMode\(See 28.8\)](#),
[DetectHiddenWindows\(See 28.5\)](#)

示例

```
#IfWinActive ahk_class Notepad
^!a::MsgBox 在记事本激活时你按了 Ctrl-Alt-A 。 ; 此热键如果在别的窗口按是无效的 (它将
"pass through"[经过] )。

#c::::MsgBox 在记事本激活时你按了 Win-C 。

::btw::此 "btw" 的替换文本将只在记事本里出现。

#IfWinActive

#c::::MsgBox 在一个不是记事本的窗口里你按了 Win-C 。
```

翻译：天堂之门 menk33@163.com 2008 年 8 月 27 日

29.10 #Include/#IncludeAgain

使脚本表现得好像指定文件的内容就出现在这个位置。

```
#Include FileOrDirectoryName
#includeagain FileOrDirectoryName
```

参数

FileOrDirectoryName	<p>File: 要包含的文件名称，如果未指定绝对路径，它将假设在启动/工作目录(除了 ahk2exe(See 8.)，它会假设文件在脚本自己的目录)。文件名称不能包含变量引用(除了 %A_ScriptDir%(See 9.), %A_AppData%(See 9.) 和 %A_AppDataCommon%(See 9.))、双引号或者通配符。转义顺序(See 29.5)除了分号 (`;) 外都不能被使用，它们也不需要，因为字符比如百分号是原义对待的。注意：SetWorkingDir(See 16.33) 对 <code>#Include</code> 不起作用，因为 <code>#Include</code> 在脚本开始执行之前就已做处理。</p> <p>Directory: 指定一个目录而不是文件从而改变后面出现的所有 <code>#Include</code> 和 FileInstall(See 16.15) 使用的工作目录。目录名称不能包含变量，除了 %A_ScriptDir%(See 9.), %A_AppData%(See 9.) 和 %A_AppDataCommon%(See 9.)。注意：以这种方式改变工作目录不会影响脚本开始运行时的初始工作目录(A_WorkingDir(See 9.))。要改变那个初始工作目录，请在脚本的顶部使用 SetWorkingDir(See 16.33)。</p>
---------------------	---

注意

脚本表现得好像被包含的文件内容就在 `#Include` 指令的位置出现(就好像从包含的文件里复制粘贴过来一样)。因此, 它通常不能将两个孤立的脚本合并成一个可运作的脚本(要实现的话, 请看 www.autohotkey.com/forum/topic18545.html)。

`#Include` 会确保 *FileName* 仅被包含一次, 即使它遇到多次重复的包含。相比之下, `#IncludeAgain` 则允许同个文件的多次包含, 而在所有其他的方面和 `#Include` 一样。

可以在 *FileName* 参数前放置 `*i` 和单个空格, 这样会使程序在加载被包含文件时忽略任何产生的故障。例如: `#Include *i SpecialOptions.ahk`。这个选项应该仅在被包含文件的内容对主脚本的操作是可有可无的情况下才去使用。

通过 [ListLines](#)(See 22.11) 或者菜单 View->Lines 显示在主窗口里的那些行总是按照它们在文件内部的物理顺序编号的。换句话说, 包含一个新的文件将仅仅改变主脚本文件的一行编号, 就是 `#Include` 这行(除了[已编译的脚本](#)(See 8.), 它会在编译时把它们包含的文件合并为一个大的脚本)。

`#Include` 常用来加载在一个外部文件里定义的[函数](#)(See 10.)。和子程序标签不同, [函数](#)(See 10.)可以被包含在脚本的最顶部而不影响[自动执行部分](#)(See 8.)。

和其他 `#` 开头的指令一样, `#Include` 不能被有条件地执行。换句话说, 下例不起作用:

```
if x = 1

#Include SomeFile.ahk ;不管 x 的值是什么, 这行都会生效。

y = 2 ;而此行是属于上面的 IF 语句的, 因为 # 开头的指令不属于 IF 语句。
```

按函数名称在一个[函数库](#)(See 10.)里调用, 可以自动地将文件包含进来(也就是不需要使用 `#Include`)。

相关命令

[Libraries of Functions](#)(See 10.), [Functions](#)(See 10.), [FileInstall](#)(See 16.15)

示例

```
#Include C:\My Documents\Scripts\Utility Subroutines.ahk

#Include %A_ScriptDir% ;为下面的 #Include 和 FileInstall 改变工作目录。

#Include C:\My Scripts ;同上, 不过换成了一个明确命名的目录。
```

翻译: 天堂之门 menk33@163.com 2008 年 11 月 6 日

29.11 #InstallKeybdHook

强制无条件地安装键盘钩子。

`#InstallKeybdHook`

注意

键盘钩子是为了激活不被 [RegisterHotkey](#)(操作系统内置的功能) 支持的 [热字串](#)(See 5.) 以及任何键盘 [热键](#)(See 4.) 这样的目的而监视键击的。它也支持一些其他特性例如 [Input](#)(See 20.11) 命令。

在 Windows 95/98/Me 下键盘热键不被支持，因为这些操作系统需要一个必须存在于 DLL 文件内的不同类型的钩子。

[AutoHotkey](#) 不会无条件地安装键盘和鼠标钩子因为它们总共消耗至少 500 KB 的内存。因此，键盘钩子通常仅在脚本包含下列条件之一时才会安装： 1) [热字串](#)(See 5.); 2) 一个或多个需要键盘钩子的 [热键](#)(See 4.) (大多数不需要); 3) [SetCaps/Scroll/Numlock AlwaysOn/AlwaysOff](#)(See 20.15); 4) [Input](#)(See 20.11) 命令，钩子在首次实际使用时安装。

相比之下，[#InstallKeybdHook](#) 指令将无条件地安装键盘钩子，它在允许 [KeyHistory](#)(See 20.9) 显示最近 20 次键击 (为了调试脚本目的) 或者避免需要使用 [#HotkeyModifierTimeout](#)(See 20.1.2) 时也许会很有用。

你能通过 [KeyHistory](#)(See 20.9) 命令或菜单项确定一个脚本是否在使用钩子。你能通过 [ListHotkeys](#)(See 20.1.11) 命令或菜单项确定哪个热键正在使用钩子。

这个指令也会使一个脚本 [persistent](#)(See 29.21)(持久运行)，意味着应该使用 [ExitApp](#)(See 17.7) 命令来终止脚本。

相关命令

[#InstallMouseHook](#)(See 20.3), [#UseHook](#)(See 20.1.9), [Hotkey](#)(See 20.1.10), [Input](#)(See 20.11), [#Persistent](#)(See 29.21), [KeyHistory](#)(See 20.9), [Hotstrings](#)(See 5.), [GetKeyState](#)(See 20.7), [KeyWait](#)(See 20.10)

示例

```
#InstallKeybdHook
```

翻译：天堂之门 menk33@163.com 2008 年 8 月 17 日

29.12 #InstallMouseHook

强制无条件地安装鼠标钩子。

```
#InstallMouseHook
```

注意

鼠标钩子是为了激活鼠标 [热键](#)(See 4.) 和 [帮助热字串](#)(See 5.) 为目的而监视鼠标点击的。在 Windows 95/98/Me 下它不被支持，因为这些操作系统需要一个必须存在于 DLL 文件内的不同类型的钩子。

[AutoHotkey](#) 不会无条件地安装键盘和鼠标钩子因为它们总共消耗至少 500 KB 的内存 (但如果键盘钩子已安装，那么安装鼠标钩子仅需要大约 50 KB 额外的内存；反之亦然)。因此，键盘钩子通常仅在脚本包含一个或多个鼠标 [热键](#)(See 4.) 时才会安装。它也会为了 [热字串](#)(See 5.) 而安装，但可以通过 [#Hotstring NoMouse](#)(See 20.1.3) 来禁用。

相比之下, `#InstallMouseHook` 指令会无条件地安装鼠标钩子, 它在允许 [KeyHistory\(See 20.9\)](#) 来监视鼠标点击时可能会很有用。

你能通过 [KeyHistory\(See 20.9\)](#) 命令或菜单项确定一个脚本是否在使用钩子。你能通过 [ListHotkeys\(See 20.1.11\)](#) 命令或菜单项确定哪个热键正在使用钩子。

这个指令也会使一个脚本 [persistent\(See 29.21\)](#)(持久运行), 意味着应该使用 [ExitApp\(See 17.7\)](#) 命令来终止脚本。

相关命令

[#InstallKeybdHook\(See 20.2\)](#), [#UseHook\(See 20.1.9\)](#), [Hotkey\(See 20.1.10\)](#), [#Persistent\(See 29.21\)](#), [KeyHistory\(See 20.9\)](#), [GetKeyState\(See 20.7\)](#), [KeyWait\(See 20.10\)](#)

示例

```
#InstallMouseHook
```

翻译: 天堂之门 menk33@163.com 2008 年 8 月 17 日

29.13 #KeyHistory

设置 [KeyHistory\(See 20.9\)](#) 窗口显示的键盘和鼠标事件的最大数量。你可以将其设为 0 来禁用 key history。

```
#KeyHistory MaxEvents
```

参数

MaxEvents	由 KeyHistory(See 20.9) 窗口显示的键盘和鼠标事件的最大数量。(默认 40, 限定 500)。指定 0 完全禁用 key history。
-----------	---

注意

因为此设置在脚本开始运行前已经生效, 所以只要指定一次 (在脚本的任意位置)。

由于每次键击或者鼠标单击由按下以及弹起事件组成, [KeyHistory\(See 20.9\)](#) 仅显示了此处指定的“完成事件”的一半。例如, 如果脚本包含 “`#KeyHistory 50`”, 将显示至多 25 次键击和鼠标鼠标点击。

相关命令

[KeyHistory\(See 20.9\)](#), [#NoTrayIcon\(See 22.1\)](#)(See 20.1.11)

示例

```
#KeyHistory 0      ; 禁用 key history。
```

```
#KeyHistory 100   ; 最多储存 100 个事件。
```

翻译: 天堂之门 menk33@163.com 2008 年 10 月 7 日

29.14 #MaxHotkeysPerInterval

随同 [#\(See 20.1.1\)HotkeyInterval](#)(See 20.1.1) 一起, 指定 热键(See 4.) 激活的速率, 当超过这一速率时, 将会显示一个警告对话框。

#MaxHotkeysPerInterval Value

参数

Value	在 #(See 20.1.1)HotkeyInterval (See 20.1.1) 指定的时间里, 不会触发一个警告对话框而能被按下的热键数量的上限。
-------	--

注意

注意不要将上述数值设定得过于宽松, 因为如果你不经意地引入一个键击的无限循环(通过一个 [Send](#)(See 20.12) 命令意外地触发其它热键)的话, 由于快速而泛滥的键盘事件, 你的电脑可能会变得反应迟钝。

举一个非常简单的例子, 热键 `^c::Send ^c` 将导致一个无限键击循环。要避免这种情况, 为热键定义添加 [\\$ 前缀](#)(See 4.) (例如 `$^c::`), 使得热键不能被 [Send](#) 命令触发。

如果没有在脚本中指定该指令, 它会像设成 70 一样来运转。

相关命令

[#HotkeyInterval](#)(See 20.1.1)

示例

```
#MaxHotkeysPerInterval 200
```

翻译: 坛友 lwjiej 修正: 天堂之门 menk33@163.com 2008 年 8 月 3 日

29.15 #MaxMem

璁剧疆姣朢金鑾檻嘶(See 9.)鑾 娇鑾丨殑鍊€澶 y 啟瀛樺厯鑱般€?/p>

#MaxMem Megabytes

鑾俗囨

Megabytes	姣朢金鑾檻嘶(See 9.)鑾 駁鑽勬啓瀛樺厯鑱般€倣 4095 鑳勬€煎曉瑙囪负 4095 銅倣囪浜?1 鑳勬€煎曉瑙囪负 1 銅?/td>
-----------	---

娉儿剩

濡俗灤鑷氣瀝涓 痘鏈爻塞渝氲繖涓 塞浠わ紅渝富曉璁句负 64 銅?/p>

闡懃埗姣志釜鑾檻嘶鑽勸え灝忋樹涓轰簡闢又 闡樂 鑽勸剗鍼 €橈雇绯葷婢鎵€鍼夊浼鑿 鏅嗚硶鉅
俗弼楂檻尙闡漘綆璇ラ榦錄躁€间筭浼氳奖鐸蔣剗鍼 浩鑑 『追锛屼箇涓斲細鑪瑰強鏞氣溌淪為檻浣跨駁鑽
勸咷瀛檻嘶([WinGetText\(See 28.20\)](#) 鎧?ControlGetText([See 28.1.7](#)) 鎧戒护鑽勸傑鑪甸擴澶柟紅澧
始姑 #MaxMem 鑽勸€煎曉浣垮召鑳借帐号欒汯澶氳浼鑑困溌)?/p>

姝よ 绰 樹鍏ㄥ眬鑑 『追锛屼箇涓斲細鑪瑰強鏞氣溌淪為檻浣跨駁鑽
塞乍搥鑽翠釜鑑氣溌鑑)?/p>

姝よ 绰 炅溌响榦錄朵簡鑾檻嘶鑽勸嚟鐸儿擇鍏告 涓恒€微畠涓慄奖鐸?[VarSetCapacity\(See 18.17\)](#)
鉅)?/p>

鑑稿另鍛戒护

[VarSetCapacity\(See 18.17\)](#), [Variables\(See 9.\)](#), [Sort\(See 27.12\)](#), [WinGetText\(See 28.20\)](#),
[ControlGetText\(See 28.1.7\)](#), #MaxThreads([See 20.1.6](#))

紺轰縲

#MaxMem 256 ;鍏佽 姣志釜鑾檻嘶鍼€澶氳娇鑑?256 MB 鏅嗚硶鉅?/span>

29.16 #MaxThreads

设置同时启动的[线程\(See 30.19\)](#)的最大数量。

#MaxThreads Value

参数

Value	可同时存在的 线程(See 30.19) 的最大总数。指定大于 255 的数值等同与指定 255(在 1.0.48 以前的版本中，这个值的上限是 20)。
-------	---

注意

此设置是全局性的，这就意味着只需要将它指定一次（在脚本任何位置）就能影响整个脚本的表现。

虽然允许但是不推荐将值设为 1，因为这样会在每次脚本显示一个 [MsgBox\(See 19.11\)](#) 或者其他对话框时阻止新的[热键\(See 4.\)](#)运行。也会在每次另一个[线程\(See 30.19\)](#)休眠或等待时阻止[定时器\(See 17.21\)](#)运行。

如果一个线程子程序的首行是 [ExitApp\(See 17.7\)](#), [Pause\(See 17.18\)](#), [Edit\(See 22.9\)](#), [Reload\(See 20.1.13\)](#), [KeyHistory\(See 20.9\)](#), [ListLines\(See 22.11\)](#), [ListVars\(See 22.12\)](#) 或 [ListHotkeys\(See 20.1.11\)](#)，那么至多两种接下来的线程类型即使在 #MaxThread 达到的情况下也可以被创建：[hotkey\(See 4.\)](#), [hotstring\(See 5.\)](#), [OnClipboardChange\(See 30.6\)](#), [GUI event\(See 19.3\)](#)。还有，不论有多少线程存在，[OnExit subroutine\(See 17.17\)](#)(子程序)总是可以启动。

如果此设置低于 [#MaxThreadsPerHotkey\(See 20.1.8\)](#)，那么它将有效地取代那个设置。

如果脚本中没有指定此指令，那么它将表现得好像被设为 10 那样。

相关命令

[#MaxThreadsPerHotkey](#)(See 20.1.8), [Threads](#)(See 30.19), [#MaxHotkeysPerInterval](#)(See 20.1.5), [#HotkeyInterval](#)(See 20.1.1), [ListHotkeys](#)(See 20.1.11), [#MaxMem](#)(See 29.15)

示例

```
#MaxThreads 2
```

翻译: 天堂之门 menk33@163.com 2008 年 11 月 24 日

29.17 #MaxThreadsBuffer

Causes some or all [hotkeys](#)(See 4.) to buffer rather than ignore keypresses when their [#MaxThreadsPerHotkey](#)(See 20.1.8) limit has been reached.

`#MaxThreadsBuffer On|Off`

参数

On Off	<p>On: All hotkey subroutines between here and the next <code>#MaxThreadsBuffer ON</code> directive will buffer rather than ignore presses of their hotkeys whenever their subroutines are at their #MaxThreadsPerHotkey(See 20.1.8) limit.</p> <p>Off: This is the default behavior. A hotkey press will be ignored whenever that hotkey is already running its maximum number of threads (usually 1, but this can be changed with #MaxThreadsPerHotkey(See 20.1.8)).</p>
--------	--

注意

This directive is rarely used because this type of buffering, which is OFF by default, usually does more harm than good. For example, if you accidentally press a hotkey twice, having this setting ON would cause that hotkey's subroutine to automatically run a second time if its first [thread](#)(See 30.19) takes less than 1 second to finish (this type of buffer expires after 1 second, by design). Note that AutoHotkey buffers hotkeys in several other ways (such as "[Thread Interrupt](#)(See 22.22)" and "[Critical](#)"(See 22.7)). It's just that this particular way can be detrimental, thus it is OFF by default.

The main use for this directive is to increase the responsiveness of the keyboard's auto-repeat feature. For example, when you hold down a hotkey whose [#MaxThreadsPerHotkey](#)(See 20.1.8) setting is 1 (the default), incoming keypresses are ignored if that hotkey subroutine is already running. Thus, when the subroutine finishes, it must wait for the next auto-repeat keypress to come in, which might take 50ms or more due to being caught in between keystrokes of the auto-repeat cycle. This 50ms delay can be avoided by enabling this directive for any hotkey that needs the best possible response time while it is being auto-repeated.

As with all # directives, this one should not be positioned in the script as though it were a command (i.e. it is not necessary to have it contained within a subroutine). Instead, position it immediately before the first hotkey label you wish to have affected by it.

相关命令

[#MaxThreads](#)(See 20.1.6), [#MaxThreadsPerHotkey](#)(See 20.1.8), [\(See 30.19\)](#)[Critical](#)(See 22.7), [Thread \(command\)](#)(See 22.22), [Threads](#)(See 30.19), [Hotkey](#)(See 20.1.10), [#MaxHotkeysPerInterval](#)(See 20.1.5), [#HotkeyInterval](#)(See 20.1.1), [ListHotkeys](#)(See 20.1.11)

示例

```
#MaxThreadsBuffer on

#x::MsgBox, This hotkey will use this type of buffering.

#y::MsgBox, And this one too.

#MaxThreadsBuffer off

#z::MsgBox, But not this one.
```

29.18 #MaxThreadsPerHotkey

设置每个[热键](#)(See 4.)或[热字串](#)(See 5.)能同时启动的[线程](#)(See 30.19)的最大数量。

#MaxThreadsPerHotkey Value

参数

Value	一个给出的热键/热字串子程序能启动的 线程 (See 30.19)的最大数量 (限制 20)。
-------	---

注意

此设置被用来控制一个给出的[热键](#)(See 4.)或[热字串](#)(See 5.)子程序允许同时存在多少“实例”。例如，假设一个热键的最大限制是 1，在它的子程序已经运行的情况下，热键被再次按下，那么第二次按键将被忽略。这对防止意外的重复按键很有帮助。不过，如果你希望这些按键被缓冲而不是被忽略，也许是要增加键盘的自动重复特征的响应性，那么可以使用 [#MaxThreadsBuffer](#)(See 20.1.7)。

与 [#MaxThreads](#)(See 20.1.6) 不同，此设置不是全局性的。将它置于你希望影响的首个热键标签的前面，这样它后面所有的热键都将使用它的值，直到遇上此指令的另一个实例。

任何首行是 [ExitApp](#)(See 17.7), [Pause](#)(See 17.18), [Edit](#)(See 22.9), [Reload](#)(See 20.1.13), [KeyHistory](#)(See 20.9), [ListLines](#)(See 22.11), [ListVars](#)(See 22.12) 或 [ListHotkeys](#)(See 20.1.11) 的[热键](#)(See 4.)子程序将总是忽略此设置而运行。

如果 [#MaxThreads](#)(See 20.1.6) 的设置低于此设置，那么会被优先采用。

如果脚本中没有指定此指令，它将表现得好像被设为 1 那样。

相关命令

[#MaxThreads\(See 20.1.6\)](#), [#MaxThreadsBuffer\(See 20.1.7\)](#), [Critical\(See 22.7\)](#), [Threads\(See 30.19\)](#), [Hotkey\(See 20.1.10\)](#), [#MaxHotkeysPerInterval\(See 20.1.5\)](#), [#HotkeyInterval\(See 20.1.1\)](#), [ListHotkeys\(See 20.1.11\)](#)

示例

```
#MaxThreadsPerHotkey 3
```

翻译：天堂之门 menk33@163.com 2008 年 11 月 24 日

29.19 #NoEnv

避免检查空变量是否为环境变量(推荐所有新建脚本使用)。

`#NoEnv`

指定 `#NoEnv` 这行在脚本的任意位置，防止空变量被作为潜在的环境变量来查找。例如：

```
#NoEnv  
MsgBox %WinDir%
```

上面的不检索 "WinDir" 环境变量(尽管通过在脚本顶部附近加入 `WinDir := A_WinDir` 能解决)。

指定 `#NoEnv` 是推荐所有新建脚本使用的，因为：

1. 当空变量被用在一个表达式或命令里的时候，它显著地改善性能。其同样改善 [DllCall\(See 18.3\)](#) 的性能，当未引用的变量类型被使用的时候(例如 `int` 对 "int")。
2. 它防止由于那些环境变量的名称意外地被脚本使用的变量名匹配而造成的错误。
3. 将来的 AutoHotkey 版本可能让此行为模式作为默认。

为了有助于改善切换到 `#NoEnv` 的情况，内置的 [Comspec\(See 9.\)](#) 和 [ProgramFiles\(See 9.\)](#) 变量被添加进来。它们与相应的环境变量包含同样的字串。

当 `#NoEnv` 起作用时，脚本应该使用 [EnvGet\(See 15.2\)](#) 来找回环境变量，或使用内置的像 [A_WinDir\(See 9.\)](#) 这类变量。

相关命令

[EnvGet\(See 15.2\)](#), [Comspec\(See 9.\)](#), [ProgramFiles\(See 9.\)](#), [A_WinDir\(See 9.\)](#)

翻译：天堂之门 menk33@163.com 2008 年 7 月 21 日

29.20 #NoTrayIcon

不显示一个托盘图标。

#NoTrayIcon

在一个脚本的任何位置指定它，当脚本被运行时将防止那个脚本托盘图标的显示（即使脚本被编译进一个 EXE）。

如果你在一个有热键的脚本使用它，你也许应该捆绑一个热键到 [ExitApp\(See 17.7\)](#) 命令。否则，将没有简单的方法退出程序（除了重启计算机或者终止进程）。例如：#x::ExitApp

托盘图标能在脚本运行的任何时候通过使用命令 [menu\(See 22.13\)](#), [tray, Icon](#) 或者 [menu\(See 22.13\)](#), [tray, NoIcon](#) 来使其消失或再现。在脚本的最顶端使用 [menu\(See 22.13\)](#), [tray, NoIcon](#) 的唯一缺点是托盘图标可能在脚本一开始运行时短暂地可见。要避免这种情况，用 `#NoTrayIcon` 代替。

如果托盘图标当前被隐藏，内置变量 **A_IconHidden** 为 1，或者反之为 0。

相关命令

[Menu\(See 22.13\)](#), [ExitApp\(See 17.7\)](#)

范例

```
#NoTrayIcon
```

翻译：天堂之门 menk33@163.com 2008 年 6 月 13 日

29.21 #Persistent

让脚本持久地运行(就是说，直到用户关闭它或者遇到了 [ExitApp\(See 17.7\)](#) 命令)。

#Persistent

如果这个指令出现在脚本的任意位置，那么脚本在自动执行部分(脚本顶部)完成后会保持运行。当一个脚本包含 [定时器\(See 17.21\)](#) 和/或 [自定义菜单项\(See 22.13\)](#)，但没有 [热键\(See 4.\)](#)、[热字符串\(See 5.\)](#) 或者没有使用任何的 [OnMessage\(\)\(See 18.11\)](#) 或 [Gui\(See 19.3\)](#) 时，它将非常有用处。

如果这个指令被添加到一个已经存在的脚本，你可能需要把一些或全部出现的 [Exit\(See 17.6\)](#) 修改成 [ExitApp\(See 17.7\)](#)。这是因为 [Exit\(See 17.6\)](#) 不能终止一个持续的脚本；它只能终止 [当前的线程\(See 30.19\)](#)。

在 v1.0.16 及之后版本，此指令也让一个脚本成为了 [single-instance\(单独实例\)](#)。要取代它或改变单独实例的运作方式，请看 [#SingleInstance\(See 22.2\)](#)。

相关命令

[#SingleInstance\(See 22.2\)](#), [SetTimer\(See 17.21\)](#), [Menu\(See 22.13\)](#), [Exit\(See 17.6\)](#), [ExitApp\(See 17.7\)](#)

示例

```
#Persistent
```

翻译：天堂之门 menk33@163.com 2008 年 9 月 27 日

29.22 #SingleInstance

在一个脚本已经运行时决定是否允许它再次运行。

```
#SingleInstance [force|ignore|off]
```

参数

	<p>这个参数决定了脚本的上个实例正在运行时，启动该脚本将会发生什么动作：</p> <p>单词 FORCE 将跳过对话框，并自动地替换旧的实例，其实际上同 Reload(See 20.1.13) 命令相似。</p> <p>单词 IGNORE 将跳过对话框，并让旧的实例运行。也就是说，试图启动一个已运行的脚本将被忽略。</p> <p>单词 OFF 允许多个脚本实例同时运行。</p> <p>如果这个参数被省略，将显示一个对话框，询问是保持旧的实例还是使用新的实例来替换它。</p>
--	---

注意

一个包含 [hotkeys\(See 4.\)](#)、[hotstrings\(See 5.\)](#)、[#Persistent\(See 29.21\)](#)、[OnMessage\(\)\(See 18.11\)](#) 或者 [Gui\(See 19.3\)](#) 命令的脚本，默认是单实例的。使用上述方法能取消或修改这一特性。

相关命令

[Reload\(See 20.1.13\)](#), [#Persistent\(See 29.21\)](#)

示例

```
#SingleInstance force
#SingleInstance ignore
#SingleInstance off
```

翻译：坛友 lwjiejie 修正：天堂之门 menk33@163.com 2008 年 8 月 3 日

29.23 #UseHook

强制使用钩子来实现部分或全部键盘热键(See 4.)。

```
#UseHook [On|Off]
```

参数

On Off	<p>#UseHook 后不带下列单词的话就等同于 #UseHook On。</p> <p>On: 将用键盘钩子(See 20.2)来实现此处和下个 #UseHook OFF (如果有的话)之间所有的键盘热键。</p> <p>Off: 将用默认的方法(如果可用 <code>RegisterHotkey()</code> 的话; 否则会用键盘钩子)实现热键。</p>
--------	---

注意

通常只要可能, 都会用 windows API 函数 `RegisterHotkey()` 来实现键盘热键。不过在某些条件下, 如果用[键盘钩子](#)(See 20.2)来代替的话, 热键的响应性可能会更好。

将此指令调为 ON 就相当于在每个受影响热键的定义中使用 [\\$ 前缀](#)(See 4.)。Windows 95/98/Me 例外, 它们忽略 #UseHook (虽然 [\\$ 前缀](#)(See 4.)能起有限的作用)。

和所有在脚本启动时仅执行一次的 # 指令一样, #UseHook 在脚本中不应该像命令那样被放置(也就是说, 没必要将它包含在子程序中)。而是将它放在你想要影响的首个热键标签前。

使用[键盘钩子](#)(See 20.2)的热键不能被[Send 命令](#)(See 20.12)触发。相似地, 鼠标热键也不能被比如[Click](#)(See 23.1) 命令触发, 因为所有的鼠标热键都使用[鼠标钩子](#)(See 20.3)。要绕弯解决这种情况, 可用[Gosub](#)(See 17.8) 直接跳转到热键的子程序。例如: `Gosub #LButton`

如果脚本中没有出现此指令, 那么将表现得像设成了 OFF。

相关命令

[#InstallKeybdHook](#)(See 20.2), [#InstallMouseHook](#)(See 20.3), [ListHotkeys](#)(See 20.1.11)

示例

```
#UseHook ;在此点后强制给热键使用钩子。
#x::MsgBox, 此热键将用钩子实现。
#y::MsgBox, 这个也一样。
#UseHook off
#z::MsgBox, 但这个没用。
```

翻译: 天堂之门 menk33@163.com 2008 年 11 月 22 日

29.24 #WinActivateForce

跳过温和的方法激活窗口而直接使用强硬的方法。

```
#WinActivateForce
```

在脚本的任意位置指定此指令将促使激活窗口的命令 -- 例如 [WinActivate\(See 28.12\)](#), [WinActivateBottom\(See 28.13\)](#), 和 [GroupActivate\(See 28.2.1\)](#) -- 跳过 "温和的" 方法激活窗口而直接使用更强硬的方法。

虽然这个指令通常不会改善如何快速地或可靠地激活窗口，但它可以防止任务栏按钮在不同窗口接连快速地激活时的闪烁。

Windows 95 和 **NT** 将可能永不需要此设置，因为它们在准许窗口被激活方面更自由。

相关命令

[WinActivate\(See 28.12\)](#), [WinActivateBottom\(See 28.13\)](#), [GroupActivate\(See 28.2.1\)](#)

示例

```
#WinActivateForce
```

翻译：天堂之门 menk33@163.com 2008 年 9 月 5 日

30. Addenda

30.1 高级快捷键功能- 鼠标和键盘快捷键

注意：下述功能中，多数需要 Windows NT/2000/XP 或更高版本操作系统。

让最容易按的键变得更有用：有些键占了好位置，易于按下，但却不常用。为什么不改造它们呢？比如，如果你的右 ALT 键很不常用，则不妨为它指定一个常用的操作：

```
RAlt:::
```

```
MsgBox 你按下了右 ALT 键。
```

```
return
```

另外，也可以把右 Alt 作为一个“引导键”，再加上一个其他键组成快捷键。这样的好处是不会占用右 Alt 的本来功能。在下面的例子中，右 Alt 就成为了引导键，和 j 一起，实现了切换窗口的功能。作为引导键，它可以和任何其他键一起按下发挥作用。而没有其他键按下时，它仍然能发挥自己的作用（如上面的例子）：

```
RAlt & j::AltTab
```

修饰键不再局限于 CTRL、ALT、SHIFT 和 WIN：你可以使用任意两个按键或鼠标按键组成自定义快捷键。例如，按住小键盘数字 0，再按下数字 1 就形成了一个快捷键（写为：Numpad0 & Numpad1::）；按住

CapsLock, 再按下另一个键或用鼠标点击右键 (写为: **CapsLock & RButton::**)。这种情况下, CapsLock 键的状态 (大写/小写) 不会发生改变。详见 [自定义组合键\(See 4.\)](#)。

用鼠标滚轮 (或其他键) 代替 Alt-Tab: 点击鼠标滚轮, 就可以显示窗口选择菜单; 上下滚动, 就在各窗口图标进行选择; 选中之后再次按下滚轮, 则切换到选中的窗口。并且, 在 Alt-Tab 菜单不出现时, 鼠标滚轮的正常功能仍然可以使用。脚本:

```
MButton::AltTabMenu
```

```
WheelDown::AltTab
```

```
WheelUp::ShiftAltTab
```

让按键变成鼠标键, 或者在按住某个键或鼠标键时, 让计算机重复某个动作。示例见 [remapping page\(See 6.\)](#)。

让快捷键区分情境: 可以让最易使用的快捷键在不同的程序中, 调用不同的最常用功能。下面的脚本可以让右 Ctrl 键在记事本和计算器中, 分别实现不同功能:

```
#IfWinActive ahk_class Notepad
```

```
RControl::WinMenuItem, , , 文件, 保存 ; 在记事本中保存当前文件
```

```
#IfWinActive ahk_class SciCalc
```

```
RControl::Send, ^c!{tab}^v ; 在计算器中, 把计算结果复制到原来的活动窗口
```

详见 [#IfWinActive\(See 20.1.4\)](#)。

热字符串: 实现缩写功能。无须经过任何培训, 就可以写出这样的脚本。例如, 下述脚本将自动把 **ceo, cfo,** 和 **btw** 替换成完整拼写: (注: 暂不支持汉字)

```
::ceo::Chief Executive Officer
```

```
::cfo::Chief Financial Officer
```

```
::btw::by the way
```

(更详内容) (See 5.)

如果你对游戏感兴趣:

- 别人拼血流汗时, 你只需动一下手指, 就能发出一系列复杂操作。因为常用操作可以定义到几乎任意键(See 7.)上, 包括单一的字母键、方向键、数字键, 甚至是特殊键 (CTRL/ALT/WIN/SHIFT)。

- 为鼠标键，包括滚轮（MButton）、上下左右滚动（WheelUp、WheelDown、WheelLeft 和 WheelRight）进行快捷键定义。还可以将鼠标键盘结合起来。如，按住 **ctrl** 点击右键可写为 **^RButton::**
- 创建“穿越型”快捷键。例如，点击鼠标左键，一方面触发快捷键动作，另一方面，仍和往常一样，游戏也接收到了左键单击（写为：**~LButton::**）。
- 使用诸如 [PixelSearch\(See 22.16\)](#)（像素搜索）、[PixelGetColor\(See 22.15\)](#)（取得像素颜色）、[ImageSearch\(See 22.10\)](#)（搜索图像）这样的命令来实现自动操作。
- 可以选择使用 [keyboard hook\(See 20.1.9\)](#) 来实现快捷键。在游戏占用较多 CPU 资源时，此方法比其他方法的响应性更好。它还能覆盖游戏软件本身对哪些键可以映射的限制。

更多信息，请见 [Hotkeys\(See 4.\)](#) 章节。

30.2 Last Found Window

这是被 [IfWin\[Not\]Exist\(See 28.7\)](#), [IfWin\[Not\]Active\(See 28.6\)](#), [WinWait\[Not\]Active\(See 28.33\)](#) 或 [WinWait\(See 28.32\)](#) 命令最近找到的窗口。由于不需要在各种窗口命令里再重复目标窗口的 WinTitle 和 WinText 参数，所以它可以使脚本更易于创建和维护。此外，因为在首次找到目标窗口后不需要再将其搜寻一遍，所以脚本执行得更好。

所有窗口命令都可以使用“最近找到”的窗口，除了 [WinWait\(See 28.32\)](#), [WinActivateBottom\(See 28.13\)](#) 和 [GroupAdd\(See 28.2.2\)](#)。要使用它的话，只需省略掉全部的四个窗口参数(WinTitle, WinText, ExcludeTitle 和 ExcludeText)。

每个[线程\(See 30.19\)](#)都保留了它自有的“最近找到”窗口的值，就是说如果[当前线程\(See 30.19\)](#)被另一个线程打断，那么在原来的线程恢复时，它仍将保持它原本的“最近找到”窗口的值，而不是那个中断它的线程的值。

如果最近找到的窗口是一个隐藏的 [Gui 窗口\(See 19.3\)](#)，那么即使在 [DetectHiddenWindows\(See 28.5\)](#) 处于 Off 状态时，它也能被使用。这常与"[Gui +LastFound\(See 19.3\)](#)"一起使用。

示例

```
Run Notepad
WinWait 无标题 - 记事本
WinActivate ;使用最近找到的窗口。

IfWinExist, 无标题 - 记事本
{
    WinActivate ;自动地使用上面找到的窗口。
    WinMaximize ;同上
    Send, Some text.{Enter}
}

return
```

```

}

IfWinNotExist, 计算器

    return

else

{

    WinActivate ;上面的 "IfWinNotExist" 也为我们设置了"最近找到"的窗口。

    WinMove, 40, 40 ;将它移动到一个新的位置。

    return

}

```

如果有多种窗口符合一个窗口命令比如 [WinMove\(See 28.27\)](#) 的 `WinTitle/Text` 条件，那么在窗口堆层次中位于其它窗口上面的那个窗口将被使用。例如，如果激活的/最前面的窗口匹配条件，就会用它，即使它下面还有其它匹配的窗口。

激活的窗口 (A): 几乎所有的窗口命令都能被告知对激活的窗口进行操作，只要指定它们的 `WinTitle` 参数为字母 A 并且省略掉 `WinText`, `ExcludeTitle` 和 `ExcludeText`。在下例中，`Win+上箭头` 变成一个最大化当前激活窗口的热键：#Up::WinMaximize A

窗口类 (ahk_class): 通过 Window Spy 显示的或者由 [WinGetClass\(See 28.18\)](#) 获得的类名都能使所有的窗口命令对一个窗口类进行操作。在下例中，一个分割视图的资源管理器窗口（译注：即左侧为文件夹树）将被激活：`WinActivate ahk_class ExploreWClass`

唯一 ID/HWND (ahk_id): 通过唯一 ID 编号，所有的窗口命令都能对特定的窗口或者控件进行操作。例如：`WinActivate ahk_id %VarContainingID%`。一个窗口的 ID 一般通过 [WinExist\(\)\(See 28.7\)](#) 或者 [WinGet\(See 28.15\)](#) 来取得。一个控件的 ID 一般通过 [ControlGet Hwnd\(See 28.1.4\)](#), [MouseGetPos\(See 23.5\)](#) 或 [DllCall\(See 18.3\)](#) 来取得。而且，即使控件是隐藏的，ahk_id 也将对它们进行操作；就是说，[DetectHiddenWindows\(See 28.5\)](#) 的设置无关紧要。

进程 ID (ahk_pid): 所有的窗口命令都能对属于一个进程标识符(PID)的窗口进行操作。例如，`WinClose ahk_pid %VarContainingPID%` 会关闭那个进程的最上层窗口。PID 能通过 [WinGet\(See 28.15\)](#) 或 [Run\(See 24.5\)](#) 或 [Process\(See 24.4\)](#) 来获得。

窗口组 (ahk_group): 通过给 `WinTitle` 参数指定 `ahk_group MyGroupName`，所有的窗口命令都能对属于一个窗口组([See 28.2.2](#))的窗口进行操作。[WinMinimize\(See 28.25\)](#), [WinMaximize\(See 28.24\)](#), [WinRestore\(See 28.28\)](#), [WinHide\(See 28.22\)](#), [WinShow\(See 28.31\)](#), [WinClose\(See 28.14\)](#) 和 [WinKill\(See 28.23\)](#) 命令会对组内**所有的**窗口起作用。相比之下，其它窗口命令例如 [WinActivate\(See 28.12\)](#) 和 [IfWinExist\(See 28.7\)](#) 将只对组内最上层的窗口进行操作。

多重条件: 和上面段落中的 `ahk_group` (其扩大了搜索)相比, 只要在 `WinTitle` 参数中指定多个条件就能缩小搜索范围。在下例中, 脚本等待标题包含 `My File.txt` 以及类是 `Notepad` 的窗口出现:

```
WinWait My File.txt ahk_class Notepad
```

```
WinActivate ;激活它找到的窗口。
```

当使用这方法时, 应该首先列出标题的文字(如果有的话就需要), 后跟一个或多个附加的条件。超出首个的条件必须精确地用一个空格或者 `tab` 来与前一个分开(任何其他的空格或 `tab` 将被看作前一个条件的一部分)。

翻译: 天堂之门 menk33@163.com 2008 年 11 月 5 日

30.3 License

GNU 通用公共授权

1991 年 6 月, 第 2 版

著作权所有 (C) 1989, 1991 Free Software Foundation, Inc.

675 Mass Ave, Cambridge, MA 02139, USA

允许每个人复制和发布本授权文件的完整副本,

但不允许对它进行任何修改。

导言

大多数软件授权声明是设计用以剥夺您共享与修改软件的自由。

相反地, GNU 通用公共授予权力图保证您分享与修改自由软件的自由,
确保软件对所有的使用者都是自由的。

通用公共授权适用于大多数自由软件基金会的软件,

以及任何作者指定使用本授权的其他软件。(有些自由软件基金会的软件, 则适用 GNU 函式库通用公共授予权规定。)

您也可以让您的软件适用本授权规定。

当我们在谈论自由软件时, 我们所指的是自由, 而不是价格。

我们的通用公共授权是设计用以确保使您拥有发布自由软件备份的自由(以及您可以决定此一服务是否收费),
确保您能收到源码或者在您需要时能得到它,
确保您能变更软件或将它的一部分用于新的自由软件;
并且确保您知道您可以做上述的这些事情。

为了保障您的权利, 我们需要作出限制:

禁止任何人否认您上述的权利, 或者要求您放弃这些权利。

如果您发布软件的副本, 或者对之加以修改, 这些限制就转化成为您的责任。

例如, 假如您发布此类程序的副本, 无论是免费或收取费用,
您必须将您所享有的一切权利给予收受者。
您也必须确保他们也能收到或得到原始程序码。

而且您必须向他们展示这些条款的内容，使他们知到他们所享有的权利。

我们采取两项措施來保护您的权利：(1)以著作权保护软件，
以及(2)提供您本授权，赋与您复制、发布并且 / 或者修改软件的法律许可。

同时，为了保护作者与我们（按：指自由软件基金会），
我们想要确定每个人都明白，自由软件是沒有担保责任的。
如果软件被他人修改并加以传播，
我们需要其收受者知道，他们所得到的并非原始版本，
因此由他人所引出的任何问题对原作者的声誉将不会有任何的影响。

最后，所有自由软件不断地受到软件专利的威胁。
我们希望能避免自由软件的再发布者以个人名义取得专利授权而使程序专有化的风险。
为了防止上述的情事发生，我们在此明确声明：
任何专利都必须为了每个人的自由使用而核准，否则就不应授与专利。

以下是有复制、发布及修改的明确条款及条件。

GNU 通用公共授权

复制、发布与修改的条款与条件

0. 凡著作权人在其程序或其他著作中声明，
该程序或著作会在通用公共授权条款下发布，本授权对其均有适用。
以下所称的"程序"，是指任何一种适用通用公共授权的程序或著作；
并且一个"基于本程序的著作"，则指本程序或任何基于著作权法所产生的衍生著作，
换言之，是指包含本程序全部或一部的著作，不论是完整的或经过修改的程序，
以及（或）翻译成其他语言的程序（以下"修改"一词包括但不限于翻译行为在内）。被授权人则称为"您"。

本授权不适用于复制、发布与修改以外的行为；
这些行为不在本授权范围内。
执行本程序的行为并不受限制，
而本程序的输出只有在其内容构成基于本程序所生的著作（而非只是因为执行本程序所造成）时，
始受本授权拘束。至于程序的输出内容是否构成本程序的衍生著作，则取决于本程序的具体用途。

1. 您可以对所收受的本程序源代码，无论以何种媒介，复制与发布其完整的复制物，
然而您必须符合以下要件：
以显著及适当的方式在每一份复制物上发布适当的著作权标示及无担保声明；
维持所有有关本授权以及无担保声明的原貌；
并将本授权的副本连同本程序一起交付予其他任何一位本程序的收受者。

您可以对让与复制物的实际行为收取一定的费用，您也可以自由决定是否提供担保以作为对价的交换。

2. 您可以修改本程序的一个或数个复制物或者本程序的任何部份，
以此形成基于本程序所生的著作，并依前述第一条规定，复制与发布此一修改过的程序或著作，

但您必须符合以下要件：

a) 您必须在所修改的档案上附加显著的标示，阐明您修改过这些档案，以及修改日期。

b) 您必须就您所发布或发行的著作，无论是包含本程序全部或一部的著作，

或者是自本程序或其任何部份所衍生的著作，

整体授权所有第三人依本授权规定使用，且不得因此项授权行为而收取任何费用。

c) 若经过修改的程序在执行时通常以交互方式读取命令时，

您必须在最常被使用的方式下，于开始进入这种交互式使用时，列印或展示以下宣告：

适当的著作权标示及无担保声明（或者声明您提供担保）、使用者可以依这些条件再发布此程序，

以及告知使用者如何浏览本授权的副本。

（例外：若本程序本身是以交互的方式执行，然而通常却不会列印该宣告时，则您基于本程序所生的著作便无需列印该宣告。）

这些要求对修改过的著作是整体适用的。

倘著作中可识别的一部份并非衍生自本程序，并且可以合理地认为是一独立的、个别的著作，

则当您将其作为个别著作加以发布时，本授权及其条款将不适用于该部分。

然而当您将上述部分，作为基于本程序所生著作的一部而发布时，

整个著作的发布必须符合本授权条款的规定，

而本授权对于其他被授权人所为的许可及于著作整体。

因此，本条规定的意图不在于主张或剥夺您对于完全由您所完成著作的权利；

应该说，本条规定意在行使对基于程序所生之衍生著作或集合著作发布行为的控制权。

此外，非基于本程序所生的其他著作与本程序（或基于本程序所生的著作）

在同一储存或发布的媒介上的单纯聚集行为，并不会使该著作因此受本授权条款约束。

3. 您可以依前述第一、二条规定，复制与发布本程序（或第二条所述基于本程序所产生的著作）的目的码或可执行形式，但您必须符合以下要件：

a) 附上完整、相对应的机器可判读源码，

而这些源码必须依前述第一、二条规定在经常用以作为软件交换的媒介物上发布；或

b) 附上至少三年有效的书面报价文件，

提供任何第三人在支付不超过实际发布源码所需成本的费用下，

取得相同源码的完整机器可读复制物，

并依前述第一、二条规定在经常用以作为软件交换的媒介物上发布该复制物；或

c) 附上您所收受有关发布相同源码的报价资讯。

（本项选择仅在非赢利发布、且仅在您依前述 b 项方式自该书面报价文件收受程序目的码或可执行形式时，始有适用。）

著作的源码，是指对著作进行修改时适用的形式。

对于一个可执行的著作而言，完整的源码是指著作中所包含所有模组的全部源码，加上相关介面的定义档，还加上用以控制该著作编译与安装的描述。然而，特别的例外情况是，所发布的源码并不需包含任何通常会随著所执行作业系统的主要组成部分（编译器、核心等等）而发布的软件（无论以源码或二进位格式），除非该部分本身即附加在可执行程序中。

若可执行码或目的码的发布方式，是以指定的地点提供存取位置供人复制，则提供可自相同地点复制源码的使用机会，视同对于源码的发布，然而第三人并不因此而负有将目的码连同源码一起复制的义务。

4. 除本授权所明示的方式外，您不得对本程序加以复制、修改、再授权或发布。任何试图以其他方式进行复制、修改、再授权或者发布本程序的行为均为无效，并且将自动终止您基于本授权所得享有的权利。然而，依本授权规定自您手中收受复制物或权利之人，只要遵守本授权规定，他们所获得的授权并不会因此终止。

5. 因为您并未在本授权上签名，所以您无须接受本授权。然而，除此之外您别无其他修改或发布本程序或其衍生著作的授权许可。若您不接受本授权，则这些行为在法律上都是被禁止的。因此，藉由对本程序（或任何基于本程序所生的著作）的修改或发布行为，您表示了对于本授权的接受，以及接受所有关于复制、发布或修改本程序或基于本程序所生著作的条款与条件。

6. 每当您再发布本程序（或任何基于本程序所生的著作）时，收受者即自动获得原授权人所授予依本授权条款与条件复制、发布或修改本程序的权利。您不得就本授权所赋予收受者行使的权利附加任何进一步的限制。您对于第三人是否履行本授权一事，无须负责。

7. 若法院判决、专利侵权主张或者其他任何理由（不限于专利争议）的结果，使得加诸于您的条件（无论是由法院命令、协议书或其他方式造成）与本授权规定有所冲突，他们并不免除您对于本授权规定的遵守。若您无法同时符合依本授权所生义务及其他相关义务而进行发布，那么其结果便是您不得发布该程序。例如，若专利授权不允许其他人直接或间接取得复制物，通过您以免付权利金的方式再发布该程序，您唯一能同时满足该义务及本授权的方式就是彻底避免进行该程序的发布。

若本条任一部份在特殊情况下被认定无效或无法执行时，本条其余部分仍应适用，且本条全部于其他情况下仍应适用。

本条的目的并不在诱使您侵害专利或其他财产权的权利主张，或就此类主张的有效性加以争执；本条的唯一目的，是在保障藉由公共授权惯例所执行自由软件发布系统的完整性。许多人信赖该系统一贯使用的应用程序，而对经由此系统发布的大量软件有相当多的贡献；作者 / 贡献者有权决定他或她是否希望经由其他的系统发布软件，而被授权人则无该种选择权。

本条的用意在于将本授权其他不确定的部分彻底解释清楚。

8. 若因为专利或享有著作权保护的介面问题，而使得本程序的发布与 / 或使用局限于某些国家时，则将本程序置于本授权规范之下的原著作权人得增列明确的发布地区限制条款，将这些国家排除在外，而使发布的许可只限在未受排除的国家之内或之中。在该等情况下，该限制条款如同以书面方式订定于本授权内容中，而成为本授权的条款。

9. 自由软件基金会得随时发表通用公共授权的修正版与 / 或新版本。新版本在精神上将近似于目前的版本，然而在细节上或所不同以因应新的问题或状况。

每一个版本都有个别的版本号码。若本程序指定有授权版本号码，表示其适用该版本或是"任何新版本"时，您可以选择遵循该版本或任何由自由软件基金会日后所发表新版本的条款与条件。若本程序并未指定授权版本号码时，您可以选择任一自由软件基金会所发表的版本。

10. 若您想将部分本程序纳入其他自由程序，而其发布的条件有所不同时，请写信取得作者的许可。若为自由软件基金会享有著作权的软件，请写信至自由软件基金会；我们有时会以例外方式予以处理。我们的决定取决于两项目标：确保我们自由软件的所有衍生著作均维持在自由的状态，并广泛地促进软件的分享与再利用。

无担保声明

11. 由于本程序是无偿授权，因此在法律许可范围内，本授权对本程序并不负担保责任。非经书面声明，著作权人与 / 或其他提供程序之人，无论明示或默许，均是依「现况」提供本程序而并无任何形式的担保责任，其包括但不限于，就适售性以及特定目的的适用性为默示性担保。有关本程序品质与效能的全部风险均由您承担。如本程序被证明有瑕疵，您应承担所有服务、修复或改正的费用。

12. 非经法律要求或书面同意，任何著作权人或任何可能依前述方式修改与 / 或发布本程序者，对于您因为使用或不能使用本程序所造成的一般性、特殊性、意外性或间接性损失，不负任何责任（包括但不限于，资料损失，资料执行不精确，或应由您或第三人承担的损失，或本程序无法与其他程序运作等），即便前述的著作权人或其他人已被告知该等损失的可能性时，也是一样。

条文结束

30.4 AutoHotkey Acknowledgements

除了已经 [提到的](#)(See 1.) AutoIt 作者，此外还有：

Robert Yaklin: 不知疲倦地测试大量孤立 bug，一个伟大的第一版安装者，以及对命令应该如何工作提出过极好的建议 :)

Jason Payam Ahdoot: 提供浮点数运算支持的建议和描述。

Jay D. Novak: 发现很多 Send 命令、Capslock 和 热键修饰符 在 Win9x 下的问题；并慷慨地分享了他大量的代码以及在热键、热字符串、钩子应用和输入加速方面的学问。

Rajat: 为原 AHK 图标制作了时尚的替换物；一个伟大的产品标志；制作了 TextPad 的自定义语法；发现注册表命令和 AutoIt v2 的兼容性的一些 bug；制作了 SmartGUI Creator；以及很多其它事情。

Beardboy: 为修补 GetKeyState 和 Send 命令做了 NT4 的测试工作；在论坛上提供了大量的帮助；以及很多建议和 bug 报告。

Gregory F. Hogg of Hogg's Software: 编写了 [SysGet](#)(See 22.21) 命令中多显示器支持的源代码。

Aurelian Maga: 编写了 [ImageSearch](#)(See 22.10) 和更快的 [PixelSearch](#)(See 22.16) 的源代码。

Joost Mulders: 他的源代码为 表达式(See 9.) 提供了基础。

Laszlo Hars: 提供了关于数据结构和算法方面的建议，有利于极大地加速数组和动态变量。

Marcus Sonntag (Ultra): 研究、设计、编码并且测试了 [DIIICall](#)(See 18.3)。

Gena Shimanovich: 协助编码和调试；并提供了一些在未来的特性上可能作为基础的高级脚本原型。

Eric Morin (numEric): 提出了数学方面的建议；在 [OnMessage](#)(See 18.11)、浮点数运算以及鼠标移动方面，做了坚定的调试工作；以及对整体品质的改善。

Philip Hazel: 提供 [Perl-Compatible Regular Expressions \(PCRE\)](#) (兼容 Perl 的正则表达式)。

Titan: 提供了在 [autohotkey.net](#) 的社区托管，创造了很多有用的脚本和类库，在论坛上彩色显示代码注解的 JavaScript 以及许多其它事情。

Philippe Lhoste (PhiLho): 在论坛上不知疲劳的 moderation(温和) 和支持，测试并建议 RegEx，语法和设计理念，以及其它很多事情。

John Biederman: 显著地改善了文档的演示和人类工程学。

Jonathan Rennison (JGR): 发展了 [RegisterCallback\(\)](#)(See 18.14)，并提供了有益的建议。

Steve Gray (Lexikos): 发展了动态函数调用及其它新的功能；分析并修补 bug；对论坛上成百上千单独的游客提供宝贵的支持和意见。

同样对其他所有提交 bug 报告或建议的人：谢谢！

翻译：坛友 lwjeee 修正：天堂之门 menk33@163.com 2008 年 8 月 3 日

30.5 Arrays

在 AutoHotkey 中，数组主要是概念上的：每个数组实际上只是一系列连续编号的变量(See 9.)或函数(See 10.)，它们每一个都被视为数组中的一个元素。AutoHotkey 不会将这些元素连接在一起。

除了像 [StringSplit\(See 27.21\)](#) 和 "[WinGet List\(See 28.15\)"](#) 这种数组创建命令外，任何能接受 [OutputVar](#) 参数或者可以为变量赋值的命令都可以被用来创建一个数组。最简单的例子是[赋值运算符\(:=\)\(See 21.12\)](#)，如下所示：

```
Array%j% := A_LoopField
```

通过在 [indices](#) (索引)之间使用一个你任选的分隔符，可以创建多维数组。例如：

```
Array%j%_%k% := A_LoopReadLine
```

下面的例子演示了如何创建并访问一个数组，在这里，是从一个文本文件获取一系列的名称：

```
;写入数组:
ArrayCount = 0
Loop, Read, C:\Guest List.txt ;此循环从文件中逐行获取内容。
{
    ArrayCount += 1 ;记录在数组中有多少个项目。
    Array%ArrayCount% := A_LoopReadLine ;将此行内容存入下一个数组元素中。
}

;从数组读取:
Loop %ArrayCount%
{
    ;下面这行使用了 := 运算符来获取一个数组元素:
    element := Array%A_Index% ; A\_Index\(See 9.\) 是一个内置变量。
    ;或者，你可以使用 "% " 前缀，从而让 MsgBox 或其他一些命令支持表达式(See 9.):
    MsgBox % "元素编号 " . A_Index . " 是 " . Array%A_Index%
}
```

~~数组也可以储存函数名称，以便之后对其动态调用(See 10.)~~

一个和数组相关的概念是使用 [NumPut\(\)\(See 10.\)](#) 和 [NumGet\(\)\(See 10.\)](#) 来储存/获取一批二进制格式的数字。在对性能和/或内存占有有要求时，会比较有用。

"Scripting.Dictionary" 是一个比 AutoHotkey 的伪数组拥有更多功能更灵活的操作系统特性。它的使用详见 www.autohotkey.com/forum/topic17838.html。

翻译：单菜子 修正：天堂之门 menk33@163.com 2009 年 1 月 21 日

30.6 Clipboard, ClipboardAll, 以及 OnClipboardChange

Clipboard 是一个内置变量(See 9.)，存储了当前 Windows 剪贴板中可以以文本形式表现的内容。相应的，*ClipboardAll* 存储了剪贴板中的所有内容，例如图片以及带格式的文本。

Clipboard 中的每行文本通常都以回车和换行(CR+LF)结尾，在脚本中可表示为 `r`n。文件(例如那些用 Control-C 从资源管理器中复制的文件)是用文本形式表现的：每当在脚本中引用 *Clipboard* 变量时，它们都会被自动转换为它们的文件名称(带完整路径)。要逐个提取文件的话，参照下例：

```
Loop, parse(See 17.14), clipboard, `n, `r
{
    MsgBox, 4, , 第 %A_Index% 个文件是 %A_LoopField%.`n`n是否继续?
    IfMsgBox, No, break
}
```

要将文件名按字母顺序排列，使用 *Sort*(See 27.12) 命令。要将剪贴板上的文件名写入文件，可用 *FileAppend*(See 16.4)，%clipboard%`r`n, C:\My File.txt。要改变让脚本试着保持打开剪贴板的这个时间 -- 例如在另一个程序使用剪贴板时 -- 请用 *#ClipboardTimeOut*(See 29.2) 命令。

基础示例：

```
clipboard = my text ;给予剪贴板全新的内容。
clipboard = ;清空剪贴板。
clipboard = %clipboard% ;将任何复制的文件、HTML 或者其它带格式的文本转换为纯文本。
clipboard = %clipboard% 添加的文本。 ;在剪贴板中追加些文本。
StringReplace, clipboard, clipboard, ABC, DEF, All ;将剪贴板中所有的 ABC 替换为 DEF
(同时也将剪贴板中的内容转换为纯文本)。
```

使用 **ClipWait** 命令改善脚本的可靠性：

```
clipboard = ;首先清空剪贴板，使 ClipWait 命令能检测到文本何时被复制到了剪贴板中。
Send ^c
ClipWait(See 15.1) ;等待文本被复制到剪贴板中。
MsgBox Control-C 复制了下列内容到剪贴板中: `n`n%clipboard%
```

ClipboardAll 存储了剪贴板中的所有内容(例如图片以及带格式的文本)。它最常用来存储剪贴板的内容以便脚本为了某个操作能临时使用剪贴板。当这个操作结束时，脚本再像下面这样将剪贴板原本的内容恢复：

```

ClipSaved := ClipboardAll ;将剪贴板的所有内容存储到你选的一个变量里。
;... 这里放临时使用剪贴板的脚本，例如用转换 Unicode(See 21.11) 命令粘贴 Unicode 编码
;的文本...
Clipboard := ClipSaved ;恢复剪贴板原本的内容。注意使用的是 Clipboard (不是
ClipboardAll)。
ClipSaved = ;释放内存以防原来剪贴板里是巨大的内容。

```

ClipboardAll 中的内容也可以存成一个文件(在这种模式下, `FileAppend` 命令会覆盖已存在的文件):
`FileAppend(See 16.4), %ClipboardAll%, C:\Company Logo.clip` ; 文件扩展名无关紧要。

要在之后将文件重新读取回剪贴板(或读取到一个变量中), 参照下例:

`FileRead(See 16.19), Clipboard, *c C:\Company Logo.clip` ;注意必须在文件名之前使用 `*c`。

注意

在上面所描述的情况之外调用 *ClipboardAll* 都会返回空值。另外, *ClipboardAll* 不能写在逗号分隔的表达式(See 9.)中; 也就是说, 它必须独占一行, 例如 `ClipSaved := ClipboardAll`。

赋值了 *ClipboardAll* 的变量将用二进制格式来存储数据, 因此使用 `MsgBox(See 19.11)` 或类似命令查看其内容时会显示乱码。并且, 修改二进制变量的内容(例如使用 `StringReplace(See 27.20)` 命令)会将其还原为普通变量, 结果会丢失其存储的剪贴板数据。在 v1.0.46 及之后的版本中, 二进制变量可以通过数值传递给函数(See 10.)(之前它们只能使用引用传递(See 10.)的方式)。

如果 *ClipboardAll* 不能获取剪贴板中的一个或多个对象(格式化的), 那么它们将被省略, 而所有剩余的对象仍然会被存储。获取行为会根据 `#ClipboardTimeOut(See 29.2)` 命令的设定再次尝试(而不是只尝试一次)。

一个包含了剪贴板数据的变量可以被复制到另一个变量中, 例如: `ClipSaved2 := ClipSaved`

`ClipWait(See 15.1)` 命令可以用来检测剪贴板何时有内容(包括非文本数据也可选)。

`StringLen(See 27.17)` 命令可以用来显示赋值了 *ClipboardAll* 的变量的总大小。

赋值了 *ClipboardAll* 的变量可以使用 `<>` 和 `=` 运算符来相互比较(但不能直接与 *ClipboardAll* 进行比较)。在下例中, 首先检查每个变量的大小。如果根据大小不能进行判断, 就将两个变量的内容进行比较:

```
if ClipSaved1 <> %ClipSaved2% ;这里必须使用老式的 IF 语句, 不能使用表达式。
```

```
MessageBox 两个变量中存储的剪贴板的内容不相同。
```

将 *ClipboardAll* 的内容存储到一个变量时, 变量大小不受 `#MaxMem(See 29.15)` 命令设置的最大内存限制。

一个存储了剪贴板内容的文件由四个字节的格式类型, 接着四个字节的数据块大小, 接着那个格式的数据块而组成。如果剪贴板包含多种格式的内容(大部分时候都是这样), 前面所述的三个组成部分会重复下去, 直到存储了所有的格式。文件最后以一个四个字节的 0 格式类型结束。

已知局限: 当剪贴板中有 Microsoft Excel 单元格的内容时, 获取 *ClipboardAll* 会导致 Excel 弹出 "找不到打印机" 的对话框。

如果存在名称为 `OnClipboardChange` 的标签，那么当任何程序(甚至是脚本自身)更改了剪贴板的内容的时候，该标签下的脚本都会自动执行。在脚本加载的时候该标签同样会执行一次。

内置变量 `A_EventInfo` 包含：

- 0 如果剪贴板当前为空；
- 1 如果剪贴板中的内容能够以文本形式表达(包括从资源管理器中[复制的文件](#))；
- 2 如果整个剪贴板里是非文本内容，例如一幅图片。

下面的例子就是一个可用的脚本。它会在剪贴板内容每次改变的时候短暂地显示一个提示。

```
#Persistent
return

OnClipboardChange:
ToolTip 剪贴板中的数据类型为: %A_EventInfo%
Sleep 1000
ToolTip ;关闭提示。
return
```

如果正在执行 `OnClipboardChange` 标签下的脚本时剪贴板中的内容发生了变化，那么那个提示事件将会丢失。如果需要避免这种情况，在 `OnClipboardChange` 标签下的首行使用 [Critical](#)(See 22.7)。不过，在 `OnClipboardChange` 线程运行这个时段产生的其它[线程](#)(See 30.19)(例如按了个热键)也会被缓冲或延迟。

如果脚本本身更改了剪贴板的内容，则脚本中的 `OnClipboardChange` 标签往往不会立即执行；就是说，更改剪贴板的命令的下一行命令很可能提前执行。要强制让 `OnClipboardChange` 标签立即执行的话，可以在更改了剪贴板之后添加一个短暂的延时例如 [Sleep 20](#)(See 17.22)。

相关内容：[OnExit](#)(See 17.17), [OnMessage\(\)](#)(See 18.11), [RegisterCallback\(\)](#)(See 18.14)

翻译：okey3m 修正：天堂之门 menk33@163.com 2008年10月26日

30.7 CLSID List (Windows Class Identifiers)

操作系统内某些特殊的文件夹是通过唯一的字符串来识别的。这些字符串中的一些字串可以被 [FileSelectFile](#)(See 16.23), [FileSelectFolder](#)(See 16.24) 和 [Run](#)(See 24.5) 使用。例如：

```
FileSelectFile, OutputVar,, ::{645ff040-5081-101b-9f08-00aa002f954e} ; 在回收站
里选择一个文件。
```

```
FileSelectFolder, OutputVar, ::{20d04fe0-3aea-1069-a2d8-08002b30309d} ; 在“我
的电脑”中选择一个文件夹。
```

CLSID(类标识符)	含义/位置	被 Run(See 24.5) 支 持？

::{d20ea4e1-3957-11d2-a40b-0c5020524153}	Administrative Tools(管理工具)	
::{85bb920-42a0-1069-a2e4-08002b30309d}	Briefcase(公文包)	
::{21ec2020-3aea-1069-a2dd-08002b30309d}	Control Panel(控制面板)	
::{d20ea4e1-3957-11d2-a40b-0c5020524152}	Fonts(字体)	
::{ff393560-c2a7-11cf-bff4-444553540000}	History(历史)	
::{00020d75-0000-0000-c000-000000000046}	Inbox(收件箱)	
::{00028b00-0000-0000-c000-000000000046}	Microsoft Network(微软网络)	
::{20d04fe0-3aea-1069-a2d8-08002b30309d}	我的电脑	是
::{450d8fba-ad25-11d0-98a8-0800361b1103}	我的文档	是
::{208d2c60-3aea-1069-a2d7-08002b30309d}	网上邻居	是
::{1f4de370-d627-11d1-ba4f-00a0c91eedba}	搜索结果 - 计算机	是
::{7007acc7-3202-11d1-aad2-00805fc1270e}	网络连接	是
::{2227a280-3aea-1069-a2de-08002b30309d}	打印机和传真	是
::{7be9d83c-a729-4d97-b5a7-1b7313c39e0a}	Programs Folder(应用程序文件夹)	
::{645ff040-5081-101b-9f08-00aa002f954e}	回收站	是
::{e211b736-43fd-11d1-9efb-0000f8757fc0}	Scanners and Cameras(扫描仪和照相机)	
::{d6277990-4c6a-11cf-8d87-00aa0060f5bf}	任务计划	是
::{48e7caab-b918-4e58-a94d-505519c795dc}	Start Menu Folder(启动菜单文件夹)	
::{7bd29e00-76c1-11cf-9dd0-00a0c9034933}	Temporary Internet Files(Internet 临时文件)	
::{bdeadf00-c265-11d0-bced-00a0c90ab50f}	Web Folders(网络文件夹)	

最后一列有“是”的项目并不是权威的: [Run](#)(See 24.5) 命令根据系统配置可能支持不同的 CLSIDs。要用 Run 命令打开一个 CLSID 文件夹, 只需将 CLSID 指定为首个参数。例如:

```
Run ::{20d04fe0-3aea-1069-a2d8-08002b30309d} ; 打开“我的电脑”。
```

```
Run ::{645ff040-5081-101b-9f08-00aa002f954e} ; 打开回收站。
```

```
Run ::{450d8fba-ad25-11d0-98a8-0800361b1103}\My Folder ; 打开“我的文档”里的一个文件夹。
```

```
Run %A_MyDocuments%(See 9.)\My Folder ; 在大部分操作系统上等同于上行命令。
```

翻译: [yugi](#) 修正: 天堂之门 menk33@163.com 2008 年 9 月 22 日

30.8 ErrorLevel

ErrorLevel 是为了判断命令运行成功与否而内置的一个变量(不过，并不是所有的命令都会改变 **ErrorLevel** 的值)。当值为 0 时通常代表成功，而其他的时意味着失败。你也可以自己定义 **ErrorLevel** 的值。

要特别注意的就是 [RunWait\(See 24.5\)](#) 命令在它所运行的程序中将 **ErrorLevel** 设置成了退出代码。大部分程序在他们成功完成之后都会产生一个退出代码 0。

每个 [线程 \(See 30.19\)](#)都保留了它自己的 **ErrorLevel** 值，意味着如果 [当前线程\(See 30.19\)](#) 被其他的线程中断后，当原线程恢复时它还保持原有的 **ErrorLevel** 值，而不会被中断线程修改原有的 **ErrorLevel** 值。

备注：由于一些命令将 **ErrorLevel** 的值设置成大于 1，所以最好不要检查 **ErrorLevel** 是否为 1，而是检查它是否为 0 来代替。

例子：

```
WinWait, MyWindow, , 1
if ErrorLevel ; 也就是说它既不是空值，也不是 0.

    MsgBox, The window does not exist.

else

    MsgBox, The window exists.
```

翻译：hsudatalks

30.9 Standard Windows Fonts

字体名称	Win95	WinNT	Win98	Win2000	WinMe	WinXP
Abadi MT Condensed Light			x			
Arial	x	x	x	x	x	x
Arial Alternative Regular					x	
Arial Alternative Symbol					x	
Arial Black			x	x	x	x
Arial Bold	x	x	x	x	x	x
Arial Bold Italic	x	x	x	x	x	x
Arial Italic	x	x	x	x	x	x
Book Antiqua			x			
Calisto MT			x			

Century Gothic			x			
Century Gothic Bold			x			
Century Gothic Bold Italic			x			
Century Gothic Italic			x			
Comic Sans MS			x	x	x	
Comic Sans MS Bold			x	x	x	x
Copperplate Gothic Bold			x			
Copperplate Gothic Light			x			
Courier	x	x	x	x	x	x
Courier New	x	x	x	x	x	x
Courier New Bold	x	x	x	x	x	x
Courier New Bold Italic	x	x	x	x	x	x
Courier New Italic	x	x	x	x	x	x
Estrangelo Edessa						x
Franklin Gothic Medium						x
Franklin Gothic Medium Italic					x	
Gautami						x
Georgia				x		x
Georgia Bold				x		x
Georgia Bold Italic				x		x
Georgia Italic				x		x
Georgia Italic Impact						x
Impact			x	x	x	
Latha						x
Lucida Console		x	x	x	x	x
Lucida Handwriting Italic			x			
Lucida Sans Italic			x			
Lucida Sans Unicode			x	x		x
Marlett			x		x	
Matisse ITC			x			
Modern	x	x	x	x		
Modern MS Sans Serif						x

MS Sans Serif	x	x	x	x	x	x
MS Serif	x	x	x	x	x	
Mv Boli						x
News Gothic MT			x			
News Gothic MT Bold			x			
News Gothic MT Italic			x			
OCR A Extended			x			
Palatino Linotype				x		x
Palatino Linotype Bold				x		x
Palatino Linotype Bold Italic			x		x	
Palatino Linotype Italic				x		x
Roman		x		x		x
Script		x		x		x
Small Fonts		x		x		x
Smallfonts	x		x		x	
Symbol	x	x	x	x	x	x
Tahoma			x	x	x	x
Tahoma Bold			x	x	x	x
Tempus Sans ITC			x	x		
Times New Roman	x	x	x	x	x	x
Times New Roman Bold	x	x	x	x	x	x
Times New Roman Bold Italic	x	x	x	x	x	x
Times New Roman Italic	x	x	x	x	x	x
Trebuchet					x	
Trebuchet Bold					x	
Trebuchet Bold Italic					x	
Trebuchet Italic					x	
Trebuchet MS				x		x
Trebuchet MS Bold				x		x
Trebuchet MS Bold Italic				x		x
Trebuchet MS Italic				x		x
Tunga						x

Verdana (包含于 MSIE 3+ 中)			x	x	x	x
Verdana Bold			x	x	x	x
Verdana Bold Italic			x	x	x	x
Verdana Italic			x	x	x	x
Webdings			x	x	x	x
Westminster			x		x	x
Wingdings	x	x		x		x
WST_Czech						x
WST_Engl						x
WST_Fren						x
WST_Germ						x
WST_Ital						x
WST_Span						x
WST_Swed						x

30.10 Language Codes

下面的列表包含了对应每种语言编码的语言名称，它能被包含在 [A_Language\(See 9.\)](#) 变量里。语言编码本身是下面等号左边的末尾的 4 位数字。例如，如果 A_Language 包含了 0436，那么系统的默认语言就是 Afrikaans。备注：包含字母的编码可以使用大写或小写。

你可以直接把 A_Language 和下面的一个或多个 4 位编码进行比较。例如:`if(A_Language = "0436")`。或者，你可以把整个列表粘贴进一个脚本，然后像列表底部的那个演示一样存取当前语言的名称。

```
languageCode_0436 = Afrikaans
languageCode_041c = Albanian
languageCode_0401 = Arabic_Saudi_Arabia
languageCode_0801 = Arabic_Iraq
languageCode_0c01 = Arabic_Egypt
languageCode_0401 = Arabic_Saudi_Arabia
languageCode_0801 = Arabic_Iraq
languageCode_0c01 = Arabic_Egypt
languageCode_1001 = Arabic.Libya
languageCode_1401 = Arabic_Algeria
languageCode_1801 = Arabic_Morocco
languageCode_1c01 = Arabic_Tunisia
languageCode_2001 = Arabic_Oman
languageCode_2401 = Arabic_Yemen
languageCode_2801 = Arabic_Syria
```

```
languageCode_2c01 = Arabic_Jordan
languageCode_3001 = Arabic_Lebanon
languageCode_3401 = Arabic_Kuwait
languageCode_3801 = Arabic_UAE
languageCode_3c01 = Arabic_Bahrain
languageCode_4001 = Arabic_Qatar
languageCode_042b = Armenian
languageCode_042c = Azeri_Latin
languageCode_082c = Azeri_Cyrillic
languageCode_042d = Basque
languageCode_0423 = Belarusian
languageCode_0402 = Bulgarian
languageCode_0403 = Catalan
languageCode_0404 = Chinese_Taiwan
languageCode_0804 = Chinese_PRC
languageCode_0c04 = Chinese_Hong_Kong
languageCode_1004 = Chinese_Singapore
languageCode_1404 = Chinese_Macau
languageCode_041a = Croatian
languageCode_0405 = Czech
languageCode_0406 = Danish
languageCode_0413 = Dutch_Standard
languageCode_0813 = Dutch_Belgian
languageCode_0409 = English_United_States
languageCode_0809 = English_United_Kingdom
languageCode_0c09 = English_Australian
languageCode_1009 = English_Canadian
languageCode_1409 = English_New_Zealand
languageCode_1809 = English_Irish
languageCode_1c09 = English_South_Africa
languageCode_2009 = English_Jamaica
languageCode_2409 = English_Caribbean
languageCode_2809 = English_Belize
languageCode_2c09 = English_Trinidad
languageCode_3009 = English_Zimbabwe
languageCode_3409 = English_Philippines
languageCode_0425 = Estonian
languageCode_0438 = Faeroese
languageCode_0429 = Farsi
languageCode_040b = Finnish
languageCode_040c = French_Standard
languageCode_080c = French_Belgian
languageCode_0c0c = French_Canadian
languageCode_100c = French_Swiss
languageCode_140c = French_Luxembourg
```

```
languageCode_180c = French_Monaco
languageCode_0437 = Georgian
languageCode_0407 = German_Standard
languageCode_0807 = German_Swiss
languageCode_0c07 = German_Austrian
languageCode_1007 = German_Luxembourg
languageCode_1407 = German_Liechtenstein
languageCode_0408 = Greek
languageCode_040d = Hebrew
languageCode_0439 = Hindi
languageCode_040e = Hungarian
languageCode_040f = Icelandic
languageCode_0421 = Indonesian
languageCode_0410 = Italian_Standard
languageCode_0810 = Italian_Swiss
languageCode_0411 = Japanese
languageCode_043f = Kazakh
languageCode_0457 = Konkani
languageCode_0412 = Korean
languageCode_0426 = Latvian
languageCode_0427 = Lithuanian
languageCode_042f = Macedonian
languageCode_043e = Malay_Malaysia
languageCode_083e = Malay_Brunei_Darussalam
languageCode_044e = Marathi
languageCode_0414 = Norwegian_Bokmal
languageCode_0814 = Norwegian_Nynorsk
languageCode_0415 = Polish
languageCode_0416 = Portuguese_Brazilian
languageCode_0816 = Portuguese_Standard
languageCode_0418 = Romanian
languageCode_0419 = Russian
languageCode_044f = Sanskrit
languageCode_081a = Serbian_Latin
languageCode_0c1a = Serbian_Cyrillic
languageCode_041b = Slovak
languageCode_0424 = Slovenian
languageCode_040a = Spanish_Traditional_Sort
languageCode_080a = Spanish_Mexican
languageCode_0c0a = Spanish_Modern_Sort
languageCode_100a = Spanish_Guatemala
languageCode_140a = Spanish_Costa_Rica
languageCode_180a = Spanish_Panama
languageCode_1c0a = Spanish_Dominican_Republic
languageCode_200a = Spanish_Venezuela
```

```

languageCode_240a = Spanish_Colombia
languageCode_280a = Spanish_Peru
languageCode_2c0a = Spanish_Argentina
languageCode_300a = Spanish_Ecuador
languageCode_340a = Spanish_Chile
languageCode_380a = Spanish_Uruguay
languageCode_3c0a = Spanish_Paraguay
languageCode_400a = Spanish_Bolivia
languageCode_440a = Spanish_El_Salvador
languageCode_480a = Spanish_Honduras
languageCode_4c0a = Spanish_Nicaragua
languageCode_500a = Spanish_Puerto_Rico
languageCode_0441 = Swahili
languageCode_041d = Swedish
languageCode_081d = Swedish_Finland
languageCode_0449 = Tamil
languageCode_0444 = Tatar
languageCode_041e = Thai
languageCode_041f = Turkish
languageCode_0422 = Ukrainian
languageCode_0420 = Urdu
languageCode_0443 = Uzbek_Latin
languageCode_0843 = Uzbek_Cyrillic
languageCode_042a = Vietnamese

```

the_language := languageCode_%A_Language% ; 获取系统的默认语言名称。

MsgBox %the_language% ; 显示语言名称。

翻译: Kookeee 修正: 天堂之门 menk33@163.com 2008 年 9 月 24 日

30.11 Creating a Keyboard Macro or Mouse Macro

宏是按需要“播放”的一系列事先编好的动作。宏最常见的用法是向一个或多个窗口发送 模拟的键击(See 20.12) 和 鼠标点击(See 23.1)。这些窗口对每个键击和鼠标点击做出响应，就好像你手动操作一样，这就使得重复性的工作得以高速可靠地自动化。

虽然宏可以手工编辑，但借助于 AutoScriptWriter，你在编写较长的宏时会感觉更轻松。它是一个附在 AutoHotkey 里的宏录制器。其监视你在何处点击并输入了什么，而且一直记录 激活的(See 28.12) 窗口。它将这些动作转录成一个之后能以更快速度“回放”的可用的宏。

宏回放最便捷的方式之一就是为其指定一个 热键(See 4.) 或者 热字符串(See 5.)。例如，下面的热键将为特定类型的收件人创建一个空白电子邮件，在发送之前允许你进行个性化编写：

```
^!s:: ; Control+Alt+S 热键。
```

```
IfWinNotExist Inbox - Microsoft Outlook
```

```

return ; Outlook 未开启正确的部分，所以什么都不做。

WinActivate ; 激活由上面命令找到的窗口。

Send ^n ; 通过 Control+N 创建一个新的/空白的电子邮件。

WinWaitActive Untitled Message

Send {Tab 2}Product Recall for ACME Rocket Skates ; 设定标题行。

Send {Tab}Dear Sir or Madam,{Enter 2}We have recently discovered a minor
defect ... ; 等等

return ; 这行用来结束热键。

```

上面这样的热键宏对你每天都要多次执行的工作非常有帮助。相比之下，不经常使用的宏可以存放在独立的脚本中，通过把它加到开始菜单或者放在桌面来运行。

想马上创建你自己的宏和热键，请阅读 [迅速开始指南\(See 2.\)](#)。

-- 主页 --

翻译: [Kookeee](#) 修正: 天堂之门 menk33@163.com 2008 年 9 月 23 日

30.12 Overriding Hotkeys

对注册表做如下更改，你可以禁用内置的 Windows 热键除了 Win+L 和 Win+U (这适用于所有的微软操作系统，不过可能需要重启生效):

```

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\Explore
r

NoWinKeys REG_DWORD 0x00000001 (1)

```

但是如果你想进一步设置而非仅仅将它们全部禁用，那么请往下看。注意下面的大部分示例在 Windows Me/98/95 中不支持。

其他程序定义的热键可以在脚本中将它们指定为一个动作来简单地覆写或禁用。此特性常用来改变 Windows 内置的热键。例如，如果你希望 Win+E (启动资源管理器的热键) 执行其他的动作，可以这样:

```

#e:::
MsgBox 现在脚本支配此热键。
return

```

在下面的示例中，用来打开运行对话框的 Win+R 热键被完全禁用了:

```
#r:::return
```

类似地，要禁用两个 Windows 键，可以这样:

```
Lwin:::return
Rwin:::return
```

要禁用或改变一个程序的非全局热键(即只有在应用程序窗口处于激活时才起作用的热键)，可参考下面这个仅在记事本中禁用 **Control+P**(打印)，而让它对所有其它类型的窗口有效的示例：

```
$^p::  
IfWinActive ahk_class Notepad  
    return ;即什么也不做，它使 Control-P 在记事本中失效。  
Send ^p  
return
```

在上例中前缀 \$ 是必需的，以便热键可以“发送自身”而不触发它自己(否则会不停地出现警告框)。同样可见：[上下文相关的热键\(See 4.\)](#)。

你可以复制上述示例到一个新的文本文件例如 "Override.ahk"，然后运行文件来试用。

翻译：yugi 修正：天堂之门 menk33@163.com 2008年10月20日

30.13 Performance

每个脚本在它被载入以及语法检查时都是半编译的。这样做除了会减少脚本消耗的内存外，也会极大地提高运行时的性能。

除了 [SetBatchLines\(See 17.20\)](#)，根据脚本的内容，下列命令同样会影响到性能：[SendMode\(See 20.13\)](#)，[SetKeyDelay\(See 20.14\)](#)，[SetMouseDelay\(See 23.8\)](#)，[SetWinDelay\(See 28.9\)](#)，[SetControlDelay\(See 28.1.13\)](#) 和 [SetDefaultMouseSpeed\(See 23.7\)](#)。

下面是优化过程的技术细节(半编译)：

- 输入输出变量(当不含其它变量的引用时)以及 [group\(See 28.2.2\)](#) 名称被解析为内存地址。
- [Loops\(See 17.12\)](#), [blocks\(See 17.2\)](#), [IFs\(See 17.10\)](#) 和 [ELSEs\(See 17.5\)](#) 会给出它们在脚本内相关的跳转点的内存地址。
- 每个 [Hotkey\(See 4.\)](#), [Gosub\(See 17.8\)](#) 和 [Goto\(See 17.9\)](#) 的终点被解析为一个内存地址，除非它是个变量。
- 在一个跳转列表里，每个命令名称都被替换成一个地址。
- 每一行都被预解析成一个参数列表。
- 每个参数和 [表达式\(See 9.\)](#) 都被预解析为一个 [变量\(See 9.\)](#) 和 [函数\(See 10.\)](#) 的列表(如果有的话)。
- 每个变量或函数的引用都被解析为一个内存地址。

翻译：hong 修正：天堂之门 menk33@163.com 2008年9月24日

30.14 正则表达式 - 快速参考

不论你在哪里: 通常, 正则表达式可以在被搜索字符串中的任意位置找到匹配文本。比如, 正则表达式 **abc** 可以匹配 **abc123**, **123abc**, 和 **123abcxyz**。您也可以使用脱字符号"****"和美元符号"**\$**"把匹配文本锚定(**anchor**)在这一行的开头和结尾。

神奇的转义: (**Hsuda**:如果需要匹配的某个字符本身就是元字符(如: **\.*?+[\{|\()^\\$**), 正则表达式会如何处理呢?)例如搜索 **www.xiaonei.com**, 而需要搜索的文本里的点号在正则表达式里是代表所有字符的元字符。这时我最终使用的正则表达式是 **www\.\.xiaonei\.com**。通过使用反斜线, 让元字符失去特殊含义而成为普通字符。反斜线(\)便是转义符。不过值得注意的是, 在字符组内, 转义字符无效。

忽略大小写: 通常, 正则表达式不会忽略大小写。但是这会给我们带来一些困扰, 比如我们在处理 HTML 文档的时候, **H1** 和 **h1** 是一个意思, 但正则表达式却无法知道。这个时候我们可以使用选项"**i**"让正则表达式忽略大小写。比如 **i)abc** 可以忽略大小写去搜索 **abc**。参照 [options\(See 18.12\)](#) 来获取更多选项信息。

.	点号 匹配单个任意字符(除了新行: 换行符('r)和回车符('n))。比如, ab. 匹配 abc 和 abz 及 ab_ 。
*	星号 可以匹配任意多次(字符, 字符组(class), 单元(subpattern)), 也可能不匹配。比如 a* 可以匹配 ab 也可以匹配 aaaab 。它还可以匹配甚至不包含 a 的字符串。 通配符: AHK 里自由的组合莫过于 点号-星号 的组合。它可以匹配任意多的任意字符, 也可以匹配 0 个字符(当然, 除了新行: 换行符('r)和回车符('n))。比如: abc.*123 匹配 abcAnything123 或者 abc123 。
?	问号 容许匹配一次(字符, 字符组(class), 单元(subpattern)), 但非必须。您可以理解为“之前的那项是可选的”。比如, colou?r 可以 匹配 color 和 colour 因为“u”是可选的。
+	加号 至少需要匹配一次(字符, 字符组(class), 单元(subpattern)), 至多可能任意多次。比如 a+ 匹配 ab 和 aaab 。但是在开头必须至少有一个 a , 这就是他与星号有区别的地方。
{min,max}	区间量词 至少需要 min 次(字符, 字符组(class), 单元(subpattern)), 至多容许 max 次。比如, a{1,2} 可以匹配 ab 但仅仅只能匹配 aaab 中的前两个 a 。 另外, {3} 意味着准确地匹配 3 次, {3,} 意味着 3 次或者更多。注意: 您所指定的数字必须小于 65536, 并且第一个数字要小于第二个数字。(地球人都知道为什么~)
[...]	字符组: (Hsuda :如果我们要搜索的是单词"grey"而又不确定它是否写作"gray", 便应当使用字符组。)在方括号中列出期望匹配的字符。[abc]可以匹配 a 或 b 或 c 。我们也可以使用连字符"-"来表示一个范围; 比如[a-z]就意味着可以匹配所有的小写字母, 很有趣, 不是么? Hsuda :下面所介绍的 \d \w 都是一种特殊的字符组。 与单个字符一样, 您可以使用 *, ?, +, or {min,max}这些量化符号跟在字符组的后面。效果同上面的介绍。比如 [0-9]+ 匹配一个或者更多的数字, 所以能够匹配 xyz123 而不能匹

	<p>配 abxyz。</p> <p>在字符组你既可以罗列你期望匹配的字符也可以使用字符序列，或者将二者组合使用。比如 [a-zA-Z0-9_] 可以匹配数字字母下划线。</p> <p>Hsuda: 我们通常所说的字符组在 POSIX 标准中被称为方括号表达式(bracket expression)。POSIX 中术语"字符组"指的是在方括号表达式内部使用的一种特殊的功能(special feature)，而我们可以认为它们是 Unicode 的字符属性的原型。</p> <p>POSIX 标准中规定的形式是 [:xxx:]，xxx 可以上下面的单词来代替：</p> <ul style="list-style-type: none"> alnum 字母字符和数字字符 alpha 字母 ascii 0-127 blank 空格和制表符 cntrl 控制字符 digit 数字 0-9 xdigit 16 进制数字 print 类似 graph 但包含空白字符 graph 非空白字符(即空白字符，控制字符之外的字符) punct 标点符号 lower 小写字母 upper 大写字母 space 所有的空白字符 word 单词字符，参考 \w <p>Within a character class, characters do not need to be escaped except when they have special meaning inside a class; e.g. [\^a], [a\~b], [a\~], and [\~a].</p>
[^...]	排除型字符组 匹配任何未列出的字符。比如 <code>[\^1-6]</code> 匹配除了 1 到 6 以外的任何字符。其他特性与字符组相同。
\d	匹配单个数字 等价于 <code>[0-9]</code> 。相反的 <code>\D</code> 匹配非数字字符 等价于 <code>[\^0-9]</code> 。这个和下面介绍的那两个都可以用在字符组里。比如 <code>[\d.-]</code> 可以匹配所有数字，点号，和连字符。
\s	匹配空白字符 通常等价于 <code>[\f\n\r\t\v]</code> 即匹配空白符，空格，制表，换行。相反的 <code>\S</code> 匹配非空白字符。
\w	匹配单词中的字符，等价于 <code>[0-9a-zA-Z_]</code> 。相反的 <code>\W</code> 匹配非单词字符 等价于 <code>[\^0-9a-zA-Z_]</code> 。

^ \$	<p>脱字符号"^"和美元符号"\$"被称为锚点是因为他们并不会匹配实际的文本，而是寻找文本中的位置。(Hsuda:我把锚点归类为“零长度断言”)</p> <p>^(脱字符)匹配需要搜索的文本的起始位置。比如：^abc 匹配 abc123 而不能 123abc。</p> <p>\$ 美元符号 匹配目标字符串的末尾。比如 abc\$ 匹配 123abc 而不是 abc123。</p> <p>两个锚点可以组合使用，比如^abc\$ 只能匹配 abc(在 abc 的前面和后面不能有任何字符)。对于有很多行的被搜索文本，我们可以使用 "m" 选项 (See 18.12)使得锚点可以用作搜索每一行而不是把所有行当作整体来搜索。比如 m)^abc\$ 可以匹配 123`r`nabc`r`n789。但是如果缺少 "m" 选项，这是不可以匹配的。</p>
\b	<p>\b 代表"单词分界符(word boundary)"，单词分界符的作用与行锚点一样，也是匹配字符串中的某些位置。它要求当前字符的状态是一个单词字符(\w)而它之前的与它相反。我们使用\b能有效地避免在匹配过程中遇到一个单词属于另一个单词时造成的麻烦。比如：\bcat\b 不能匹配 catfish 而仅仅只能匹配 cat。 \B 则起到的是完全相反的作用。</p>
 	<p>竖线将两个或者更多的选项分隔开来。选项中的任意一项满足条件便可以完成匹配。比如 gray grey 可以匹配 gray 和 grey。相似的，表达式 gr(a e)y 所产生的作用与上面这个例子一样。</p>
(...)	<p>将表达式放入括号里通常应用于：</p> <ul style="list-style-type: none"> 确定赋值的类型。比如(Sun Mon Tues Wednes Thurs Fri Satur)day 可以匹配任何一天的名字。 应用 *, ?, +, 或 {min,max} 来量化多于一个的字符。比如(abc)+ 可以匹配一连串的"abc"，他可以匹配 abcabc 而不能匹配 ab123 或 bc123。 捕获子表达式的值就像捕获 abc(.*)xyz 中点号所匹配的部分。比如 RegExMatch()(See 18.12)在他的输出数组(output array(See 18.12))中存储子表达式所匹配的值。相似的，在 RegExReplace()(See 18.13) 中，允许我们使用类似\$1 这样的回溯(backreferences(See 18.13))将匹配每个子表达式的文本重新插入到结果中。不过捕获并不总是好事，为了避免捕获带来的烦恼，正则表达式提供了非捕获型括号，在左边的括号的后面写上?: 作为前两个字符，比如(?:.*)(?:abc)。 在工作状态下改变选项(options(See 18.12))。比如 (?im) 能够让表达式的剩余部分忽略大小写选项和多行选项(如果它在一个子表达式中，则改变这个子表达式的)。相反的， (?-im) 将把这两个参数全部关闭。除了 DPS`r`n`a 外支持所有的选项。
\t \r etc.	<p>这些转义字符代表特殊字符。最常见的是\t 代表制表符，\r 代表回车，\n 代表换行。在 AutoHotkey 里，这里例子里的反斜线可以用重音符号" ` "来代替(当然是为了方便群众)。像\xhh 这种形式的转义字符也是支持的，hh 则是代表 ANSI 字符的十六进制数(00-FF)。</p> <p>在 v1.0.46.06 以上的版本中，\R 代表"任何形式的单个换行符"，与之功能相同的是 `a option(See 18.12)(然而，\R 在字符组(character class)内仅仅是字母"R")。在 In</p>

v1.0.47.05 以上的版本中，在表达式的前面指定(*BSR_ANYCRLF)可以让 \R 匹配 CR LF 和 CRLF。比如: im)(*BSR_ANYCRLF)abc\Rxyz。

匹配优先模式:通常， *， ?， +， 和 {min,max} 是匹配优先量词因为他们都是尽可能多地去匹配字符。

Hsuda:这里插播一个例子，对于匹配优先模式，我们将遇到一个很常见的陷阱。比如，我们要搜索的句子是

They call me "Hsudatacks" or "Freedom".

我希望搜索第一对引号里的内容，可能会产生这样的表达式".+"，可是由于*是匹配优先的量词，我们得到的结果是

"Hsudatacks" or "Freedom"

这明显不是我们所期望的结果。那该怎么办呢？请看下面的帮助：

在他们后面使用问号则可以让他们尽可能少的匹配字符。那上面的例子我们采用".+?", 所得到的便是 "Hsudatacks"。

例：表达式<.+>所表示的意思是先搜索<，在匹配一个或者更多的任意字符，后面跟着一个>。在匹配时会出现 text，为了避免这种情况，我们使用<.+?> 使得表达式仅仅只会匹配标记。

顺序环视和逆序环视: (?=, (?!=...), (?<= 和 (?<!...)) 被成为环视，因为他们都是形容一个匹配条件而不消耗任何字符。比如: abc(?=.*xyz) 匹配右边的某个地方存在字符 xyz 的字符 abc(如果不存在，便不能匹配)。 (?=被称为肯定顺序环视，因为他需要指定的字符在后面存在才能够完成匹配过程。相反的，(?!=...)是否定顺序环视，因为他是在不存在指定字符的条件下的条件下完成匹配。相似的，(?<= 和 (?<!...))被称为肯定和否定的逆序环视，因为他们是在当前位置的左边寻找指定的字符存在或者不存在。逆序要比顺序更加局限，因为他不支持*,?和+这样的量词。

相关: 正则表达式由 [RegExMatch\(\)](#)(See 18.12), [RegExReplace\(\)](#)(See 18.13)和 [SetTitleMatchMode\(\)](#)(See 28.8)支持。

最后: 尽管这个页面介绍了正则表达式中最常用的元字符和一些特性，但是还有比如条件子表达式等相当一部分的其他特性您期望了解的。完整的 PCRE 手册请访问 www.pcre.org/pcre.txt。**(Hsuda:**写到这里，发现 AHK 里的正则表达式原来用的是 PCRE 的库，关于 PCRE，大家需要了解的是这是个地道地道的传统 NFA 引擎的正则库，所以，在编写正则表达式的时候是很考验您的优化功力的)

翻译: hsudatacks 已校对。

30.15 Remapping a Joystick to Keyboard or Mouse

- [重要事项](#)
- [让游戏杆按钮发送键击或者鼠标点击](#)
- [让游戏杆轴向\(Joystick Axis\)或者视点帽\(POV Hat\)发送键击或者鼠标点击](#)
- [备注](#)

- 虽然可以把游戏杆按钮或轴向重映射为按键或者鼠标按钮，但是不能把它映射为另一个游戏杆按钮或轴向。只能用游戏杆模拟器例如 [PPJoy](#) 来实现。
- AutoHotkey 用 1 至 32 之间一个唯一的数字来定义游戏杆上的每个按钮。要确定这些数字，请用[游戏杆测试脚本](#)(See 30.26)。

以下是三种由易到难的方式。最复杂的方法适用于更广泛的情况（例如需要将按键或者鼠标按钮按住不放的游戏）。

方法 #1: 此方法发送简单的键击和鼠标点击。例如：

```
Joy1::Send(See 20.12) {Left} ;让按钮 #1 发送一个左箭头键击。
Joy2::Click(See 23.1) ;让按钮 #2 发送一个鼠标左键点击。
Joy3::Send a{Esc}{Space}{Enter} ;让按钮 #3 发送字母"a"后紧跟 Escape, Space 和 Enter。
Joy4::Send Sincerely,{Enter}John Smith ;让按钮 #4 发送一个两行的签名。
```

要让一个按钮执行多个命令，只要将第一个命令放在按钮名称之下并且使最后一个命令为 [return](#)(See 17.19)。例如：

```
Joy5::
Run Notepad
WinWait 无标题 - 记事本
WinActivate
Send This is the text that will appear in Notepad.{Enter}
return
```

请看[按键列表](#)(See 7.)以获得完整的按键和鼠标/游戏杆按钮列表。

方法 #2: 此方法是在你按住一个游戏杆按钮的全程，键盘按键或者鼠标按钮也必须一直按住的情况下才使用的。下面的例子使游戏杆的第二个按钮成为左箭头按键：

```
Joy2::
Send {Left down} ;按住左箭头按键。
KeyWait(See 20.10) Joy2 ;等待用户松开游戏杆按钮。
Send {Left up} ;释放左箭头按键。
return
```

方法 #3: 此方法是你有多个在方法 #2 中所描述的游戏杆热键，并且有时会同时按下或释放这些热键的情况下才使用。下面的例子使游戏杆的第三个按钮成为鼠标左键：

```
Joy3::
Send {LButton down} ;按住鼠标左键。
SetTimer, WaitForButtonUp3, 10
```

```

return

WaitForButtonUp3:
if GetKeyState(See 10.)("Joy3") ;按钮仍处于按下状态的话就继续等待。
return
;否则，按键已松开。
Send {LButton up} ;松开鼠标左键。
SetTimer, WaitForButtonUp3, off
return

```

自动重复某个键击：有些程序或者游戏可能需要你重复地发送某个按键（就像你在键盘上按着它不放）。下面的例子在你按住游戏杆的第二个按钮时通过重复地发送空格键键击来达到此目的。

```

Joy2::
Send {Space down} ;按下空格键。
SetTimer, WaitForJoy2, 30 ;将数字 30 减为 20 或 10 可以更快地发送按键。增加它会发送地更慢。
return

WaitForJoy2:
if not GetKeyState("Joy2") ;按钮已经释放。
{
Send {Space up} ;释放空格键。
SetTimer, WaitForJoy2, off ;停止监视按钮。
return
}
;因为上面没有“return”，所以按钮依然处于按住状态。
Send {Space down} ;发送另一个空格键键击。
return

```

环境敏感的游戏杆按钮：#IfWinActive/Exist(See 20.1.4) 指令使选择的游戏杆按钮根据激活或存在的窗口类型而执行不同的动作(或者根本不执行)。

把游戏杆当鼠标来用：Joystick-To-Mouse 脚本(See 30.25)通过重映射游戏杆的按钮和控制轴向把它变成鼠标。

要让脚本对游戏杆轴向或者视点帽的移动做出响应，请用 SetTimer(See 17.21) 和 GetKeyState(See 20.7)。下面的例子将使游戏杆的 X 和 Y 轴表现得和键盘上的箭头按键群（左、右、上、下）一样。

```

#Persistent ;保持脚本运行，直到用户明确地退出。
SetTimer(See 17.21), WatchAxis, 5
return

WatchAxis:

```

```

GetKeyState(See 20.7), JoyX, JoyX ;得到 x 轴的坐标。
GetKeyState, JoyY, JoyY ;得到 y 轴的坐标。
KeyToHoldDownPrev = %KeyToHoldDown% ; Prev 现在保存了先前按下的按键（如果有的话）。

if JoyX > 70
    KeyToHoldDown = Right
else if JoyX < 30
    KeyToHoldDown = Left
else if JoyY > 70
    KeyToHoldDown = Down
else if JoyY < 30
    KeyToHoldDown = Up
else
    KeyToHoldDown =

if KeyToHoldDown = %KeyToHoldDownPrev% ;已经按下了正确的按键（或不需要按键）。
    return ;什么也不做。

;否则，松开先前的按键并按下一个新键：
SetKeyDelay -1 ;避免击键之间的延时。
if KeyToHoldDownPrev ;先前的按键需要松开。
    Send, {%KeyToHoldDownPrev% up} ;松开它。
if KeyToHoldDown ;要按下一个键。
    Send, {%KeyToHoldDown% down} ;按下它。
return

```

下面的例子将让游戏杆的视点帽起到和键盘上的箭头按键群一样的作用；就是说，视点帽将发送方向键击（左、右、上、下）：

```

#Persistent ;保持脚本运行，直到用户明确地退出。
SetTimer, WatchPOV, 5
return

WatchPOV:
GetKeyState, POV, JoyPOV ;获得视点控制的坐标。
KeyToHoldDownPrev = %KeyToHoldDown% ; Prev 现在存储了先前按下的键（如果有的话）。

;有一些游戏杆会有平滑/连续的视角切换而不是固定的增幅。
;要全部支持它们，可用一个值域：
if POV < 0 ;没有角度
    KeyToHoldDown =
else if POV > 31500 ; 315 至 360 度：向上
    KeyToHoldDown = Up

```

```

else if POV between 0 and 4500 ; 0 至 45 度: 向上
    KeyToHoldDown = Up
else if POV between 4501 and 13500 ; 45 至 135 度: 向右
    KeyToHoldDown = Right
else if POV between 13501 and 22500 ; 135 至 225 度: 向下
    KeyToHoldDown = Down
else ; 225 至 315 度: 向左
    KeyToHoldDown = Left

if KeyToHoldDown = %KeyToHoldDownPrev% ;已经按下了正确的键(或者不需要任何键)。
    return ;什么也不做。

;否则, 释放先前的键并且按下一个新键:
SetKeyDelay -1 ;避免击键之间的延时。
if KeyToHoldDownPrev ;先前的按键需要释放。
    Send, {%KeyToHoldDownPrev% up} ;释放它。
if KeyToHoldDown ;需要按下一个键。
    Send, {%KeyToHoldDown% down} ;按下它。
return

```

自动重复某个键击: 以上两个例子都可以修改为重复地发送按键, 而不仅仅是按住它 (也就是说, 它们可以模拟按住了键盘上的某键)。要做到这一点, 只需将下面这行:

```
return ;什么也不做。
```

替换为:

```
{
if KeyToHoldDown
    Send, {%KeyToHoldDown% down} ;自动地重复键击。
return
}
```

也可以使用除了第一个之外的游戏杆, 只要通过在按钮或者轴名之前加上游戏杆编号。例如, `2joy1` 就是指二号杆的第一个按钮。

要获得更多有用的游戏杆脚本, 请访问 [AutoHotkey 论坛](#)。搜索关键词例如 `Joystick` 和 `GetKeyState` 和 `Send` 有可能找到你想要的主题。

[Joystick-To-Mouse 脚本 \(将游戏杆当鼠标用\) \(See 30.25\)](#)

[游戏杆按钮、轴向以及控制器的列表\(See 7.\)](#)

[GetKeyState\(See 20.7\)](#)[重映射键盘和鼠标\(See 6.\)](#)

翻译: handt 修正: 天堂之门 menk33@163.com 2008 年 10 月 20 日

30.16 PostMessage/SendMessage Tutorial

本页讨论 [PostMessage\(See 28.1.12\)](#) 和 [SendMessage\(See 28.1.12\)](#) 命令, 并将回答这样一些问题:

“我如何按下已被最小化的窗口中的按钮？”

“当 [WinMenuItem\(See 28.1.14\)](#) 不起作用时, 我如何选中一个菜单项? !”

“这是个可更换皮肤的窗口……如何保证我发送的命令每次都能生效?”

“**隐藏**窗口的话该怎么办? !”

环境要求: AutoHotkey v1.0.09+ 以及 Winspector Spy (www.windows-spy.com)

例如第一个例子中, 注意 [WinMenuItem\(See 28.1.14\)](#) 命令就对 Outlook Express 的"New Message"窗口上的菜单栏无效。换言之, 该代码不起任何作用:

WinMenuItem, New Message,, &Insert, &Picture...

但 [PostMessage\(See 28.1.12\)](#) 命令就能完成这个工作:

PostMessage, 0x111, 40239, 0, , New Message

好神奇啊! 但那是啥鬼东西? 0x111 是 [wm_command 消息\(See 30.17\)](#) 的十六进制代码, 40239 是让这个特殊的窗口理解为选择 'Insert Picture' 菜单项的代码。现在让我来告诉你如何获得一个类似 40239 这样的值:

1. 启动 Winspector Spy 软件, 打开 "New Message" 窗口。
2. 拖动 Winspector Spy 窗口中的十字光标到 "New Message" 窗口的标题栏上(未被 Winspector Spy 窗口的覆盖效果遮蔽的部分)。
3. 在左边列表中被选中的窗口名称上点击右键, 选择 'Messages'。
4. 右键点击空白窗口并选择 'Edit message filter'。
5. 按 'filter all' 按钮, 然后双击左边列表中的 'wm_command' 项。这样你将只监视此消息。
6. 现在转到 "New Message" 窗口并从菜单栏选择: **Insert > Picture**。
7. 返回 Winspector Spy 并按下信号灯按钮来暂停监视。
8. 展开收集到的 [wm_command](#) 消息(忽略其它消息)。
9. 你想要找的(通常)是一个代码为 0 的消息。有时那里描述为 'win activated' 或者 'win destroyed' 以及其它等等……都是不需要的内容。你要么发现那有一个描述为 'Control ID: 40239' 的消息……那就是它了!
10. 现在将它放入上面的命令中你就搞定了! 它就是 wParam 参数的值。

在下一个例子中，我会用到画图，因为大概每个人都会有。现在，让我们假设你要使用 AutoHotkey 从一个程序的工具栏选择一个工具；假设选择的就是取色这个工具。

你会怎么做？很可能是在工具栏按钮上用鼠标点击，对吧？但是工具栏可以被移动或者隐藏！在画图中也可以。所以如果目标用户移动或隐藏了工具栏，那么你的脚本在那个点上将会失效。但是下面的命令仍将有效：

`PostMessage, 0x111, 639,,, 未命名 - 画图`

[PostMessage\(See 28.1.12\)](#) 的另一个好处是窗口可以处在后台；相比之下，发送鼠标击键需要窗口处于激活状态。

下面有更多的例子。注意：我用的是 WinXP Pro (SP1).....如果你使用的是不同的操作系统，那可能需要调整参数(仅对于像写字板和记事本这种 windows 自带的应用程序；其它的程序而言参数应该不用改变)。

;写字板字体设为青色

`PostMessage(See 28.1.12), 0x111, 32788, 0, , 文档 - 写字板`

;在记事本打开关于对话框

`PostMessage(See 28.1.12), 0x111, 65, 0, , 无标题 - 记事本`

;在记事本切换自动换行

`PostMessage(See 28.1.12), 0x111, 32, 0, , 无标题 - 记事本`

;播放/暂停 Windows Media Player

`PostMessage(See 28.1.12), 0x111, 32808, 0, , Windows Media Player`

;挂起一个正在运行的 AHK 脚本的热键

`DetectHiddenWindows, on`

`SetTitleMatchMode, 2`

`PostMessage, 0x111, 65305,,, MyScript.ahk - AutoHotkey ;对比挂起操作，使用 65306 来暂停脚本。`

上文讲的是 PostMessage 命令。[SendMessage\(See 28.1.12\)](#) 命令与之相同，此外还等待一个消息返回值。可以用作例如获取 Winamp 中当前播放的曲目(请看 [Automating Winamp\(See 30.20\)](#) 里的例子)。

这里还有些注意事项：

- 上文也提到了操作系统是 XP 并且要谨记消息值随着不同的操作系统而改变。如果你发现一个消息可以在你的系统上使用(对一个软件的某个版本)，那么它肯定也能用同样版本的软件在其它系统上使用。此外，大多数应用程序即使在不同的版本中也保持相同的消息值(例如 Windows Media Player 和 Winamp)。
- 如果你已在 Winspector Spy 设置了过滤仅显示 `wm_command` 消息，而你仍然获得大量的消息，这时，在那消息上点右键选择 `hide` (消息名称).....你不会想要看到那个不是你与目标软件交互而出现的消息。

- 在 Winspector Spy 中向右的箭头显示消息值，模糊的向左的箭头显示返回值。返回值为 0，可默认安全地视为 'no error'(使用 SendMessage 命令，返回值将会在 %ErrorLevel%(See 30.8) 中)。
- 要给部分标题匹配的窗口传递消息，在脚本里加上这行：
`SetTitleMatchMode, 2`
- 要给隐藏窗口传递消息，在脚本里加上这行：
`DetectHiddenWindows, On`

注意：此技术对某些应用程序无效。我对 VB 和 Delphi 编写的应用程序用时只能侥幸。此技术最适于 C, C++ 编写的应用程序。对于 VB 应用程序，同个命令的 'LParam' 参数在传递时总是变化。对于 Delphi 应用程序.....一些程序的 GUI (用户图形界面)甚至不使用 `wm_command` 。它大概使用了鼠标位置和点击。

去探索吧.....要在 AutoHotkey 论坛分享你的经验哦。欢迎您反馈信息！

这个指南并不是为新手们准备的(没有冒犯的意思)，因为这些命令被认为是高级专题。所以如果读完本文，你还是摸不着头脑的话，请忘了它吧。

-Rajat

翻译：lwjeee 修正：天堂之门 menk33@163.com 2008 年 8 月 24 日

30.17 List of Windows Messages

下文列举了 [PostMessage\(See 28.1.12\)](#) 和 [SendMessage\(See 28.1.12\)](#) 命令中 *Msg* 参数的值。更多关于特定消息(例如 WM_VSCROLL)的使用，请参考 <http://msdn.microsoft.com> 或使用你喜欢的搜索引擎。同样，查阅 [Message Tutorial\(See 30.16\)](#) (消息指南)。

```
WM_NULL = 0x00
WM_CREATE = 0x01
WM_DESTROY = 0x02
WM_MOVE = 0x03
WM_SIZE = 0x05
WM_ACTIVATE = 0x06
WM_SETFOCUS = 0x07
WM_KILLFOCUS = 0x08
WM_ENABLE = 0x0A
WM_SETREDRAW = 0x0B
WM_SETTEXT = 0x0C
WM_GETTEXT = 0x0D
WM_GETTEXTLENGTH = 0x0E
```

WM_PAINT = 0x0F
WM_CLOSE = 0x10
WM_QUERYENDSESSION = 0x11
WM_QUIT = 0x12
WM_QUERYOPEN = 0x13
WM_ERASEBKGND = 0x14
WM_SYSCOLORCHANGE = 0x15
WM_ENDSESSION = 0x16
WM_SYSTEMERROR = 0x17
WM_SHOWWINDOW = 0x18
WM_CTLCOLOR = 0x19
WM_WININICHANGE = 0x1A
WM_SETTINGCHANGE = 0x1A
WM_DEVMODECHANGE = 0x1B
WM_ACTIVATEAPP = 0x1C
WM_FONTCHANGE = 0x1D
WM_TIMECHANGE = 0x1E
WM_CANCELMODE = 0x1F
WM_SETCURSOR = 0x20
WM_MOUSEACTIVATE = 0x21
WM_CHILDACTIVATE = 0x22
WM_QUEUESYNC = 0x23
WM_GETMINMAXINFO = 0x24
WM_PAINTICON = 0x26
WM_ICONERASEBKGND = 0x27
WM_NEXTDLGCTL = 0x28
WM_SPOOLERSTATUS = 0x2A
WM_DRAWITEM = 0x2B
WM_MEASUREITEM = 0x2C

WM_DELETEITEM = 0x2D
WM_VKEYTOITEM = 0x2E
WM_CHARTOITEM = 0x2F

WM_SETFONT = 0x30
WM_GETFONT = 0x31
WM_SETHOTKEY = 0x32
WM_GETHOTKEY = 0x33
WM_QUERYDRAGICON = 0x37
WM_COMPAREITEM = 0x39
WM_COMPACTING = 0x41
WM_WINDOWPOSCHANGING = 0x46
WM_WINDOWPOSCHANGED = 0x47
WM_POWER = 0x48
WM_COPYDATA = 0x4A
WM_CANCELJOURNAL = 0x4B
WM_NOTIFY = 0x4E
WM_INPUTLANGCHANGEREQUEST = 0x50
WM_INPUTLANGCHANGE = 0x51
WM_TCARD = 0x52
WM_HELP = 0x53
WM_USERCHANGED = 0x54
WM_NOTIFYFORMAT = 0x55
WM_CONTEXTMENU = 0x7B
WM_STYLECHANGING = 0x7C
WM_STYLECHANGED = 0x7D
WM_DISPLAYCHANGE = 0x7E
WM_GETICON = 0x7F
WM_SETICON = 0x80

```
WM_NCCREATE = 0x81  
WM_NCDESTROY = 0x82  
WM_NCCALCSIZE = 0x83  
WM_NCHITTEST = 0x84  
WM_NCPAINT = 0x85  
WM_NCACTIVATE = 0x86  
WM_GETDLGCODE = 0x87  
WM_NCMOUSEMOVE = 0xA0  
WM_NCLBUTTONDOWN = 0xA1  
WM_NCLBUTTONUP = 0xA2  
WM_NCLBUTTONDOWNDBLCLK = 0xA3  
WM_NCRBUTTONDOWN = 0xA4  
WM_NCRBUTTONUP = 0xA5  
WM_NCRBUTTONDOWNDBLCLK = 0xA6  
WM_NCMBUTTONDOWN = 0xA7  
WM_NCMBUTTONUP = 0xA8  
WM_NCMBUTTONDOWNDBLCLK = 0xA9  
  
WM_KEYFIRST = 0x100  
WM_KEYDOWN = 0x100  
WM_KEYUP = 0x101  
WM_CHAR = 0x102  
WM_DEADCHAR = 0x103  
WM_SYSKEYDOWN = 0x104  
WM_SYSKEYUP = 0x105  
WM_SYSCHAR = 0x106  
WM_SYSDEADCHAR = 0x107  
WM_KEYLAST = 0x108
```

WM_IME_STARTCOMPOSITION = 0x10D
WM_IME_ENDCOMPOSITION = 0x10E
WM_IME_COMPOSITION = 0x10F
WM_IME_KEYLAST = 0x10F

WM_INITDIALOG = 0x110
WM_COMMAND = 0x111
WM_SYSCOMMAND = 0x112
WM_TIMER = 0x113
WM_HSCROLL = 0x114
WM_VSCROLL = 0x115
WM_INITMENU = 0x116
WM_INITMENUPOPUP = 0x117
WM_MENUSELECT = 0x11F
WM_MENUCHAR = 0x120
WM_ENTERIDLE = 0x121

WM_CTLCOLORMSGBOX = 0x132
WM_CTLCOLOREDIT = 0x133
WM_CTLCOLORLISTBOX = 0x134
WM_CTLCOLORBTN = 0x135
WM_CTLCOLORDLG = 0x136
WM_CTLCOLORSCROLLBAR = 0x137
WM_CTLCOLORSTATIC = 0x138

WM_MOUSEFIRST = 0x200
WM_MOUSEMOVE = 0x200
WM_LBUTTONDOWN = 0x201

WM_LBUTTONDOWN = 0x201
WM_LBUTTONUP = 0x202
WM_LBUTTONDBLCLK = 0x203
WM_RBUTTONDOWN = 0x204
WM_RBUTTONUP = 0x205
WM_RBUTTONDBLCLK = 0x206
WM_MBUTTONDOWN = 0x207
WM_MBUTTONUP = 0x208
WM_MBUTTONDBLCLK = 0x209
WM_MOUSEWHEEL = 0x20A
WM_MOUSEHWHEEL = 0x20E

WM_PARENTNOTIFY = 0x210
WM_ENTERMENULOOP = 0x211
WM_EXITMENULOOP = 0x212
WM_NEXTMENU = 0x213
WM_SIZING = 0x214
WM_CAPTURECHANGED = 0x215
WM_MOVING = 0x216
WM_POWERBROADCAST = 0x218
WM_DEVICECHANGE = 0x219

WM_MDICREATE = 0x220
WM_MDIDESTROY = 0x221
WM_MDIACTIVATE = 0x222
WM_MDIRESTORE = 0x223
WM_MDINEXT = 0x224
WM_MDIMAXIMIZE = 0x225
WM_MDTILE = 0x226
WM_MDICASCADE = 0x227

WM_MDIICONARRANGE = 0x228

WM_MDIGETACTIVE = 0x229

WM_MDISETMENU = 0x230

WM_ENTERSIZEMOVE = 0x231

WM_EXITSIZEMOVE = 0x232

WM_DROPFILES = 0x233

WM_MDIREFRESHMENU = 0x234

WM_IME_SETCONTEXT = 0x281

WM_IME_NOTIFY = 0x282

WM_IME_CONTROL = 0x283

WM_IME_COMPOSITIONFULL = 0x284

WM_IME_SELECT = 0x285

WM_IME_CHAR = 0x286

WM_IME_KEYDOWN = 0x290

WM_IME_KEYUP = 0x291

WM_MOUSEHOVER = 0x2A1

WM_NCMOUSELEAVE = 0x2A2

WM_MOUSELEAVE = 0x2A3

WM_CUT = 0x300

WM_COPY = 0x301

WM_PASTE = 0x302

WM_CLEAR = 0x303

WM_UNDO = 0x304

WM_RENDERFORMAT = 0x305

WM_RENDERALLFORMATS = 0x306

WM_DESTROYCLIPBOARD = 0x307
WM_DRAWCLIPBOARD = 0x308
WM_PAINTCLIPBOARD = 0x309
WM_VSCROLLCLIPBOARD = 0x30A
WM_SIZECLIPBOARD = 0x30B
WM_ASKCBFORMATNAME = 0x30C
WM_CHANGEBCBCHAIN = 0x30D
WM_HSCROLLCLIPBOARD = 0x30E
WM_QUERYNEWPALETTE = 0x30F
WM_PALETTEISCHANGING = 0x310
WM_PALETTECHANGED = 0x311

WM_HOTKEY = 0x312
WM_PRINT = 0x317
WM_PRINTCLIENT = 0x318

WM_HANDHELDFIRST = 0x358
WM_HANDHELDLAST = 0x35F
WM_PENWINFIRST = 0x380
WM_PENWINLAST = 0x38F
WM_COALESCE_FIRST = 0x390
WM_COALESCE_LAST = 0x39F
WM_DDE_FIRST = 0x3E0
WM_DDE_INITIATE = 0x3E0
WM_DDE_TERMINATE = 0x3E1
WM_DDE_ADVISE = 0x3E2
WM_DDE_UNADVISE = 0x3E3
WM_DDE_ACK = 0x3E4
WM_DDE_DATA = 0x3E5

```

WM_DDE_REQUEST = 0x3E6
WM_DDE_POKE = 0x3E7
WM_DDE_EXECUTE = 0x3E8
WM_DDE_LAST = 0x3E8

WM_USER = 0x400
WM_APP = 0x8000

```

30.18 GUI Styles

Table of Contents

- [Styles Common to Gui/Parent Windows and Most Control Types](#)
- [Text | Edit | UpDown | Picture](#)
- [Button | Checkbox | Radio | GroupBox](#)
- [DropDownList | ComboBox](#)
- [ListBox | ListView | TreeView](#)
- [DateTime | MonthCal](#)
- [Slider | Progress | Tab | StatusBar](#)

Common styles	Value	Description
default for GUI window		WS_POPUP, WS_CAPTION, WS_SYSMENU, WS_MINIMIZEBOX
forced for GUI window		WS_CLIPSIBLINGS
WS_BORDER	0x800000	+/-Border. Creates a window that has a thin-line border.
WS_POPUP	0x80000000	Creates a pop-up window. This style cannot be used with the WS_CHILD style.
WS_CAPTION	0xC00000	+/-Caption. Creates a window that has a title bar. This style is a numerical combination of WS_BORDER and WS_DLGFRA ME.
WS_DISABLED	0x80000000	+/-Disabled. Creates a window that is initially disabled.
WS_DLGFRA ME	0x400000	Creates a window that has a border of a style typically used with dialog boxes.

WS_GROUP	0x20000	+/-Group. Indicates that this control is the first one in a group of controls. This style is automatically applied to manage the "only one at a time" behavior of radio buttons. In the rare case where two groups of radio buttons are added consecutively (with no other control types in between them), this style may be applied manually to the first control of the second radio group, which splits it off from the first.
WS_HSCROLL	0x100000	Creates a window that has a horizontal scroll bar.
WS_MAXIMIZE	0x1000000	Creates a window that is initially maximized.
WS_MAXIMIZEBOX	0x10000	+/-MaximizeBox. Creates a window that has a maximize button. Cannot be combined with the WS_EX_CONTEXTHELP style. The WS_SYSMENU style must also be specified.
WS_MINIMIZE	0x20000000	Creates a window that is initially minimized.
WS_MINIMIZEBOX	0x20000	+/-MinimizeBox. Creates a window that has a minimize button. Cannot be combined with the WS_EX_CONTEXTHELP style. The WS_SYSMENU style must also be specified.
WS_OVERLAPPED	0	Creates an overlapped window. An overlapped window has a title bar and a border. Same as the WS_TILED style.
WS_OVERLAPPEDWINDOW	0xCF0000	Creates an overlapped window with the WS_OVERLAPPED, WS_CAPTION, WS_SYSMENU, WS_THICKFRAME, WS_MINIMIZEBOX, and WS_MAXIMIZEBOX styles. Same as the WS_TILEWINDOW style.
WS_POPUPWINDOW	0x80880000	Creates a pop-up window with WS_BORDER, WS_POPUP, and WS_SYSMENU styles. The WS_CAPTION and WS_POPUPWINDOW styles must be combined to make the window menu visible.
WS_SIZEBOX	0x40000	+/-Resize. Creates a window that has a sizing border. Same as the WS_THICKFRAME style.
WS_SYSMENU	0x80000	+/-SysMenu. Creates a window that has a window menu on its title bar. The WS_CAPTION style must also be specified.
WS_TABSTOP	0x10000	+/-Tabstop. Specifies a control that can receive the keyboard focus when the user presses the TAB key. Pressing the TAB key changes the keyboard focus to the next control with the WS_TABSTOP style.

WS_THICKFRAME	0x40000	Creates a window that has a sizing border. Same as the WS_SIZEBOX style
WS_VSCROLL	0x200000	Creates a window that has a vertical scroll bar.
WS_VISIBLE	0x10000000	Creates a window that is initially visible.
WS_CHILD	0x40000000	Creates a child window. A window with this style cannot have a menu bar. This style cannot be used with the WS_POPUP style.

Text(See 19.4) styles	Value	Description
default		None
forced		None
SS_BLACKFRAME	0x7	Specifies a box with a frame drawn in the same color as the window frames. This color is black in the default color scheme.
SS_BLACKRECT	0x4	Specifies a rectangle filled with the current window frame color. This color is black in the default color scheme.
SS_CENTER	0x1	+/-Center. Specifies a simple rectangle and centers the error value text in the rectangle. The control automatically wraps words that extend past the end of a line to the beginning of the next centered line.
SSETCHEDFRAME	0x12	Draws the frame of the static control using the EDGEETCHED edge style.
SSETCHEDHORZ	0x10	Draws the top and bottom edges of the static control using the EDGEETCHED edge style.
SSETCHEDVERT	0x11	Draws the left and right edges of the static control using the EDGEETCHED edge style.
SS_GRAYFRAME	0x8	Specifies a box with a frame drawn with the same color as the screen background (desktop). This color is gray in the default color scheme
SSGRAYRECT	0x5	Specifies a rectangle filled with the current screen background color. This color is gray in the default color scheme.
SSLEFT	0	+/-Left. This is the default. It specifies a simple rectangle and left-aligns the text in the rectangle. The text is formatted before it is displayed. Words that extend past the end of a line are automatically wrapped to the beginning of the next left-aligned line. Words that are longer than the

		width of the control are truncated.
SS_LEFTNOWORDWRAP	0xC	+/-Wrap. Specifies a rectangle and left-aligns the text in the rectangle. Tabs are expanded, but words are not wrapped. Text that extends past the end of a line is clipped.
SS_NOPREFIX	0x80	Prevents interpretation of any ampersand (&) characters in the control's text as accelerator prefix characters. This can be useful when file names or other strings that might contain an ampersand (&) must be displayed within a text control
SS_NOTIFY	0x100	Sends the parent window the STN_CLICKED notification when the user clicks the control.
SS_RIGHT	0x2	+/-Right. Specifies a rectangle and right-aligns the specified text in the rectangle.
SS_SUNKEN	0x1000	Draws a half-sunken border around a static control.
SS_WHITEFRAME	0x9	Specifies a box with a frame drawn with the same color as the window background. This color is white in the default color scheme.
SS_WHITERECT	0x6	Specifies a rectangle filled with the current window background color. This color is white in the default color scheme.

Edit(See 19.4) styles	Value	Description
default		<p>WS_TABSTOP and WS_EX_CLIENTEDGE (extended style E0x200)</p> <p>If an Edit is auto-detected as multi-line due to its starting contents containing multiple lines, its height being taller than 1 row, or its row-count having been explicitly specified as greater than 1, the following styles will be applied by default:</p> <p>WS_VSCROLL, ES_WANTRETURN, and ES_AUTOVSCROLL</p> <p>If an Edit is auto-detected as a single line, it defaults to having ES_AUTOHSCROLL.</p>
forced		None
ES_AUTOHSCROLL	0x80	+/-Wrap for multi-line edits, and +/-Limit for single-line edits. Automatically scrolls text to the right by 10 characters when the user types a character at the end of the line. When the user

		presses the ENTER key, the control scrolls all text back to the zero position.
ES_AUTOVSCROLL	0x40	Scrolls text up one page when the user presses the ENTER key on the last line.
ES_CENTER	0x1	+/-Center. Centers text in a multiline edit control.
ES_LOWERCASE	0x10	+/-Lowercase. Converts all characters to lowercase as they are typed into the edit control.
ES_NOHIDESEL	0x100	Negates the default behavior for an edit control. The default behavior hides the selection when the control loses the input focus and inverts the selection when the control receives the input focus. If you specify ES_NOHIDESEL, the selected text is inverted, even if the control does not have the focus.
ES_NUMBER	0x2000	+/-Number. Prevents the user from typing anything other than digits in the control.
ES_OEMCONVERT	0x400	This style is most useful for edit controls that contain file names.
ES_MULTILINE	0x4	+/-Multi. Designates a multiline edit control. The default is a single-line edit control.
ES_PASSWORD	0x20	+/-Password. Displays a masking character in place of each character that is typed into the edit control, which conceals the text.
ES_READONLY	0x800	+/-ReadOnly. Prevents the user from typing or editing text in the edit control.
ES_RIGHT	0x2	+/-Right. Right-aligns text in a multiline edit control.
ES_UPPERCASE	0x8	+/-Uppercase. Converts all characters to uppercase as they are typed into the edit control.
ES_WANTRETURN	0x1000	+/-WantReturn. Specifies that a carriage return be inserted when the user presses the ENTER key while typing text into a multiline edit control in a dialog box. If you do not specify this style, pressing the ENTER key has the same effect as pressing the dialog box's default push button. This style has no effect on a single-line edit control.

UpDown(See 19.4) styles	Value	Description
default		UDS_ARROWKEYS, UDS_ALIGNRIGHT, UDS_SETBUDDYINT, and UDS_AUTOBUDDY.
forced		None

UDS_WRAP	0x1	Named option "Wrap". Causes the control to wrap around to the other end of its range when the user attempts to go beyond the minimum or maximum. Without <i>Wrap</i> , the control stops when the minimum or maximum is reached.
UDS_SETBUDDYINT	0x2	Causes the UpDown control to set the text of the buddy control (using the WM_SETTEXT message) when the position changes. However, if the buddy is a ListBox, the ListBox's current selection is changed instead.
UDS_ALIGNRIGHT	0x4	Named option "Right" (default). Positions UpDown on the right side of its buddy control.
UDS_ALIGNLEFT	0x8	Named option "Left". Positions UpDown on the left side of its buddy control.
UDS_AUTOBUDDY	0x10	Automatically selects the previous control in the z-order as the UpDown control's buddy control.
UDS_ARROWKEYS	0x20	Allows the user to press the Up or Down arrow keys on the keyboard to increase or decrease the UpDown control's position.
UDS_HORZ	0x40	Named option "Horz". Causes the control's arrows to point left and right instead of up and down.
UDS_NOTHOUSANDS	0x80	Does not insert a thousands separator between every three decimal digits in the buddy control.
UDS_HOTTRACK	0x100	Causes the control to exhibit "hot tracking" behavior. That is, it highlights the control's buttons as the mouse passes over them. This style requires Microsoft Windows 98 or Windows 2000. If the system is running Windows 95 NT4, the flag is ignored. The flag is also ignored on Windows XP when the desktop theme overrides it.
(hex)		<p>The number format displayed inside the buddy control may be changed from decimal to hexadecimal by following this example:</p> <pre>Gui +LastFound SendMessage, 1133, 16, 0, msctls_updown321 ; 1133 is UDM_SETBASE</pre> <p>However, this affects only the buddy control, not the UpDown's reported position.</p>

Picture(See 19.4)	Value	Description
-------------------	-------	-------------

styles		
default		None
forced		SS_ICON for icons and cursors. SS_BITMAP for other image types.
SS_REALSIZECONTROL	0x40	[Windows XP or later] Adjusts the bitmap to fit the size of the control.
SS_CENTERIMAGE	0x200	Centers the bitmap in the control. If the bitmap is too large, it will be clipped. For text controls If the control contains a single line of text, the text is centered vertically within the available height of the control Microsoft Windows XP: This style bit no longer results in unused portions of the control being filled with the color of the top left pixel of the bitmap or icon. Unused portions of the control will remain the background color.

Button(See 19.4) , Checkbox(See 19.4) , Radio(See 19.4) , and GroupBox(See 19.4) styles	Value	Description
default		BS_MULTILINE (except for GroupBox, and except for buttons, checkboxes, and radios that have no explicitly set width or height, nor any CR/LF characters in their text) WS_TABSTOP (except for GroupBox) -- however, radio buttons other than the first of each radio group lack WS_TABSTOP by default. In addition, radio buttons have BS_NOTIFY so that double clicks can be detected.
forced		Button: BS_PUSHBUTTON or BS_DEFPUSHBUTTON Radio: BS_AUTORADIOBUTTON Checkbox: BS_AUTOCHECKBOX or BS_AUTO3STATE GroupBox: BS_GROUPBOX
BS_LEFT	0x100	+/-Left. Left-aligns the text.
BS_PUSHLIKE	0x1000	Makes a checkbox or radio button look and act like a push button. The button looks raised when it isn't pushed or checked, and sunken when it is pushed or checked.

BS_RIGHT	0x200	+/-Right. Right-aligns the text.
BS_RIGHTBUTTON	0x20	+Right (i.e. +Right includes both BS_RIGHT and BS_RIGHTBUTTON, but -Right removes only BS_RIGHT, not BS_RIGHTBUTTON). Positions a checkbox square or radio button circle on the right side of the control's available width instead of the left.
BS_BOTTOM	0x800	Places the text at the bottom of the control's available height.
BS_CENTER	0x300	+/-Center. Centers the text horizontally within the control's available width.
BS_DEFPUSHBUTTON	0x1	+/-Default. Creates a push button with a heavy black border. If the button is in a dialog box, the user can select the button by pressing the ENTER key, even when the button does not have the input focus. This style is useful for enabling the user to quickly select the most likely option.
BS_MULTILINE	0x2000	+/-Wrap. Wraps the text to multiple lines if the text is too long to fit on a single line in the control's available width. This also allows linefeed (`n) to start new lines of text.
BS_TOP	0x400	Places text at the top of the control's available height.
BS_VCENTER	0xC00	Vertically centers text in the control's available height.
BS_FLAT	0x8000	Specifies that the button is two-dimensional; it does not use the default shading to create a 3-D effect.

DropDownList(See 19.4) and ComboBox(See 19.4) styles	Value	Description
default		WS_TABSTOP (+/-Tabstop) DropDownList: WS_VSCROLL ComboBox: WS_VSCROLL, CBS_AUTOHSCROLL
forced		DropDownList: CBS_DROPDOWNLIST ComboBox: Either CBS_DROPDOWN or CBS_SIMPLE
CBS_AUTOHSCROLL	0x40	+/-Limit. Automatically scrolls the text in an edit control to the right when the user types a character at the end of the line. If this style is not set, only text that fits within the rectangular boundary is enabled.
CBS_DISABLENOSCROLL	0x800	Shows a disabled vertical scroll bar in the list box when the box does not contain enough items to scroll. Without this style, the scroll bar is hidden when the list box does not contain enough items.

CBS LOWERCASE	0x4000	+/-Lowercase. Converts to lowercase any uppercase characters that are typed into the edit control of a combo box.
CBS_NOINTEGRALHEIGHT	0x400	Specifies that the combo box will be exactly the size specified by the application when it created the combo box. Usually, Windows CE sizes a combo box so that it does not display partial items.
CBS_OEMCONVERT	0x80	Converts text typed in the combo box edit control from the Windows CE character set to the OEM character set and then back to the Windows CE set. This style is most useful for combo boxes that contain file names. It applies only to combo boxes created with the CBS_DROPDOWN style.
CBS_SIMPLE	0x1	+/-Simple (ComboBox only). Displays the list box at all times. The current selection in the list box is displayed in the edit control.
CBS_SORT	0x100	+/-Sort. Sorts the items in the drop-list alphabetically.
CBS_UPPERCASE	0x2000	+/-Uppercase. Converts to uppercase any lowercase characters that are typed into the edit control of a ComboBox.

ListBox(See 19.4) styles	Value	Description
default		WS_TABSTOP, LBS_USETABSTOPS, WS_VSCROLL, and WS_EX_CLIENTEDGE (extended style E0x200).
forced		LBS_NOTIFY (supports detection of double-clicks)
LBS_DISABLENOSCROLL	0x1000	Shows a disabled vertical scroll bar for the list box when the box does not contain enough items to scroll. If you do not specify this style, the scroll bar is hidden when the list box does not contain enough items.
LBS_NOINTEGRALHEIGHT	0x100	Specifies that the list box will be exactly the size specified by the application when it created the list box.
LBS_EXTENDEDSEL	0x800	+/-Multi. Allows multiple selections via control-click and shift-click.
LBS_MULTIPLESEL	0x8	A simplified version of multi-select in which control-click and shift-click are not necessary because normal left clicks serve to extend the selection or de-select a selected

		item.
LBS_NOSEL	0x4000	+/-ReadOnly. Specifies that the user can view list box strings but cannot select them.
LBS_SORT	0x2	+/-Sort. Sorts the items in the list box alphabetically.
LBS_USETABSTOPS	0x80	Enables a ListBox to recognize and expand tab characters when drawing its strings. The default tab positions are 32 dialog box units. A dialog box unit is equal to one-fourth of the current dialog box base-width unit.

ListView(See 19.7) styles	Value	Description
default		WS_TABSTOP, LVS_SHOWSELALWAYS, LVS_EX_FULLROWSELECT, LVS_EX_HEADERDRAGDROP, WS_EX_CLIENTEDGE (E0x200)
forced		None
LVS_ALIGNLEFT	0x800	Items are left-aligned in icon and small icon view.
LVS_ALIGNTOP	0	Items are aligned with the top of the list-view control in icon and small icon view. This is the default.
LVS_AUTOARRANGE	0x100	Icons are automatically kept arranged in icon and small icon view.
LVS_EDITLABELS	0x200	+/-ReadOnly. Specifying -ReadOnly (or +0x200) allows the user to edit the first field of each row in place.
LVS_ICON	0	+Icon. Specifies large-icon view.
LVS_LIST	0x3	+List. Specifies list view.
LVS_NOCOLUMNHEADER	0x4000	+/-Hdr. Avoids displaying column headers in report view.
LVS_NOLABELWRAP	0x80	Item text is displayed on a single line in icon view. By default, item text may wrap in icon view.
LVS_NOSCROLL	0x2000	Scrolling is disabled. All items must be within the client area. This style is not compatible with the LVS_LIST or LVS_REPORT styles.
LVS_NOSORTHEADER	0x8000	+/-NoSortHdr. Column headers do not work like buttons. This style can be used if clicking a column header in report view does not carry out

		an action, such as sorting.
LVS_OWNERDATA	0x1000	This style specifies a virtual list-view control (not directly supported by AutoHotkey).
LVS_OWNERDRAWFIXED	0x400	The owner window can paint items in report view in response to WM_DRAWITEM messages (not directly supported by AutoHotkey).
LVS_REPORT	0x1	+Report. Specifies report view.
LVS_SHAREIMAGELISTS	0x40	The image list (See 19.7) will not be deleted when the control is destroyed. This style enables the use of the same image lists with multiple list-view controls.
LVS_SHOWSELALWAYS	0x8	The selection, if any, is always shown, even if the control does not have keyboard focus.
LVS_SINGLESEL	0x4	+/-Multi. Only one item at a time can be selected. By default, multiple items can be selected.
LVS_SMALLICON	0x2	+IconSmall. Specifies small-icon view.
LVS_SORTASCENDING	0x10	+/-Sort. Rows are sorted in ascending order based on the contents of the first field.
LVS_SORTDESCENDING	0x20	+/-SortDesc. Same as above but in descending order.
LVS_EX_BORDERSELECT	LV0x8000	When an item is selected the border color of the item changes rather than the item being highlighted (might be non-functional in recent operating systems).
LVS_EX_CHECKBOXES	LV0x4	<p>+/-Checked. Displays a checkbox with each item. When set to this style, the control creates and sets a state image list with two images using DrawFrameControl. State image 1 is the unchecked box, and state image 2 is the checked box. Setting the state image to zero removes the check box altogether.</p> <p>Windows XP or later: Checkboxes are visible and functional with all list-view modes except the tile view mode introduced in Windows XP. Clicking a checkbox in tile view mode only selects the item; the state does not change.</p>
LVS_EX_FLATSB	LV0x100	Enables flat scroll bars in the list view.

LVS_EX_FULLROWSELECT	LV0x20	When a row is selected, all its fields are highlighted. This style is available only in conjunction with the LVS_REPORT style.
LVS_EX_GRIDLINES	LV0x1	+/-Grid. Displays gridlines around rows and columns. This style is available only in conjunction with the LVS_REPORT style.
LVS_EX_HEADERDRAGDROP	LV0x10	Enables drag-and-drop reordering of columns in a list-view control. This style is only available to list-view controls that use the LVS_REPORT style.
LVS_EX_INFOTIP	LV0x400	When a list-view control uses the LVS_EX_INFOTIP style, the LVN_GETINFOTIP notification message is sent to the parent window before displaying an item's ToolTip.
LVS_EX_LABELTIP	LV0x4000	If a partially hidden label in any list-view mode lacks ToolTip text, the list-view control will unfold the label. If this style is not set, the list-view control will unfold partly hidden labels only for the large icon mode. Requires Windows XP or later, or the DLLs distributed with Internet Explorer 5.0 or later.
LVS_EX_MULTIWORKAREAS	LV0x2000	If the list-view control has the LVS_AUTOARRANGE style, the control will not autoarrange its icons until one or more work areas are defined (see LVM_SETWORKAREAS). To be effective, this style must be set before any work areas are defined and any items have been added to the control.
LVS_EX_ONECLICKACTIVATE	LV0x40	The list-view control sends an LVN_ITEMACTIVATE notification message to the parent window when the user clicks an item. This style also enables hot tracking in the list-view control. Hot tracking means that when the cursor moves over an item, it is highlighted but not selected.
LVS_EX_REGIONAL	LV0x200	Sets the list-view window region to include only the item icons and text using SetWindowRgn. Any area that is not part of an item is excluded from the window region. This style is only available to list-view controls that use the LVS_ICON style.
LVS_EX_SIMPLESELECT	LV0x100000	In icon view, moves the state image of the control to the top right of the large icon rendering. In

		views other than icon view there is no change. When the user changes the state by using the space bar, all selected items cycle over, not the item with the focus. Requires Windows XP or later.
LVS_EX_SUBITEMIMAGES	LV0x2	Allows images to be displayed for fields beyond the first. This style is available only in conjunction with the LVS_REPORT style.
LVS_EX_TRACKSELECT	LV0x8	Enables hot-track selection in a list-view control. Hot track selection means that an item is automatically selected when the cursor remains over the item for a certain period of time. The delay can be changed from the default system setting with a LVM_SETHOVERTIME message. This style applies to all styles of list-view control. You can check whether hot-track selection is enabled by calling SystemParametersInfo.
LVS_EX_TWOCLICKACTIVATE	LV0x80	The list-view control sends an LVN_ITEMACTIVATE notification message to the parent window when the user double-clicks an item. This style also enables hot tracking in the list-view control. Hot tracking means that when the cursor moves over an item, it is highlighted but not selected.
LVS_EX_UNDERLINECOLD	LV0x1000	Causes those non-hot items that may be activated to be displayed with underlined text. This style requires that LVS_EX_TWOCLICKACTIVATE be set also.
LVS_EX_UNDERLINEHOT	LV0x800	Causes those hot items that may be activated to be displayed with underlined text. This style requires that LVS_EX_ONECLICKACTIVATE or LVS_EX_TWOCLICKACTIVATE also be set.

TreeView(See 19.8) styles	Value	Description
default		WS_TABSTOP, TVS_SHOWSELALWAYS, TVS_HASLINES, TVS_LINESATROOT, TVS_HASBUTTONS, WS_EX_CLIENTEDGE (E0x200)
forced		None

TVS_CHECKBOXES	0x100	+/-Checked. Displays a checkbox next to each item.
TVS_DISABLEDRAGDROP	0x10	Prevents the tree-view control from sending TVN_BEGINDRAG notification messages.
TVS_EDITLABELS	0x8	+/-ReadOnly. Allows the user to edit the names of tree-view items.
TVS_FULLROWSELECT	0x1000	Enables full-row selection in the tree view. The entire row of the selected item is highlighted, and clicking anywhere on an item's row causes it to be selected. This style cannot be used in conjunction with the TVS_HASLINES style.
TVS_HASBUTTONS	0x1	+/-Buttons. Displays plus (+) and minus (-) buttons next to parent items. The user clicks the buttons to expand or collapse a parent item's list of child items. To include buttons with items at the root of the tree view, TVS_LINESATROOT must also be specified.
TVS_HASLINES	0x2	+/-Lines. Uses lines to show the hierarchy of items.
TVS_INFOTIP	0x800	Obtains ToolTip information by sending the TVN_GETINFOTIP notification.
TVS_LINESATROOT	0x4	+/-Lines. Uses lines to link items at the root of the tree-view control. This value is ignored if TVS_HASLINES is not also specified.
TVS_NOHSCROLL	0x8000	+/-HScroll. Disables horizontal scrolling in the control. The control will not display any horizontal scroll bars.
TVS_NONEVENHEIGHT	0x4000	Sets the height of the items to an odd height with the TVM_SETITEMHEIGHT message. By default, the height of items must be an even value.
TVS_NOSCROLL	0x2000	Disables both horizontal and vertical scrolling in the control. The control will not display any scroll bars.
TVS_NOTOOLTIPS	0x80	Disables ToolTips.
TVS_RTLREADING	0x40	Causes text to be displayed from right-to-left (RTL). Usually, windows display text left-to-right (LTR).
TVS_SHOWSELALWAYS	0x20	Causes a selected item to remain selected when the tree-view control loses focus.
TVS_SINGLEEXPAND	0x400	Causes the item being selected to expand and the item being unselected to collapse upon selection in the tree-view. If the user holds down the CTRL key while selecting an item, the item being unselected will not be collapsed.
TVS_TRACKSELECT	0x200	Enables hot tracking of the mouse in a tree-view control.

DateTime (See 19.4) styles	Value	Description
default		DTS_SHORTDATECENTURYFORMAT and WS_TABSTOP (+/-Tabstop)
forced		None
DTS_UPDOWN	0x1	Provides an up-down control to the right of the control to modify date-time values, which replaces the of the drop-down month calendar that would otherwise be available.
DTS_SHOWNONE	0x2	Displays a checkbox inside the control that users can uncheck to make the control have no date/time selected. Whenever the control has no date/time, Gui Submit (See 19.3) and GuiControlGet (See 19.6) will retrieve a blank value (empty string).
DTS_SHORTDATEFORMAT	0x0	Displays the date in short format. In some locales, it looks like 6/1/05 or 6/1/2005. On older operating systems, a two-digit year might be displayed. This is why DTS_SHORTDATECENTURYFORMAT is the default and not DTS_SHORTDATEFORMAT.
DTS_LONGDATEFORMAT	0x4	Format option (See 19.4) "LongDate". Displays the date in long format. In some locales, it looks like Wednesday, June 01, 2005.
DTS_SHORTDATECENTURYFORMAT	0xC	Format option (See 19.4) blank/omitted. Displays the date in short format with four-digit year. In some locales, it looks like 6/1/2005. If the system's version of Comctl32.dll is older than 5.8, this style is not supported and DTS_SHORTDATEFORMAT is automatically substituted.
DTS_TIMEFORMAT	0x9	Format option (See 19.4) "Time". Displays only the time, which in some locales looks like 5:31:42 PM.
DTS_APPCANPARSE	0x10	Not yet supported. Allows the owner to parse user input and take necessary action. It enables users to edit within the client area of the control when they press the F2 key. The control sends DTN_USERSTRING notification messages when users are finished.

DTS_RIGHTALIGN	0x20	+/-Right. The calendar will drop down on the right side of the control instead of the left.
(colors inside the drop-down calendar)		<p>The colors of the day numbers inside the drop-down calendar obey that set by the Gui Font(See 19.3) command or the c (Color)(See 19.3) option. To change the colors of other parts of the calendar, follow this example:</p> <pre>Gui +LastFound SendMessage(See 28.1.12), 0x1006, 4, 0xFFAA99, SysDateTimePick321 ; 0x1006 is DTM_SETMCCOLOR. 4 is MCSC_MONTHBK (background color). The color must be specified in BGR vs. RGB format (red and blue components swapped).</pre>

MonthCal(See 19.4) styles	Value	Description
default		None
forced		None
MCS_DAYSTATE	0x1	Makes the control send MCN_GETDAYSTATE notifications to request information about which days should be displayed in bold. [Not yet supported]
MCS_MULTISELECT	0x2	Named option "Multi". Allows the user to select a range of dates rather than being limited to a single date. By default, the maximum range is 366 days, which can be changed by sending the MCM_SETMAXSELCOUNT message to the control. For example: <pre>Gui +LastFound SendMessage(See 28.1.12), 0x1004, 7, 0, SysMonthCal321 ; 7 days. 0x1004 is MCM_SETMAXSELCOUNT.</pre>
MCS_WEEKNUMBERS	0x4	Displays week numbers (1-52) to the left of each row of days. Week 1 is defined as the first week that contains at least four days.
MCS_NOTODAYCIRCLE	0x8	Prevents the circling of today's date within the control.
MCS_NOTODAY	0x10	Prevents the display of today's date at the bottom of the control.
(colors)		The colors of the day numbers inside the calendar obey that set by the Gui Font (See 19.3) command or the c (Color) (See 19.3) option. To change the colors of other parts of the

		calendar, follow this example: Gui +LastFound SendMessage (See 28.1.12), 0x100A, 5, 0xFFAA99, SysMonthCal321 ; 0x100A is MCM_SETCOLOR. 5 is MCSC_TITLETEXT (color of title text). The color must be specified in BGR vs. RGB format (red and blue components swapped).
--	--	--

Slider(See 19.4) styles	Value	Description
default		WS_TABSTOP (+/-Tabstop)
forced		None
TBS_VERT	0x2	+/-Vertical. The control is oriented vertically.
TBS_LEFT / TBS_TOP	0x4	+/-Left. The control displays tick marks at the top of the control (or to its left if TBS_VERT is present).
TBS_BOTH	0x8	+/-Center. The control displays tick marks on both sides of the control. This will be both top and bottom when used with TBS_HORZ or both left and right if used with TBS_VERT.
TBS_AUTOTICKS	0x1	The control has a tick mark for each increment in its range of values. Use +/-TickInterval to have more flexibility.
TBS_ENABLESEL RANGE	0x20	<p>The control displays a selection range only. The tick marks at the starting and ending positions of a selection range are displayed as triangles (instead of vertical dashes), and the selection range is highlighted (highlighting might require that the theme be removed via "Gui -theme").</p> <p>To set the selection range, follow this example, which sets the starting position to 55 and the ending position to 66:</p> <pre>SendMessage(See 28.1.12), 1035, 1, 55, msctls_trackbar321, WinTitle SendMessage, 1036, 1, 66, msctls_trackbar321, WinTitle</pre>
TBS_FIXEDLENGTH	0x40	+/-Thick. Allows the thumb's size to be changed.
TBS_NOTHUMB	0x80	The control does not display the moveable bar.
TBS_NOTICKS	0x10	+/-NoTicks. The control does not display any tick marks.
TBS_TOOLTIPS	0x100	+/-Tooltip. The control supports ToolTips. When a control is created using this style, it automatically creates a default ToolTip control that displays the slider's current position. You

		can change where the ToolTips are displayed by using the TBM_SETTIPSIDE message. Windows 95 and NT4 require Internet Explorer 3.0 or later to support this style.
TBS_REVERSED	0x200	Unfortunately, this style has no effect on the actual behavior of the control, so there is probably no point in using it (instead, use +Invert in the control's options to reverse it). Depending on OS version, this style might require Internet Explorer 5.0 or greater.
TBS_DOWNISLEFT	0x400	Unfortunately, this style has no effect on the actual behavior of the control, so there is probably no point in using it. Depending on OS version, this style might require Internet Explorer 5.01 or greater.

Progress(See 19.4) styles	Value	Description
default		PBS_SMOOTH
forced		None
PBS_SMOOTH	0x1	+/-Smooth. The progress bar displays progress status in a smooth scrolling bar instead of the default segmented bar. When this style is present, the control automatically reverts to the Classic Theme appearance on Windows XP or later. Windows 95 and NT4 require Internet Explorer 3.0 or later to support this option.
PBS_VERTICAL	0x4	+/-Vertical. The progress bar displays progress status vertically, from bottom to top. Windows 95 and NT4 require Internet Explorer 3.0 or later to support this option.
PBS_MARQUEE	0x8	[Requires Windows XP or later] The progress bar moves like a marquee; that is, each change to its position causes the bar to slide further along its available length until it wraps around to the other side. A bar with this style has no defined position. Each attempt to change its position will instead slide the bar by one increment. This style is typically used to indicate an ongoing operation whose completion time is unknown.

Tab(See 19.4) styles	Value	Description
default		WS_TABSTOP and TCS_MULTILINE
forced		WS_CLIPSIBLINGS

		TCS_OWNERDRAWFIXED: forced on or off as required by the control's background color and/or text color.
TCS_SCROLLTOPPOSITE	0x1	Unneeded tabs scroll to the opposite side of the control when a tab is selected.
TCS_BOTTOM	0x2	+/-Bottom. Tabs appear at the bottom of the control instead of the top.
TCS_RIGHT	0x2	Tabs appear vertically on the right side of controls that use the TCS_VERTICAL style.
TCS_MULTISELECT	0x4	Multiple tabs can be selected by holding down CTRL when clicking. This style must be used with the TCS_BUTTONS style.
TCS_FLATBUTTONS	0x8	Selected tabs appear as being indented into the background while other tabs appear as being on the same plane as the background. This style only affects tab controls with the TCS_BUTTONS style.
TCS_FORCEICONLEFT	0x10	Icons are aligned with the left edge of each fixed-width tab. This style can only be used with the TCS_FIXEDWIDTH style.
TCS_FORCELABELLEFT	0x20	Labels are aligned with the left edge of each fixed-width tab; that is, the label is displayed immediately to the right of the icon instead of being centered. This style can only be used with the TCS_FIXEDWIDTH style, and it implies the TCS_FORCEICONLEFT style.
TCS_HOTTRACK	0x40	Items under the pointer are automatically highlighted
TCS_VERTICAL	0x80	+/-Left or +/-Right. Tabs appear at the left side of the control, with tab text displayed vertically. This style is valid only when used with the TCS_MULTILINE style. To make tabs appear on the right side of the control, also use the TCS_RIGHT style. This style will not correctly display the tabs if a custom background color or text color is in effect. To workaround this, specify -Background and/or cDefault in the tab control's options.
TCS_BUTTONS	0x100	+/-Buttons. Tabs appear as buttons, and no border is drawn around the display area.
TCS_SINGLELINE	0	+/-Wrap. Only one row of tabs is displayed. The user

		can scroll to see more tabs, if necessary. This style is the default.
TCS_MULTILINE	0x200	+/-Wrap. Multiple rows of tabs are displayed, if necessary, so all tabs are visible at once.
TCS_RIGHTJUSTIFY	0	<p>This is the default. The width of each tab is increased, if necessary, so that each row of tabs fills the entire width of the tab control.</p> <p>This window style is ignored unless the TCS_MULTILINE style is also specified.</p>
TCS_FIXEDWIDTH	0x400	All tabs are the same width. This style cannot be combined with the TCS_RIGHTJUSTIFY style.
TCS_RAGGEDRIGHT	0x800	Rows of tabs will not be stretched to fill the entire width of the control. This style is the default.
TCS_FOCUSONBUTTONDOWN	0x1000	The tab control receives the input focus when clicked.
TCS_OWNERDRAWFIXED	0x2000	The parent window is responsible for drawing tabs.
TCS_TOOLTIPS	0x4000	The tab control has a tooltip control associated with it.
TCS_FOCUSNEVER	0x8000	The tab control does not receive the input focus when clicked.

Status Bar (See 19.4) styles	Value	Description
default		SBARS_TOOLTIPS and SBARS_SIZEGRIP (the latter only if the window is resizable).
forced		None
SBARS_TOOLTIPS	0x800	<p>Displays a tooltip when the mouse hovers over a part of the status bar that: 1) has too much text to be fully displayed; or 2) has an icon but no text.</p> <p>The text of the ToolTip can be set via:</p> <pre>Gui +LastFound SendMessage, 0x410, 0, "Text to display", msctls_statusbar321</pre> <p>The bold 0 above is the zero-based part number. To use a part other than the first, specify 1 for second, 2 for the third, etc.</p> <p>NOTE: The ToolTip might never appear on certain OS versions.</p>

SBARS_SIZEGRIP	0x100	Includes a sizing grip at the right end of the status bar. A sizing grip is similar to a sizing border; it is a rectangular area that the user can click and drag to resize the parent window.
----------------	-------	--

30.19 Threads

由最近的[热键\(See 4.\)](#)、[定时的子程序\(See 17.21\)](#)、[自定义的菜单项\(See 22.13\)](#) 和 [GUI 事件\(See 19.3\)](#)所调用的执行流被定义为**当前线程**。当前线程可以是它自身子程序或由子程序调用的其它子程序中的执行命令。

尽管 AutoHotkey 没有真正地使用多线程，但却模拟了它的特性：如果启动了第二个线程，例如前一个热键仍在执行时又按了另一个热键，那么**当前线程**将被中断(暂时地停止)以允许新的线程变成**当前的线程**。如果在第二个线程还在运行时又启动了第三个线程，那么第二个和第一个线程都会进入休眠状态，以此类推。

在**当前线程**结束时，最近中断的那个线程将被恢复，以此类推，直到所有线程最终结束。在线程被恢复时，它的设置比如 [ErrorLevel\(See 30.8\)](#) 和 [SendMode \(See 20.13\)](#)将自动恢复到中断前的状态；换句话说，线程不会受到中断的副作用(除了在[激活的窗口\(See 28.12\)](#)中可能有改变)。

注意：[KeyHistory\(See 20.9\)](#) 命令或菜单项显示了有多少线程处于中断状态，[ListHotkeys\(See 20.1.11\)](#) 命令或菜单项显示了哪些热键拥有线程。

一个脚本可以同时产生多个 [MsgBox\(See 19.11\)](#), [InputBox\(See 19.10\)](#), [FileSelectFile\(See 16.23\)](#) 和 [FileSelectFolder\(See 16.24\)](#) 对话框。这是在先前的线程已经显示了一个对话框的情况下通过启用新线程来实现的(通过[热键\(See 4.\)](#)、[定时的子程序\(See 17.21\)](#)、[自定义的菜单项\(See 22.13\)](#)等等)。

默认情况下，如果一个给定的[热键\(See 4.\)](#)或[热字符串\(See 5.\)](#)子程序已经运行，那么它不能再次运行。可用 [#MaxThreadsPerHotkey\(See 20.1.8\)](#) 改变此特性。

比**当前线程**优先级低的线程([热键\(See 4.\)](#)、[定时的子程序\(See 17.21\)](#)、[自定义菜单项\(See 22.13\)](#)等等)不能中断它。在这个时候，那些定时器都不会运行，并且如果用户尝试创建线程(比如按下[热键\(See 4.\)](#)或[GUI 按钮\(See 19.4\)](#))的话，将无任何反应也不会被缓存起来。由于这个原因，通常设计时最好让高优先级的线程尽快结束，或者使用 [Critical\(See 22.7\)](#) 而不是将它们设置成高优先级。

默认优先级是 0。所有线程都使用默认优先级除非被下列方法之一所改变：

- 1) 通过 [SetTimer\(See 17.21\)](#) 给一个定时的子程序指定优先级。
- 2) 通过 [Hotkey\(See 20.1.10\)](#) 命令给一个热键指定优先级。
- 3) 在定义一个[热字符串\(See 5.\)](#)时指定优先级，或者通过 [#Hotstring\(See 20.1.3\)](#) 指令。
- 4) 通过 [Menu\(See 22.13\)](#) 命令给一个自定义菜单项指定优先级。
- 5) 通过 [Thread\(See 22.22\)](#) 命令来设置**当前线程**自身的优先级。

不管**当前线程**的优先级高低，[OnExit\(See 17.17\)](#) 子程序(如果有的话)在它被调用时将总能运行起来。

翻译：yugi 修正：天堂之门 menk33@163.com 2008 年 11 月 23 日

30.20 Automating Winamp

这一节主要演示当 Winamp 处于最小化或非激活状态下时如何通过热键控制 Winamp。这些代码已经在 Winamp 2.78c 下测试成功其他发行版应该同样运行得很好。如有错误请到论坛提交改进方法或者联系作者。

这个示例使得 **Ctrl+Alt+P** 热键等价于按了 Winamp 的暂停/开始按钮：

```
^!p::  
IfWinNotExist ahk_class Winamp v1.x  
    return  
; 否则，下面将使用最近使用到的窗口。  
ControlSend(See 20.6), ahk_parent, c ; 暂停/开始  
return
```

下面是 Winamp 2.x (可能其他的版本中也可工作) 中的一些快捷键。上面的示例可以使用下面这些快捷键：

要发送的键	效果
c	暂停/开始
x	播放/重放/开始
v	停止
+v	渐隐停止
^v	播放该曲后停止
b	下一曲
z	上一曲
{left}	后退五秒钟
{right}	前进五秒钟
{up}	增大音量
{down}	减小音量

```
; 如下示例获取 Winamp 当前曲目序数：  
SendMessage(See 28.1.12), 1024, 0, 120, ahk_class Winamp v1.x  
if ErrorLevel <> FAIL
```

```
{
    ErrorLevel += 1 ; Winamp 从 0 开始计数, 所以调整为 1.

    MsgBox, Track #%ErrorLevel% is active or playing.

}
```

翻译: xiaohui 2008 年 9 月 23 日

30.21 Context Sensitive Help in Any Editor -- by Rajat

This script makes Ctrl+2 (or another hotkey of your choice) show the help file page for the selected AutoHotkey command or keyword. If nothing is selected, the command name will be extracted from the beginning of the current line.

[Download This Script](#) | [Other Sample Scripts](#)(See 13.) | [Home](#)

```
; The hotkey below uses the clipboard to provide compatibility with the maximum
; number of editors (since ControlGet doesn't work with most advanced editors).

; It restores the original clipboard contents afterward, but as plain text,
; which seems better than nothing.

$^2:::
; The following values are in effect only for the duration of this hotkey thread.
; Therefore, there is no need to change them back to their original values
; because that is done automatically when the thread ends:

SetWinDelay 10
SetKeyDelay 0
AutoTrim, On

if A_OSType = WIN32_WINDOWS ; Windows 9x
    Sleep, 500 ; Give time for the user to release the key.

C_ClipboardPrev = %clipboard%
```

```
clipboard =  
;  
; Use the highlighted word if there is one (since sometimes the user might  
; intentionally highlight something that isn't a command):  
  
Send, ^c  
  
ClipWait, 0.1  
  
if ErrorLevel <> 0  
  
{  
  
    ; Get the entire line because editors treat cursor navigation keys differently:  
  
    Send, {home}+{end}^c  
  
    ClipWait, 0.2  
  
    if ErrorLevel <> 0 ; Rare, so no error is reported.  
  
    {  
  
        clipboard = %C_ClipboardPrev%  
  
        return  
  
    }  
  
}  
  
C_Cmd = %clipboard% ; This will trim leading and trailing tabs & spaces.  
  
clipboard = %C_ClipboardPrev% ; Restore the original clipboard for the user.  
  
Loop, parse, C_Cmd, %A_Space%, ; The first space or comma is the end of the  
command.  
  
{  
  
    C_Cmd = %A_LoopField%  
  
    break ; i.e. we only need one iteration.  
  
}  
  
IfWinNotExist, AutoHotkey Help  
  
{  
  
    ; Determine AutoHotkey's location:  
  
    RegRead, ahk_dir, HKEY_LOCAL_MACHINE, SOFTWARE\AutoHotkey, InstallDir  
  
    if ErrorLevel ; Not found, so look for it in some other common locations.
```

```

{
    if A_AhkPath

        SplitPath, A_AhkPath,, ahk_dir

    else IfExist ..\..\AutoHotkey.chm

        ahk_dir = ..\..

    else IfExist %A_ProgramFiles%\AutoHotkey\AutoHotkey.chm

        ahk_dir = %A_ProgramFiles%\AutoHotkey

    else

    {

        MsgBox Could not find the AutoHotkey folder.

        return

    }

}

Run %ahk_dir%\AutoHotkey.chm

WinWait AutoHotkey Help

}

; The above has set the "last found" window which we use below:

WinActivate

WinWaitActive

StringReplace, C_Cmd, C_Cmd, #, {#}

send, !n{home}+{end}%C_Cmd%{enter}

return

```

30.22 Easy Window Dragging (requires XP/2k/NT)

Normally, a window can only be dragged by clicking on its title bar. This script extends that so that any point inside a window can be dragged. To activate this mode, hold down CapsLock or the middle mouse button while clicking, then drag the window to a new position.

[Download This Script](#) | [Other Sample Scripts](#)(See 13.) | [Home](#)

; Note: You can optionally release Capslock or the middle mouse button after

```
; pressing down the mouse button rather than holding it down the whole time.  
; This script requires v1.0.25+.  
  
~MButton & LButton::  
  
CapsLock & LButton::  
  
CoordMode, Mouse ; Switch to screen/absolute coordinates.  
  
MouseGetPos, EWD_MouseStartX, EWD_MouseStartY, EWD_MouseWin  
  
WinGetPos, EWD_OriginalPosX, EWD_OriginalPosY,, ahk_id %EWD_MouseWin%  
  
WinGet, EWD_WinState, MinMax, ahk_id %EWD_MouseWin%  
  
if EWD_WinState = 0 ; Only if the window isn't maximized  
    SetTimer, EWD_WatchMouse, 10 ; Track the mouse as the user drags it.  
  
return  
  
  
EWD_WatchMouse:  
  
GetKeyState, EWD_LButtonState, LButton, P  
  
if EWD_LButtonState = U ; Button has been released, so drag is complete.  
{  
    SetTimer, EWD_WatchMouse, off  
    return  
}  
  
GetKeyState, EWD_EscapeState, Escape, P  
  
if EWD_EscapeState = D ; Escape has been pressed, so drag is cancelled.  
{  
    SetTimer, EWD_WatchMouse, off  
    WinMove,  
ahk_id %EWD_MouseWin%,,%EWD_OriginalPosX%,,%EWD_OriginalPosY%  
    return  
}  
  
; Otherwise, reposition the window to match the change in mouse coordinates
```

```
; caused by the user having dragged the mouse:

CoordMode, Mouse

MouseGetPos, EWD_MouseX, EWD_MouseY

WinGetPos, EWD_WinX, EWD_WinY,,, ahk_id %EWD_MouseWin%

SetWinDelay, -1 ; Makes the below move faster/smoothier.

WinMove, ahk_id %EWD_MouseWin%,, EWD_WinX + EWD_MouseX - EWD_MouseStartX,
EWD_WinY + EWD_MouseY - EWD_MouseStartY

EWD_MouseStartX := EWD_MouseX ; Update for the next timer-call to this subroutine.

EWD_MouseStartY := EWD_MouseY

return
```

30.23 Easy Access to Favorite Folders -- by Savage

When you click the middle mouse button while certain types of windows are active, this script displays a menu of your favorite folders. Upon selecting a favorite, the script will instantly switch to that folder within the active window. The following window types are supported: 1) Standard file-open or file-save dialogs; 2) Explorer windows; 3) Console (command prompt) windows. The menu can also be optionally shown for unsupported window types, in which case the chosen favorite will be opened as a new Explorer window.

[Download This Script](#) | [Other Sample Scripts](#)(See 13.) | [Home](#)

```
; Note: In Windows Explorer, if "View > Toolbars > Address Bar" is
; not enabled, the menu will not be shown if the hotkey chosen below
; has a tilde. If it does have a tilde, the menu will be shown
; but the favorite will be opened in a new Explorer window rather
; than switching the active Explorer window to that folder.

; CONFIG: CHOOSE YOUR HOTKEY
; If your mouse has more than 3 buttons, you could try using
; XButton1 (the 4th) or XButton2 (the 5th) instead of MButton.
; You could also use a modified mouse button (such as ^MButton) or
; a keyboard hotkey. In the case of MButton, the tilde (~) prefix
```

```
; is used so that MButton's normal functionality is not lost when
; you click in other window types, such as a browser. The presence
; of a tilde tells the script to avoid showing the menu for
; unsupported window types. In other words, if there is no tilde,
; the hotkey will always display the menu; and upon selecting a
; favorite while an unsupported window type is active, a new
; Explorer window will be opened to display the contents of that
; folder.

f_Hotkey = ~MButton

; CONFIG: CHOOSE YOUR FAVORITES
; Update the special commented section below to list your favorite
; folders. Specify the name of the menu item first, followed by a
; semicolon, followed by the name of the actual path of the favorite.
; Use a blank line to create a separator line.

/*
ITEMS IN FAVORITES MENU <-- Do not change this string.

Desktop ; %A/Desktop%
Favorites ; %A/Desktop%\..\Favorites

My Documents ; %A/MyDocuments%

Program Files; %A/ProgramFiles%
*/

; END OF CONFIGURATION SECTION
; Do not make changes below this point unless you want to change
; the basic functionality of the script.
```

```
#SingleInstance ; Needed since the hotkey is dynamically created.

Hotkey, %f_Hotkey%, f_DisplayMenu

StringLeft, f_HotkeyFirstChar, f_Hotkey, 1

if f_HotkeyFirstChar = ~ ; Show menu only for certain window types.

    f_AlwaysShowMenu = n

else

    f_AlwaysShowMenu = y

; Used to reliably determine whether script is compiled:

SplitPath, A_ScriptName,,, f_FileExt

if f_FileExt = Exe ; Read the menu items from an external file.

    f_FavoritesFile = %A_ScriptDir%\Favorites.ini

else ; Read the menu items directly from this script file.

    f_FavoritesFile = %A_ScriptFullPath%

;----Read the configuration file.

f_AtStartingPos = n

f_MenuItemCount = 0

Loop, Read, %f_FavoritesFile%

{

    if f_FileExt <> Exe

    {

        ; Since the menu items are being read directly from this

        ; script, skip over all lines until the starting line is

        ; arrived at.

        if f_AtStartingPos = n

        {
```

```

IfInString, A_LoopReadLine, ITEMS IN FAVORITES MENU

f_AtStartingPos = y

continue ; Start a new loop iteration.

}

; Otherwise, the closing comment symbol marks the end of the list.

if A_LoopReadLine = /*

    break ; terminate the loop

}

; Menu separator lines must also be counted to be compatible

; with A_ThisMenuItemPos:

f_MenuItemCount++

if A_LoopReadLine = ; Blank indicates a separator line.

    Menu, Favorites, Add

else

{

    StringSplit, f_line, A_LoopReadLine, `;

    f_line1 = %f_line1% ; Trim leading and trailing spaces.

    f_line2 = %f_line2% ; Trim leading and trailing spaces.

    ; Resolve any references to variables within either field, and

    ; create a new array element containing the path of this favorite:

    Transform, f_path%f_MenuItemCount%, deref, %f_line2%

    Transform, f_line1, deref, %f_line1%

    Menu, Favorites, Add, %f_line1%, f_OpenFavorite

}

}

return ;----End of auto-execute section.

;----Open the selected favorite

```

```

f_OpenFavorite:

; Fetch the array element that corresponds to the selected menu item:

StringTrimLeft, f_path, f_path%A_ThisMenuItemPos%, 0

if f_path =

    return

if f_class = #32770 ; It's a dialog.

{

    if f_Edit1Pos <> ; And it has an Edit1 control.

    {

        ; Activate the window so that if the user is middle-clicking
        ; outside the dialog, subsequent clicks will also work:

        WinActivate ahk_id %f_window_id%

        ; Retrieve any filename that might already be in the field so
        ; that it can be restored after the switch to the new folder:

        ControlGetText, f_text, Edit1, ahk_id %f_window_id%

        ControlSetText, Edit1, %f_path%, ahk_id %f_window_id%

        ControlSend, Edit1, {Enter}, ahk_id %f_window_id%

        Sleep, 100 ; It needs extra time on some dialogs or in some cases.

        ControlSetText, Edit1, %f_text%, ahk_id %f_window_id%

        return

    }

    ; else fall through to the bottom of the subroutine to take standard action.

}

else if f_class in ExploreWClass,CabinetWClass ; In Explorer, switch folders.

{

    if f_Edit1Pos <> ; And it has an Edit1 control.

    {

        ControlSetText, Edit1, %f_path%, ahk_id %f_window_id%

        ; Tekl reported the following: "If I want to change to Folder L:\folder

```

```
; then the addressbar shows http://www.L:\folder.com. To solve this,
; I added a {right} before {Enter}":

ControlSend, Edit1, {Right}{Enter}, ahk_id %f_window_id%

return

}

; else fall through to the bottom of the subroutine to take standard action.

}

else if f_class = ConsoleWindowClass ; In a console window, CD to that directory

{

    WinActivate, ahk_id %f_window_id% ; Because sometimes the mclick deactivates it.

    SetKeyDelay, 0 ; This will be in effect only for the duration of this thread.

    IfInString, f_path, : ; It contains a drive letter

    {

        StringLeft, f_path_drive, f_path, 1

        Send %f_path_drive%:{enter}

    }

    Send, cd %f_path%{Enter}

    return

}

; Since the above didn't return, one of the following is true:

; 1) It's an unsupported window type but f_AlwaysShowMenu is y (yes).

; 2) It's a supported type but it lacks an Edit1 control to facilitate the custom

;     action, so instead do the default action below.

Run, Explorer %f_path% ; Might work on more systems without double quotes.

return

;

;----Display the menu

f_DisplayMenu:
```

```

; These first few variables are set here and used by f_OpenFavorite:

WinGet, f_window_id, ID, A

WinGetClass, f_class, ahk_id %f_window_id%

if f_class in #32770,ExploreWClass,CabinetWClass ; Dialog or Explorer.

    ControlGetPos, f_Edit1Pos,,, Edit1, ahk_id %f_window_id%

if f_AlwaysShowMenu = n ; The menu should be shown only selectively.

{

    if f_class in #32770,ExploreWClass,CabinetWClass ; Dialog or Explorer.

    {

        if f_Edit1Pos = ; The control doesn't exist, so don't display the menu

            return

    }

    else if f_class <> ConsoleWindowClass

        return ; Since it's some other window type, don't display menu.

}

; Otherwise, the menu should be presented for this type of window:

Menu, Favorites, show

return

```

30.24 IntelliSense -- by Rajat (requires XP/2k/NT)

This script watches while you edit an AutoHotkey script. When it sees you type a command followed by a comma or space, it displays that command's parameter list to guide you. In addition, you can press Ctrl+F1 (or another hotkey of your choice) to display that command's page in the help file. To dismiss the parameter list, press Escape or Enter.

[Download This Script](#) | [Other Sample Scripts](#)(See 13.) | [Home](#)

```

; Requires v1.0.41+

;

; CONFIGURATION SECTION: Customize the script with the following variables.

```

```
; The hotkey below is pressed to display the current command's page in the
; help file:
I_HelpHotkey = ^F1

; The string below must exist somewhere in the active window's title for
; IntelliSense to be in effect while you're typing. Make it blank to have
; IntelliSense operate in all windows. Make it Pad to have it operate in
; editors such as Metapad, Notepad, and Textpad. Make it .ahk to have it
; operate only when a .ahk file is open in Notepad, Metapad, etc.
I_Editor = pad

; If you wish to have a different icon for this script to distinguish it from
; other scripts in the tray, provide the filename below (leave blank to have
; no icon). For example: E:\stuff\Pics\icons\GeoIcons\Information.ico
I_Icon =

; END OF CONFIGURATION SECTION (do not make changes below this point unless
; you want to change the basic functionality of the script).

SetKeyDelay, 0
#SingleInstance

if I_HelpHotkey <>
    Hotkey, %I_HelpHotkey%, I_HelpHotkey

; Change tray icon (if one was specified in the configuration section above):
if I_Icon <>
    IfExist, %I_Icon%
        Menu, Tray, Icon, %I_Icon%
```

```

; Determine AutoHotkey's location:

RegRead, ahk_dir, HKEY_LOCAL_MACHINE, SOFTWARE\AutoHotkey, InstallDir
if ErrorLevel ; Not found, so look for it in some other common locations.

{
    if A_AhkPath
        SplitPath, A_AhkPath,, ahk_dir
    else IfExist ..\..\AutoHotkey.chm
        ahk_dir = ..\..
    else IfExist %A_ProgramFiles%\AutoHotkey\AutoHotkey.chm
        ahk_dir = %A_ProgramFiles%\AutoHotkey
    else
    {
        MsgBox Could not find the AutoHotkey folder.
        ExitApp
    }
}

ahk_help_file = %ahk_dir%\AutoHotkey.chm

; Read command syntaxes:

Loop, Read, %ahk_dir%\Extras\Editors\Syntax\Commands.txt
{
    I_FullCmd = %A_LoopReadLine%
    ; Directives have a first space instead of a first comma.
    ; So use whichever comes first as the end of the command name:
    StringGetPos, I_cPos, I_FullCmd, `,
    StringGetPos, I_sPos, I_FullCmd, %A_Space%

```

```

if (I_cPos = -1 or (I_cPos > I_sPos and I_sPos <> -1))

    I_EndPos := I_sPos

else

    I_EndPos := I_cPos


if I_EndPos <> -1

    StringLeft, I_CurrCmd, I_FullCmd, %I_EndPos%

else ; This is a directive/command with no parameters.

    I_CurrCmd = %A_LoopReadLine%


StringReplace, I_CurrCmd, I_CurrCmd, [,, All

StringReplace, I_CurrCmd, I_CurrCmd, %A_Space%,, All

StringReplace, I_FullCmd, I_FullCmd, ``n, `n, All

StringReplace, I_FullCmd, I_FullCmd, ``t, `t, All


; Make arrays of command names and full cmd syntaxes:

I_Cmd%A_Index% = %I_CurrCmd%

I_FullCmd%A_Index% = %I_FullCmd%


}

;

; Use the Input command to watch for commands that the user types:

Loop

{

    ; Editor window check:

    WinGetTitle, ActiveTitle, A

    IfNotInString, ActiveTitle, %I_Editor%


    {

        ToolTip

        Sleep, 500
}

```

```
Continue

}

; Get all keys till endkey:

Input, I_Word, V, {enter}{escape}{space}`,
I_EndKey = %ErrorLevel%


; Tooltip is hidden in these cases:

if I_EndKey in EndKey:Enter,EndKey:Escape
{
    Tooltip
    Continue
}

; Editor window check again!

WinGetActiveTitle, ActiveTitle
IfNotInString, ActiveTitle, %I_Editor%
{
    Tooltip
    Continue
}

; Compensate for any indentation that is present:

StringReplace, I_Word, I_Word, %A_Space%,, All
StringReplace, I_Word, I_Word, %A_Tab%,, All
if I_Word =
{
    Continue
}

; Check for commented line:
```

```
StringLeft, I_Check, I_Word, 1

if (I_Check = ";" or I_Word = "If") ; "If" seems a little too annoying to show tooltip
for.

    Continue

; Match word with command:

I_Index =

Loop

{

    ; It helps performance to resolve dynamic variables only once.

    ; In addition, the value put into I_ThisCmd is also used by the
    ; I_HelpHotkey subroutine:

I_ThisCmd := I_Cmd%A_Index%

if I_ThisCmd =

    break

if (I_Word = I_ThisCmd)

{

    I_Index := A_Index

    I_HelpOn = %I_ThisCmd%

    break

}

}

; If no match then resume watching user input:

if I_Index =

    Continue

; Show matched command to guide the user:

I_ThisFullCmd := I_FullCmd%I_Index%
```

```
ToolTip, %I_ThisFullCmd%, A_CaretX, A_CaretY + 20
}

I_HelpHotkey:
WinGetTitle, ActiveTitle, A
IfNotInString, ActiveTitle, %I_Editor%, Return

ToolTip ; Turn off syntax helper since there is no need for it now.

SetTitleMatchMode, 1 ; In case it's 3. This setting is in effect only for this thread.

IfWinNotExist, AutoHotkey Help
{
    IfNotExist, %ahk_help_file%
    {
        MsgBox, Could not find the help file: %ahk_help_file%.
        return
    }
    Run, %ahk_help_file%
    WinWait, AutoHotkey Help
}

if I_ThisCmd = ; Instead, use what was most recently typed.
    I_ThisCmd := I_Word

; The above has set the "last found" window which we use below:
WinActivate
WinWaitActive
```

```

StringReplace, I_ThisCmd, I_ThisCmd, #, {#} ; Replace leading #, if any.

Send, !{home}+{end}%I_HelpOn%{enter}

return

```

30.25 Using a Joystick as a Mouse

This script converts a joystick into a three-button mouse. It allows each button to drag just like a mouse button and it uses virtually no CPU time. Also, it will move the cursor faster depending on how far you push the joystick from center. You can personalize various settings at the top of the script.

[Download This Script](#) | [Other Sample Scripts](#)(See 13.) | [Home](#)

```

; Increase the following value to make the mouse cursor move faster:

JoyMultiplier = 0.30


; Decrease the following value to require less joystick displacement-from-center
; to start moving the mouse. However, you may need to calibrate your joystick
; -- ensuring it's properly centered -- to avoid cursor drift. A perfectly tight
; and centered joystick could use a value of 1:

JoyThreshold = 3


; Change the following to true to invert the Y-axis, which causes the mouse to
; move vertically in the direction opposite the stick:

InvertYAxis := false


; Change these values to use joystick button numbers other than 1, 2, and 3 for
; the left, right, and middle mouse buttons. Available numbers are 1 through 32.
; Use the Joystick Test Script to find out your joystick's numbers more easily.

ButtonLeft = 1
ButtonRight = 2
ButtonMiddle = 3

```

```
; If your joystick has a POV control, you can use it as a mouse wheel. The
; following value is the number of milliseconds between turns of the wheel.
; Decrease it to have the wheel turn faster:

WheelDelay = 250

; If your system has more than one joystick, increase this value to use a joystick
; other than the first:

JoystickNumber = 1

; END OF CONFIG SECTION -- Don't change anything below this point unless you want
; to alter the basic nature of the script.

#SingleInstance

JoystickPrefix = %JoystickNumber%Joy

Hotkey, %JoystickPrefix%%ButtonLeft%, ButtonLeft
Hotkey, %JoystickPrefix%%ButtonRight%, ButtonRight
Hotkey, %JoystickPrefix%%ButtonMiddle%, ButtonMiddle

; Calculate the axis displacements that are needed to start moving the cursor:

JoyThresholdUpper := 50 + JoyThreshold
JoyThresholdLower := 50 - JoyThreshold
if InvertYAxis
    YAxisMultiplier = -1
else
    YAxisMultiplier = 1

SetTimer, WatchJoystick, 10 ; Monitor the movement of the joystick.
```

```
GetKeyState, JoyInfo, %JoystickNumber%JoyInfo

IfInString, JoyInfo, P ; Joystick has POV control, so use it as a mouse wheel.

SetTimer, MouseWheel, %WheelDelay%


return ; End of auto-execute section.

; The subroutines below do not use KeyWait because that would sometimes trap the
; WatchJoystick quasi-thread beneath the wait-for-button-up thread, which would
; effectively prevent mouse-dragging with the joystick.

ButtonLeft:

SetMouseDelay, -1 ; Makes movement smoother.

MouseClick, left,,, 1, 0, D ; Hold down the left mouse button.

SetTimer, WaitForLeftButtonUp, 10

return


ButtonRight:

SetMouseDelay, -1 ; Makes movement smoother.

MouseClick, right,,, 1, 0, D ; Hold down the right mouse button.

SetTimer, WaitForRightButtonUp, 10

return


ButtonMiddle:

SetMouseDelay, -1 ; Makes movement smoother.

MouseClick, middle,,, 1, 0, D ; Hold down the right mouse button.

SetTimer, WaitForMiddleButtonUp, 10

return
```

```
WaitForLeftButtonUp:

if GetKeyState(JoystickPrefix . ButtonLeft)

    return ; The button is still, down, so keep waiting.

; Otherwise, the button has been released.

SetTimer, WaitForLeftButtonUp, off

SetMouseDelay, -1 ; Makes movement smoother.

MouseClick, left,,, 1, 0, U ; Release the mouse button.

return


WaitForRightButtonUp:

if GetKeyState(JoystickPrefix . ButtonRight)

    return ; The button is still, down, so keep waiting.

; Otherwise, the button has been released.

SetTimer, WaitForRightButtonUp, off

MouseClick, right,,, 1, 0, U ; Release the mouse button.

return


WaitForMiddleButtonUp:

if GetKeyState(JoystickPrefix . ButtonMiddle)

    return ; The button is still, down, so keep waiting.

; Otherwise, the button has been released.

SetTimer, WaitForMiddleButtonUp, off

MouseClick, middle,,, 1, 0, U ; Release the mouse button.

return


WatchJoystick:

MouseNeedsToBeMoved := false ; Set default.

SetFormat, float, 03
```

```
GetKeyState, joyx, %JoystickNumber%JoyX  
GetKeyState, joyy, %JoystickNumber%JoyY  
  
if joyx > %JoyThresholdUpper%  
{  
    MouseNeedsToBeMoved := true  
  
    DeltaX := joyx - JoyThresholdUpper  
}  
  
else if joyx < %JoyThresholdLower%  
{  
    MouseNeedsToBeMoved := true  
  
    DeltaX := joyx - JoyThresholdLower  
}  
  
else  
    DeltaX = 0  
  
if joyy > %JoyThresholdUpper%  
{  
    MouseNeedsToBeMoved := true  
  
    DeltaY := joyy - JoyThresholdUpper  
}  
  
else if joyy < %JoyThresholdLower%  
{  
    MouseNeedsToBeMoved := true  
  
    DeltaY := joyy - JoyThresholdLower  
}  
  
else  
    DeltaY = 0  
  
if MouseNeedsToBeMoved  
{  
    SetMouseDelay, -1 ; Makes movement smoother.
```

```

MouseMove, DeltaX * JoyMultiplier, DeltaY * JoyMultiplier * YAxisMultiplier, 0, R
}

return

MouseWheel:

GetKeyState, JoyPOV, %JoystickNumber%JoyPOV

if JoyPOV = -1 ; No angle.

    return

if (JoyPOV > 31500 or JoyPOV < 4500) ; Forward

    Send {WheelUp}

else if JoyPOV between 13500 and 22500 ; Back

    Send {WheelDown}

return

```

30.26 Joystick Test Script

This script helps determine the button numbers and other attributes of your joystick. It might also reveal if your joystick is in need of calibration; that is, whether the range of motion of each of its axes is from 0 to 100 percent as it should be. If calibration is needed, use the operating system's control panel or the software that came with your joystick.

[Download This Script](#) | [Other Sample Scripts](#)(See 13.) | [Home](#)

```

; July 6, 2005: Added auto-detection of joystick number.

; May 8, 2005 : Fixed: JoyAxes is no longer queried as a means of
; detecting whether the joystick is connected. Some joysticks are
; gamepads and don't have even a single axis.

; If you want to unconditionally use a specific joystick number, change
; the following value from 0 to the number of the joystick (1-32).

; A value of 0 causes the joystick number to be auto-detected:

JoystickNumber = 0

```

```
; END OF CONFIG SECTION. Do not make changes below this point unless
; you wish to alter the basic functionality of the script.

; Auto-detect the joystick number if called for:

if JoystickNumber <= 0

{

    Loop 32 ; Query each joystick number to find out which ones exist.

    {

        GetKeyState, JoyName, %A_Index%JoyName

        if JoyName <>

        {

            JoystickNumber = %A_Index%

            break

        }

    }

    if JoystickNumber <= 0

    {

        MsgBox The system does not appear to have any joysticks.

        ExitApp

    }

}

#SingleInstance

SetFormat, float, 03 ; Omit decimal point from axis position percentages.

GetKeyState, joy_buttons, %JoystickNumber%JoyButtons

GetKeyState, joy_name, %JoystickNumber%JoyName

GetKeyState, joy_info, %JoystickNumber%JoyInfo

Loop
```

```
{  
    buttons_down =  
  
    Loop, %joy_buttons%  
  
    {  
        GetKeyState, joy%a_index%, %JoystickNumber%joy%a_index%  
  
        if joy%a_index% = D  
  
            buttons_down = %buttons_down%%a_space%%a_index%  
  
    }  
  
    GetKeyState, joyx, %JoystickNumber%JoyX  
  
    axis_info = X%joyx%  
  
    GetKeyState, joyy, %JoystickNumber%JoyY  
  
    axis_info = %axis_info%%a_space%%a_space%Y%joyy%  
  
    IfInString, joy_info, Z  
  
    {  
  
        GetKeyState, joyz, %JoystickNumber%JoyZ  
  
        axis_info = %axis_info%%a_space%%a_space%Z%joyz%  
  
    }  
  
    IfInString, joy_info, R  
  
    {  
  
        GetKeyState, joyr, %JoystickNumber%JoyR  
  
        axis_info = %axis_info%%a_space%%a_space%R%joyr%  
  
    }  
  
    IfInString, joy_info, U  
  
    {  
  
        GetKeyState, joyu, %JoystickNumber%JoyU  
  
        axis_info = %axis_info%%a_space%%a_space%U%joyu%  
  
    }  
  
    IfInString, joy_info, V  
  
    {
```

```

GetKeyState, joyv, %JoystickNumber%JoyV

axis_info = %axis_info%%a_space%%a_space%V%joyv%

}

IfInString, joy_info, P

{

    GetKeyState, joyp, %JoystickNumber%JoyPOV

    axis_info = %axis_info%%a_space%%a_space%POV%joyp%

}

ToolTip, %joy_name% (%JoystickNumber%):`n%axis_info%`nButtons
Down: %buttons_down%`n`n(right-click the tray icon to exit)

Sleep, 100

}

return

```

30.27 On-Screen Keyboard (requires XP/2k/NT) -- by Jon

This script creates a mock keyboard at the bottom of your screen that shows the keys you are pressing in real time. I made it to help me to learn to touch-type (to get used to not looking at the keyboard). The size of the on-screen keyboard can be customized at the top of the script. Also, you can double-click the tray icon to show or hide the keyboard.

[Download This Script](#) | [Other Sample Scripts](#)(See 13.) | [Home](#)

```

;---- Configuration Section: Customize the size of the on-screen keyboard and
; other options here.

; Changing this font size will make the entire on-screen keyboard get
; larger or smaller:

k_FontSize = 10

k_FontName = Verdana ; This can be blank to use the system's default font.

k_FontStyle = Bold ; Example of an alternative: Italic Underline

```

```
; Names for the tray menu items:  
k_MenuItemHide = Hide on-screen &keyboard  
k_MenuItemShow = Show on-screen &keyboard  
  
; To have the keyboard appear on a monitor other than the primary, specify  
; a number such as 2 for the following variable. Leave it blank to use  
; the primary:  
k_Monitor =  
  
;---- End of configuration section. Don't change anything below this point  
; unless you want to alter the basic nature of the script.  
  
;---- Alter the tray icon menu:  
Menu, Tray, Add, %k_MenuItemHide%, k_ShowHide  
Menu, Tray, Add, &Exit, k_MenuExit  
Menu, Tray, Default, %k_MenuItemHide%  
Menu, Tray, NoStandard  
  
;---- Calculate object dimensions based on chosen font size:  
k_KeyWidth = %k_FontSize%  
k_KeyWidth *= 3  
k_KeyHeight = %k_FontSize%  
k_KeyHeight *= 3  
k_KeyMargin = %k_FontSize%  
k_KeyMargin /= 6  
k_SpacebarWidth = %k_FontSize%  
k_SpacebarWidth *= 25
```

```
k_KeyWidthHalf = %k_KeyWidth%
k_KeyWidthHalf /= 2

k_KeySize = w%k_KeyWidth% h%k_KeyHeight%
k_Position = x+%k_KeyMargin% %k_KeySize%

;---- Create a GUI window for the on-screen keyboard:
Gui, Font, s%k_FontSize% %k_FontStyle%, %k_FontName%
Gui, -Caption +E0x200 +ToolWindow
TransColor = F1ECED
Gui, Color, %TransColor% ; This color will be made transparent later below.

;---- Add a button for each key. Position the first button with absolute
; coordinates so that all other buttons can be positioned relative to it:
Gui, Add, Button, section %k_KeySize% xm+%k_KeyWidth%, 1
Gui, Add, Button, %k_Position%, 2
Gui, Add, Button, %k_Position%, 3
Gui, Add, Button, %k_Position%, 4
Gui, Add, Button, %k_Position%, 5
Gui, Add, Button, %k_Position%, 6
Gui, Add, Button, %k_Position%, 7
Gui, Add, Button, %k_Position%, 8
Gui, Add, Button, %k_Position%, 9
Gui, Add, Button, %k_Position%, 0
Gui, Add, Button, %k_Position%, -
Gui, Add, Button, %k_Position%, =
Gui, Add, Button, %k_Position%, Bk

Gui, Add, Button, xm y+%k_KeyMargin% h%k_KeyHeight%, Tab ; Auto-width.
```

```

Gui, Add, Button, %k_Position%, Q
Gui, Add, Button, %k_Position%, W
Gui, Add, Button, %k_Position%, E
Gui, Add, Button, %k_Position%, R
Gui, Add, Button, %k_Position%, T
Gui, Add, Button, %k_Position%, Y
Gui, Add, Button, %k_Position%, U
Gui, Add, Button, %k_Position%, I
Gui, Add, Button, %k_Position%, O
Gui, Add, Button, %k_Position%, P
Gui, Add, Button, %k_Position%, [
Gui, Add, Button, %k_Position%, ]
Gui, Add, Button, %k_Position%, \

Gui, Add, Button, xs+%k_KeyWidthHalf% y+%k_KeyMargin% %k_KeySize%, A
Gui, Add, Button, %k_Position%, S
Gui, Add, Button, %k_Position%, D
Gui, Add, Button, %k_Position%, F
Gui, Add, Button, %k_Position%, G
Gui, Add, Button, %k_Position%, H
Gui, Add, Button, %k_Position%, J
Gui, Add, Button, %k_Position%, K
Gui, Add, Button, %k_Position%, L
Gui, Add, Button, %k_Position%, `;
Gui, Add, Button, %k_Position%, '
Gui, Add, Button, x+%k_KeyMargin% h%k_KeyHeight%, Enter ; Auto-width.

; The first button below adds %A_Space% at the end to widen it a little,
; making the layout of keys next to it more accurately reflect a real keyboard:

```

```

Gui, Add, Button, xm y+%k_KeyMargin% h%k_KeyHeight%,
Shift%A_Space%%A_Space%

Gui, Add, Button, %k_Position%, Z

Gui, Add, Button, %k_Position%, X

Gui, Add, Button, %k_Position%, C

Gui, Add, Button, %k_Position%, V

Gui, Add, Button, %k_Position%, B

Gui, Add, Button, %k_Position%, N

Gui, Add, Button, %k_Position%, M

Gui, Add, Button, %k_Position%, `,

Gui, Add, Button, %k_Position%, .

Gui, Add, Button, %k_Position%, /


Gui, Add, Button, xm y+%k_KeyMargin% h%k_KeyHeight%, Ctrl ; Auto-width.

Gui, Add, Button, h%k_KeyHeight% x+%k_KeyMargin%, Win ; Auto-width.

Gui, Add, Button, h%k_KeyHeight% x+%k_KeyMargin%, Alt ; Auto-width.

Gui, Add, Button, h%k_KeyHeight% x+%k_KeyMargin% w%k_SpacebarWidth%, Space


;---- Show the window:

Gui, Show

k_IsVisible = y


WinGet, k_ID, ID, A ; Get its window ID.

WinGetPos,,, k_WindowWidth, k_WindowHeight, A


;---- Position the keyboard at the bottom of the screen (taking into account
; the position of the taskbar):

SysGet, k_WorkArea, MonitorWorkArea, %k_Monitor%

```

```

; Calculate window's X-position:

k_WindowX = %k_WorkAreaRight%

k_WindowX -= %k_WorkAreaLeft% ; Now k_WindowX contains the width of this monitor.

k_WindowX -= %k_WindowWidth%

k_WindowX /= 2 ; Calculate position to center it horizontally.

; The following is done in case the window will be on a non-primary monitor

; or if the taskbar is anchored on the left side of the screen:

k_WindowX += %k_WorkAreaLeft%


; Calculate window's Y-position:

k_WindowY = %k_WorkAreaBottom%

k_WindowY -= %k_WindowHeight%


WinMove, A,, %k_WindowX%, %k_WindowY%

WinSet, AlwaysOnTop, On, ahk_id %k_ID%

WinSet, TransColor, %TransColor% 220, ahk_id %k_ID%


;---- Set all keys as hotkeys. See www.asciitable.com

k_n = 1

k_ASCII = 45


Loop

{

    Transform, k_char, Chr, %k_ASCII%

    StringUpper, k_char, k_char

    if k_char not in <,>,^,~,• ,`,

        Hotkey, ~*%k_char%, k_KeyPress

```

```
; In the above, the asterisk prefix allows the key to be detected regardless  
; of whether the user is holding down modifier keys such as Control and Shift.  
  
if k_ASCII = 93  
    break  
  
k_ASCII++  
}  
  
  
return ; End of auto-execute section.  
  
  
;---- When a key is pressed by the user, click the corresponding button on-screen:  
  
  
~*Backspace::  
  
ControlClick, Bk, ahk_id %k_ID%, , LEFT, 1, D  
  
KeyWait, Backspace  
  
ControlClick, Bk, ahk_id %k_ID%, , LEFT, 1, U  
  
return  
  
  
  
  
; LShift and RShift are used rather than "Shift" because when used as a hotkey,  
; "Shift" would default to firing upon release of the key (in older AHK versions):  
  
~*LShift::  
  
~*RShift::  
  
~*LCtrl:: ; Must use Ctrl not Control to match button names.  
  
~*RCtrl::  
  
~*LAlt::  
  
~*RAlt::  
  
~*LWin::  
  
~*RWin::
```

```

StringTrimLeft, k_ThisHotkey, A_ThisHotkey, 3

ControlClick, %k_ThisHotkey%, ahk_id %k_ID%, , LEFT, 1, D

KeyWait, %k_ThisHotkey%

ControlClick, %k_ThisHotkey%, ahk_id %k_ID%, , LEFT, 1, U

return


~*,:::

~*'::

~*Space::

~*Enter::

~*Tab::

k_KeyPress:

StringReplace, k_ThisHotkey, A_ThisHotkey, ~

StringReplace, k_ThisHotkey, k_ThisHotkey, *

SetTitleMatchMode, 3 ; Prevents the T and B keys from being confused with Tab and
Backspace.

ControlClick, %k_ThisHotkey%, ahk_id %k_ID%, , LEFT, 1, D

KeyWait, %k_ThisHotkey%

ControlClick, %k_ThisHotkey%, ahk_id %k_ID%, , LEFT, 1, U

Return


k_ShowHide:

if k_IsVisible = y

{

    Gui, Cancel

    Menu, Tray, Rename, %k_MenuItemHide%, %k_MenuItemShow%


    k_IsVisible = n

```

```

}

else

{

    Gui, Show

    Menu, Tray, Rename, %k_MenuItemShow%, %k_MenuItemHide%

    k_IsVisible = y

}

return

GuiClose:

k_MenuExit:

ExitApp

```

30.28 Minimize Window to Tray Menu

This script assigns a hotkey of your choice to hide any window so that it becomes an entry at the bottom of the script's tray menu. Hidden windows can then be unhidden individually or all at once by selecting the corresponding item on the menu. If the script exits for any reason, all the windows that it hid will be unhidden automatically.

[Download This Script](#) | [Other Sample Scripts](#)(See 13.) | [Home](#)

```

; CHANGES:

; July 22, 2005 (changes provided by egilmour):
; - Added new hotkey to unhide the last hidden window (Win+U)

;

; November 3, 2004 (changes provided by trogdor):
; - Program manager is prevented from being hidden.
; - If there is no active window, the minimize-to-tray hotkey will have
;   no effect rather than waiting indefinitely.

;

```

```
; October 23, 2004:  
;  
; - The taskbar is prevented from being hidden.  
;  
; - Some possible problems with long window titles have been fixed.  
;  
; - Windows without a title can be hidden without causing problems.  
;  
; - If the script is running under AHK v1.0.22 or greater, the  
;  
;   maximum length of each menu item is increased from 100 to 260.  
  
;  
; CONFIGURATION SECTION: Change the below values as desired.  
  
;  
; This is the maximum number of windows to allow to be hidden (having a  
;  
; limit helps performance):  
mwt_MaxWindows = 50  
  
;  
; This is the hotkey used to hide the active window:  
mwt_Hotkey = #h ; Win+H  
  
;  
; This is the hotkey used to unhide the last hidden window:  
mwt_UnHotkey = #u ; Win+U  
  
;  
; If you prefer to have the tray menu empty of all the standard items,  
;  
; such as Help and Pause, use N. Otherwise, use Y:  
mwt_StandardMenu = N  
  
;  
; These next few performance settings help to keep the action within the  
;  
; #HotkeyModifierTimeout period, and thus avoid the need to release and  
;  
; press down the hotkey's modifier if you want to hide more than one  
;  
; window in a row. These settings are not needed if you choose to have the  
;  
; script use the keyboard hook via #InstallKeybdHook or other means:  
#HotkeyModifierTimeout 100
```

```
SetWinDelay 10
SetKeyDelay 0

#SingleInstance ; Allow only one instance of this script to be running.

; END OF CONFIGURATION SECTION (do not make changes below this point
; unless you want to change the basic functionality of the script).

Hotkey, %mwt_Hotkey%, mwt_Minimize
Hotkey, %mwt_UnHotkey%, mwt_UnMinimize

; If the user terminates the script by any means, unhide all the
; windows first:
OnExit, mwt_RestoreAllThenExit

if mwt_StandardMenu = Y
    Menu, Tray, Add
else
{
    Menu, Tray, NoStandard
    Menu, Tray, Add, E&xit and Unhide All, mwt_RestoreAllThenExit
}
Menu, Tray, Add, &Unhide All Hidden Windows, mwt_RestoreAll
Menu, Tray, Add ; Another separator line to make the above more special.

if a_AhkVersion = ; Since it's blank, version is older than 1.0.22.
    mwt_MaxLength = 100
else
    mwt_MaxLength = 260 ; Reduce this to restrict the width of the menu.
```

```
return ; End of auto-execute section.

mwt_Minimize:

if mwt_WindowCount >= %mwt_MaxWindows%

{

    MsgBox No more than %mwt_MaxWindows% may be hidden simultaneously.

    return

}

; Set the "last found window" to simplify and help performance.

; Since in certain cases it is possible for there to be no active window,
; a timeout has been added:

WinWait, A,, 2

if ErrorLevel <> 0 ; It timed out, so do nothing.

    return

; Otherwise, the "last found window" has been set and can now be used:

WinGet, mwt_ActiveID, ID

WinGetTitle, mwt_ActiveTitle

WinGetClass, mwt_ActiveClass

if mwt_ActiveClass in Shell_TrayWnd,Progman

{

    MsgBox The desktop and taskbar cannot be hidden.

    return

}

; Because hiding the window won't deactivate it, activate the window
; beneath this one (if any). I tried other ways, but they wound up
```

```

; activating the task bar. This way sends the active window (which is
; about to be hidden) to the back of the stack, which seems best:

Send, !{esc}

; Hide it only now that WinGetTitle/WinGetClass above have been run (since
; by default, those commands cannot detect hidden windows):

WinHide

; If the title is blank, use the class instead. This serves two purposes:
; 1) A more meaningful name is used as the menu name.
; 2) Allows the menu item to be created (otherwise, blank items wouldn't
;     be handled correctly by the various routines below).

if mwt_ActiveTitle =
    mwt_ActiveTitle = ahk_class %mwt_ActiveClass%
; Ensure the title is short enough to fit. mwt_ActiveTitle also serves to
; uniquely identify this particular menu item.

StringLeft, mwt_ActiveTitle, mwt_ActiveTitle, %mwt_MaxLength%

; In addition to the tray menu requiring that each menu item name be
; unique, it must also be unique so that we can reliably look it up in
; the array when the window is later unhidden. So make it unique if it
; isn't already:

Loop, %mwt_MaxWindows%
{
    if mwt_WindowTitle%a_index% = %mwt_ActiveTitle%
    {
        ; Match found, so it's not unique.

        ; First remove the 0x from the hex number to conserve menu space:

        StringTrimLeft, mwt_ActiveIDShort, mwt_ActiveID, 2
        StringLen, mwt_ActiveIDShortLength, mwt_ActiveIDShort

```

```

StringLen, mwt_ActiveTitleLength, mwt_ActiveTitle
mwt_ActiveTitleLength += %mwt_ActiveIDShortLength%
mwt_ActiveTitleLength += 1 ; +1 the 1 space between title & ID.
if mwt_ActiveTitleLength > %mwt_MaxLength%
{
    ; Since menu item names are limited in length, trim the title
    ; down to allow just enough room for the Window's Short ID at
    ; the end of its name:
    TrimCount = %mwt_ActiveTitleLength%
    TrimCount -= %mwt_MaxLength%
    StringTrimRight, mwt_ActiveTitle, mwt_ActiveTitle, %TrimCount%
}
; Build unique title:
mwt_ActiveTitle = %mwt_ActiveTitle% %mwt_ActiveIDShort%
break
}

; First, ensure that this ID doesn't already exist in the list, which can
; happen if a particular window was externally unhidden (or its app unhid
; it) and now it's about to be re-hidden:
mwt_AlreadyExists = n
Loop, %mwt_MaxWindows%
{
    if mwt_WindowID%a_index% = %mwt_ActiveID%
    {
        mwt_AlreadyExists = y
        break
    }
}

```

```
}
```

; Add the item to the array and to the menu:

```
if mwt_AlreadyExists = n
{
    Menu, Tray, add, %mwt_ActiveTitle%, RestoreFromTrayMenu
    mwt_WindowCount += 1
    Loop, %mwt_MaxWindows% ; Search for a free slot.
    {
        ; It should always find a free slot if things are designed right.
        if mwt_WindowID%a_index% = ; An empty slot was found.
        {
            mwt_WindowID%a_index% = %mwt_ActiveID%
            mwt_WindowTitle%a_index% = %mwt_ActiveTitle%
            break
        }
    }
}
return
```

RestoreFromTrayMenu:

```
Menu, Tray, delete, %A_ThisMenuItem%
; Find window based on its unique title stored as the menu item name:
Loop, %mwt_MaxWindows%
{
    if mwt_WindowTitle%a_index% = %A_ThisMenuItem% ; Match found.
    {
        StringTrimRight, IDToRestore, mwt_WindowID%a_index%, 0
```

```

WinShow, ahk_id %IDToRestore%

WinActivate ahk_id %IDToRestore% ; Sometimes needed.

mwt_WindowID%a_index% = ; Make it blank to free up a slot.

mwt_WindowTitle%a_index% =

mwt_WindowCount -= 1

break

}

}

return

;; This will pop the last minimized window off the stack and unhide it.

mwt_UnMinimize:

;; Make sure there's something to unhide.

if mwt_WindowCount > 0

{

;; Get the id of the last window minimized and unhide it

StringTrimRight, IDToRestore, mwt_WindowID%mwt_WindowCount%, 0

WinShow, ahk_id %IDToRestore%

WinActivate ahk_id %IDToRestore%


;; Get the menu name of the last window minimized and remove it

StringTrimRight, MenuToRemove, mwt_WindowTitle%mwt_WindowCount%, 0

Menu, Tray, delete, %MenuToRemove%


;; clean up our 'arrays' and decrement the window count

mwt_WindowID%mwt_WindowCount% =

mwt_WindowTitle%mwt_WindowCount% =

mwt_WindowCount -= 1

```

```
}

return

mwt_RestoreAllThenExit:

Gosub, mwt_RestoreAll

ExitApp ; Do a true exit.

mwt_RestoreAll:

Loop, %mwt_MaxWindows%{

    if mwt_WindowID%a_index% <>

    {

        StringTrimRight, IDToRestore, mwt_WindowID%a_index%, 0

        WinShow, ahk_id %IDToRestore%

        WinActivate ahk_id %IDToRestore% ; Sometimes needed.

        ; Do it this way vs. DeleteAll so that the sep. line and first

        ; item are retained:

        StringTrimRight, MenuToRemove, mwt_WindowTitle%a_index%, 0

        Menu, Tray, delete, %MenuToRemove%

        mwt_WindowID%a_index% = ; Make it blank to free up a slot.

        mwt_WindowTitle%a_index% =

        mwt_WindowCount -= 1

    }

    if mwt_WindowCount = 0

        break

}

return
```

30.29 Mouse Gestures -- by deguix

This script watches how you move the mouse whenever the right mouse button is being held down. If it sees you "draw" a recognized shape or symbol, it will launch a program or perform another custom action of your choice (just like hotkeys). See the included README file for how to define gestures.

30.30 Changing MsgBox's Button Names

This is a working example script that uses a timer to change the names of the buttons in a MsgBox dialog. Although the button names are changed, the IfMsgBox command still requires that the buttons be referred to by their original names.

[Download This Script](#) | [Other Sample Scripts\(See 13.\)](#) | [Home](#)

```
#SingleInstance

SetTimer, ChangeButtonNames, 50

MsgBox, 4, Add or Delete, Choose a button:

IfMsgBox, YES

    MsgBox, You chose Add.

else

    MsgBox, You chose Delete.

return


ChangeButtonNames:

IfWinNotExist, Add or Delete

    return ; Keep waiting.

SetTimer, ChangeButtonNames, off

WinActivate

ControlSetText, Button1, &Add

ControlSetText, Button2, &Delete

return
```

30.31 Numpad 000 Key

This example script makes the special 000 key that appears on certain keypads into an equals key. You can change the action by replacing the "Send, =" line with line(s) of your choice.

[Download This Script](#) | [Other Sample Scripts](#)(See 13.) | [Home](#)

```
#MaxThreadsPerHotkey 5 ; Allow multiple threads for this hotkey.

$Numpad0:::

#MaxThreadsPerHotkey 1

; Above: Use the $ to force the hook to be used, which prevents an
; infinite loop since this subroutine itself sends Numpad0, which
; would otherwise result in a recursive call to itself.

SetBatchLines, 100 ; Make it run a little faster in this case.

DelayBetweenKeys = 30 ; Adjust this value if it doesn't work.

if A_PriorHotkey = %A_ThisHotkey%
{
    if A_TimeSincePriorHotkey < %DelayBetweenKeys%
    {
        if Numpad0Count =
            Numpad0Count = 2 ; i.e. This one plus the prior one.
        else if Numpad0Count = 0
            Numpad0Count = 2
        else
        {
            ; Since we're here, Numpad0Count must be 2 as set by
            ; prior calls, which means this is the third time the
            ; the key has been pressed. Thus, the hotkey sequence
            ; should fire:
            Numpad0Count = 0
            Send, = ; ***** This is the action for the 000 key
    }
}
```

```

; In all the above cases, we return without further action:

CalledReentrantly = y

return

}

;

; Otherwise, this Numpad0 event is either the first in the series

; or it happened too long after the first one (e.g. perhaps the

; user is holding down the Numpad0 key to auto-repeat it, which

; we want to allow). Therefore, after a short delay -- during

; which another Numpad0 hotkey event may re-entrantly call this

; subroutine -- we'll send the key on through if no reentrant

; calls occurred:

Numpad0Count = 0

CalledReentrantly = n

; During this sleep, this subroutine may be reentrantly called

; (i.e. a simultaneous "thread" which runs in parallel to the

; call we're in now):

Sleep, %DelayBetweenKeys%

if CalledReentrantly = y ; Another "thread" changed the value.

{

    ; Since it was called reentrantly, this key event was the first in

    ; the sequence so should be suppressed (hidden from the system):

    CalledReentrantly = n

    return

}

;

; Otherwise it's not part of the sequence so we send it through normally.

; In other words, the *real* Numpad0 key has been pressed, so we want it

; to have its normal effect:

Send, {Numpad0}

```

```
return
```

30.32 Using Keyboard Numpad as a Mouse -- by deguix

This script makes mousing with your keyboard almost as easy as using a real mouse (maybe even easier for some tasks). It supports up to five mouse buttons and the turning of the mouse wheel. It also features customizable movement speed, acceleration, and "axis inversion".

[Download This Script](#) | [Other Sample Scripts](#)(See 13.) | [Home](#)

```
/*
-----o-----o
|Using Keyboard Numpad as a Mouse
|-----)
| By deguix      / A Script file for AutoHotkey 1.0.22+ |
|-----|
|-----|
|-----|
| This script is an example of use of AutoHotkey. It uses |
| the remapping of numpad keys of a keyboard to transform it |
| into a mouse. Some features are the acceleration which |
| enables you to increase the mouse movement when holding |
| a key for a long time, and the rotation which makes the |
| numpad mouse to "turn". I.e. NumPadDown as NumPadUp |
| and vice-versa. See the list of keys used below: |
|-----|
|-----|
|-----|
| Keys          | Description |
|-----|
| ScrollLock (toggle on)| Activates numpad mouse mode. |
|-----|
|-----|
| NumPad0        | Left mouse button click. |
|-----|
```

NumPad5	Middle mouse button click.	
NumPadDot	Right mouse button click.	
NumPadDiv/NumPadMult	X1/X2 mouse button click. (Win 2k+)	
NumPadSub/NumPadAdd	Moves up/down the mouse wheel.	
----- -----		
NumLock (toggled off)	Activates mouse movement mode.	
----- -----		
NumPadEnd/Down/PgDn/	Mouse movement.	
/Left/Right/Home/Up/		
/PgUp		
----- -----		
NumLock (toggled on)	Activates mouse speed adj. mode.	
----- -----		
NumPad7/NumPad1	Inc./dec. acceleration per	
	button press.	
NumPad8/NumPad2	Inc./dec. initial speed per	
	button press.	
NumPad9/NumPad3	Inc./dec. maximum speed per	
	button press.	
^NumPad7/^NumPad1	Inc./dec. wheel acceleration per	
	button press*.	
^NumPad8/^NumPad2	Inc./dec. wheel initial speed per	
	button press*.	
^NumPad9/^NumPad3	Inc./dec. wheel maximum speed per	
	button press*.	
NumPad4/NumPad6	Inc./dec. rotation angle to	
	right in degrees. (i.e. 180° =	

```
|           (= inverted controls).|  
|-----|  
| * = These options are affected by the mouse wheel speed |  
| adjusted on Control Panel. If you don't have a mouse with |  
| wheel, the default is 3 +/- lines per option button press. |  
o-----o  
*/  
  
;START OF CONFIG SECTION  
  
#SingleInstance force  
#MaxHotkeysPerInterval 500  
  
; Using the keyboard hook to implement the Numpad hotkeys prevents  
; them from interfering with the generation of ANSI characters such  
; as à. This is because AutoHotkey generates such characters  
; by holding down ALT and sending a series of Numpad keystrokes.  
; Hook hotkeys are smart enough to ignore such keystrokes.  
#UseHook  
  
MouseSpeed = 1  
MouseAccelerationSpeed = 1  
MouseMaxSpeed = 5  
  
;Mouse wheel speed is also set on Control Panel. As that  
;will affect the normal mouse behavior, the real speed of  
;these three below are times the normal mouse wheel speed.  
MouseWheelSpeed = 1  
MouseWheelAccelerationSpeed = 1
```

```
MouseWheelMaxSpeed = 5

MouseRotationAngle = 0

;END OF CONFIG SECTION

;This is needed or key presses would faulty send their natural
;actions. Like NumPadDiv would send sometimes "/" to the
;screen.

#InstallKeybdHook

Temp = 0
Temp2 = 0

MouseRotationAnglePart = %MouseRotationAngle%
;Divide by 45° because MouseMove only supports whole numbers,
;and changing the mouse rotation to a number lesser than 45°
;could make strange movements.

;
;For example: 22.5° when pressing NumPadUp:
; First it would move upwards until the speed
; to the side reaches 1.

MouseRotationAnglePart /= 45

MouseCurrentAccelerationSpeed = 0
MouseCurrentSpeed = %MouseSpeed%

MouseWheelCurrentAccelerationSpeed = 0
MouseWheelCurrentSpeed = %MouseSpeed%
```

```
SetKeyDelay, -1  
SetMouseDelay, -1  
  
Hotkey, *NumPad0, ButtonLeftClick  
Hotkey, *NumpadIns, ButtonLeftClickIns  
Hotkey, *NumPad5, ButtonMiddleClick  
Hotkey, *NumpadClear, ButtonMiddleClickClear  
Hotkey, *NumPadDot, ButtonRightClick  
Hotkey, *NumPadDel, ButtonRightClickDel  
Hotkey, *NumPadDiv, ButtonX1Click  
Hotkey, *NumPadMult, ButtonX2Click  
  
Hotkey, *NumpadSub, ButtonWheelUp  
Hotkey, *NumpadAdd, ButtonWheelDown  
  
Hotkey, *NumPadUp, ButtonUp  
Hotkey, *NumPadDown, ButtonDown  
Hotkey, *NumPadLeft, ButtonLeft  
Hotkey, *NumPadRight, ButtonRight  
Hotkey, *NumPadHome, ButtonUpLeft  
Hotkey, *NumPadEnd, ButtonUpRight  
Hotkey, *NumPadPgUp, ButtonDownLeft  
Hotkey, *NumPadPgDn, ButtonDownRight  
  
Hotkey, Numpad8, ButtonSpeedUp  
Hotkey, Numpad2, ButtonSpeedDown  
Hotkey, Numpad7, ButtonAccelerationSpeedUp  
Hotkey, Numpad1, ButtonAccelerationSpeedDown
```

```
Hotkey, NumPad9, ButtonMaxSpeedUp  
Hotkey, NumPad3, ButtonMaxSpeedDown  
  
Hotkey, NumPad6, ButtonRotationAngleUp  
Hotkey, NumPad4, ButtonRotationAngleDown  
  
Hotkey, !NumPad8, ButtonWheelSpeedUp  
Hotkey, !NumPad2, ButtonWheelSpeedDown  
Hotkey, !NumPad7, ButtonWheelAccelerationSpeedUp  
Hotkey, !NumPad1, ButtonWheelAccelerationSpeedDown  
Hotkey, !NumPad9, ButtonWheelMaxSpeedUp  
Hotkey, !NumPad3, ButtonWheelMaxSpeedDown  
  
Gosub, ~ScrollLock ; Initialize based on current ScrollLock state.  
return  
  
;Key activation support  
  
~ScrollLock::  
; Wait for it to be released because otherwise the hook state gets reset  
; while the key is down, which causes the up-event to get suppressed,  
; which in turn prevents toggling of the ScrollLock state/light:  
KeyWait, ScrollLock  
GetKeyState, ScrollLockState, ScrollLock, T  
If ScrollLockState = D  
{  
    Hotkey, *NumPad0, on  
    Hotkey, *NumPadIns, on  
    Hotkey, *NumPad5, on
```

Hotkey, *NumPadDot, on

Hotkey, *NumPadDel, on

Hotkey, *NumPadDiv, on

Hotkey, *NumPadMult, on

Hotkey, *NumpadSub, on

Hotkey, *NumpadAdd, on

Hotkey, *NumPadUp, on

Hotkey, *NumPadDown, on

Hotkey, *NumPadLeft, on

Hotkey, *NumPadRight, on

Hotkey, *NumPadHome, on

Hotkey, *NumPadEnd, on

Hotkey, *NumPadPgUp, on

Hotkey, *NumPadPgDn, on

Hotkey, Numpad8, on

Hotkey, Numpad2, on

Hotkey, Numpad7, on

Hotkey, Numpad1, on

Hotkey, Numpad9, on

Hotkey, Numpad3, on

Hotkey, Numpad6, on

Hotkey, Numpad4, on

Hotkey, !Numpad8, on

Hotkey, !Numpad2, on

```
Hotkey, !Numpad7, on  
Hotkey, !Numpad1, on  
Hotkey, !Numpad9, on  
Hotkey, !Numpad3, on  
}  
else  
{  
    Hotkey, *NumPad0, off  
    Hotkey, *NumpadIns, off  
    Hotkey, *NumPad5, off  
    Hotkey, *NumPadDot, off  
    Hotkey, *NumPadDel, off  
    Hotkey, *NumPadDiv, off  
    Hotkey, *NumPadMult, off  
  
    Hotkey, *NumpadSub, off  
    Hotkey, *NumpadAdd, off  
  
    Hotkey, *NumPadUp, off  
    Hotkey, *NumPadDown, off  
    Hotkey, *NumPadLeft, off  
    Hotkey, *NumPadRight, off  
    Hotkey, *NumPadHome, off  
    Hotkey, *NumPadEnd, off  
    Hotkey, *NumPadPgUp, off  
    Hotkey, *NumPadPgDn, off  
  
    Hotkey, Numpad8, off  
    Hotkey, Numpad2, off
```

```
Hotkey, Numpad7, off
Hotkey, Numpad1, off
Hotkey, Numpad9, off
Hotkey, Numpad3, off

Hotkey, Numpad6, off
Hotkey, Numpad4, off

Hotkey, !Numpad8, off
Hotkey, !Numpad2, off
Hotkey, !Numpad7, off
Hotkey, !Numpad1, off
Hotkey, !Numpad9, off
Hotkey, !Numpad3, off

}

return

;Mouse click support

ButtonLeftClick:
GetKeyState, already_down_state, LButton
If already_down_state = D
    return
Button2 = NumPad0
ButtonClick = Left
Goto ButtonClickStart

ButtonLeftClickIns:
GetKeyState, already_down_state, LButton
If already_down_state = D
```

```
return

Button2 = NumPadIns

ButtonClick = Left

Goto ButtonClickStart


ButtonMiddleClick:

GetKeyState, already_down_state, MButton

If already_down_state = D

    return

Button2 = NumPad5

ButtonClick = Middle

Goto ButtonClickStart

ButtonMiddleClickClear:

GetKeyState, already_down_state, MButton

If already_down_state = D

    return

Button2 = NumPadClear

ButtonClick = Middle

Goto ButtonClickStart


ButtonRightClick:

GetKeyState, already_down_state, RButton

If already_down_state = D

    return

Button2 = NumPadDot

ButtonClick = Right

Goto ButtonClickStart

ButtonRightClickDel:

GetKeyState, already_down_state, RButton
```

```
If already_down_state = D
    return

Button2 = NumPadDel
ButtonClick = Right
Goto ButtonClickStart

ButtonX1Click:
GetKeyState, already_down_state, XButton1
If already_down_state = D
    return
Button2 = NumPadDiv
ButtonClick = X1
Goto ButtonClickStart

ButtonX2Click:
GetKeyState, already_down_state, XButton2
If already_down_state = D
    return
Button2 = NumPadMult
ButtonClick = X2
Goto ButtonClickStart

ButtonClickStart:
MouseClick, %ButtonClick%,,, 1, 0, D
SetTimer, ButtonClickEnd, 10
return

ButtonClickEnd:
GetKeyState, kclickstate, %Button2%, P
```

```
if kclickstate = D
    return

SetTimer, ButtonClickEnd, off
MouseClick, %ButtonClick%,,, 1, 0, U
return

;Mouse movement support

ButtonSpeedUp:
MouseSpeed++
ToolTip, Mouse speed: %MouseSpeed% pixels
SetTimer, RemoveToolTip, 1000
return

ButtonSpeedDown:
If MouseSpeed > 1
    MouseSpeed--
If MouseSpeed = 1
    ToolTip, Mouse speed: %MouseSpeed% pixel
else
    ToolTip, Mouse speed: %MouseSpeed% pixels
SetTimer, RemoveToolTip, 1000
return

ButtonAccelerationSpeedUp:
MouseAccelerationSpeed++
ToolTip, Mouse acceleration speed: %MouseAccelerationSpeed% pixels
SetTimer, RemoveToolTip, 1000
return

ButtonAccelerationSpeedDown:
```

```
If MouseAccelerationSpeed > 1  
    MouseAccelerationSpeed--  
  
If MouseAccelerationSpeed = 1  
    ToolTip, Mouse acceleration speed: %MouseAccelerationSpeed% pixel  
  
else  
    ToolTip, Mouse acceleration speed: %MouseAccelerationSpeed% pixels  
  
SetTimer, RemoveToolTip, 1000  
  
return  
  
  
ButtonMaxSpeedUp:  
  
MouseMaxSpeed++  
  
ToolTip, Mouse maximum speed: %MouseMaxSpeed% pixels  
  
SetTimer, RemoveToolTip, 1000  
  
return  
  
  
ButtonMaxSpeedDown:  
  
If MouseMaxSpeed > 1  
    MouseMaxSpeed--  
  
If MouseMaxSpeed = 1  
    ToolTip, Mouse maximum speed: %MouseMaxSpeed% pixel  
  
else  
    ToolTip, Mouse maximum speed: %MouseMaxSpeed% pixels  
  
SetTimer, RemoveToolTip, 1000  
  
return  
  
  
ButtonRotationAngleUp:  
  
MouseRotationAnglePart++  
  
If MouseRotationAnglePart >= 8  
    MouseRotationAnglePart = 0  
  
MouseRotationAngle = %MouseRotationAnglePart%
```

```
MouseRotationAngle *= 45

ToolTip, Mouse rotation angle: %MouseRotationAngle%°

SetTimer, RemoveToolTip, 1000

return

ButtonRotationAngleDown:

MouseRotationAnglePart--

If MouseRotationAnglePart < 0

    MouseRotationAnglePart = 7

MouseRotationAngle = %MouseRotationAnglePart%

MouseRotationAngle *= 45

ToolTip, Mouse rotation angle: %MouseRotationAngle%°

SetTimer, RemoveToolTip, 1000

return


ButtonUp:

ButtonDown:

ButtonLeft:

ButtonRight:

ButtonUpLeft:

ButtonUpRight:

ButtonDownLeft:

ButtonDownRight:

If Button <> 0

{

    IfNotInString, A_ThisHotkey, %Button%


    MouseCurrentAccelerationSpeed = 0

    MouseCurrentSpeed = %MouseSpeed%


}
```

```

}

StringReplace, Button, A_ThisHotkey, *

ButtonAccelerationStart:

If MouseAccelerationSpeed >= 1

{

If MouseMaxSpeed > %MouseCurrentSpeed%

{

Temp = 0.001

Temp *= %MouseAccelerationSpeed%

MouseCurrentAccelerationSpeed += %Temp%

MouseCurrentSpeed += %MouseCurrentAccelerationSpeed%


}

}

;

;MouseRotationAngle conversion to speed of button direction

{

MouseCurrentSpeedToDirection = %MouseRotationAngle%

MouseCurrentSpeedToDirection /= 90.0

Temp = %MouseCurrentSpeedToDirection%


if Temp >= 0

{

if Temp < 1

{



MouseCurrentSpeedToDirection = 1

MouseCurrentSpeedToDirection -= %Temp%


Goto EndMouseCurrentSpeedToDirectionCalculation


}

```

```
}

if Temp >= 1

{

    if Temp < 2

    {

        MouseCurrentSpeedToDirection = 0

        Temp -= 1

        MouseCurrentSpeedToDirection -= %Temp%

        Goto EndMouseCurrentSpeedToDirectionCalculation

    }

}

if Temp >= 2

{

    if Temp < 3

    {

        MouseCurrentSpeedToDirection = -1

        Temp -= 2

        MouseCurrentSpeedToDirection += %Temp%

        Goto EndMouseCurrentSpeedToDirectionCalculation

    }

}

if Temp >= 3

{

    if Temp < 4

    {

        MouseCurrentSpeedToDirection = 0

        Temp -= 3

        MouseCurrentSpeedToDirection += %Temp%

        Goto EndMouseCurrentSpeedToDirectionCalculation

    }

}
```

```
        }

    }

}

EndMouseCurrentSpeedToDirectionCalculation:

;MouseRotationAngle converton to speed of 90 degrees to right

{

    MouseCurrentSpeedToSide = %MouseRotationAngle%

    MouseCurrentSpeedToSide /= 90.0

    Temp = %MouseCurrentSpeedToSide%

    Transform, Temp, mod, %Temp%, 4


    if Temp >= 0

    {

        if Temp < 1

        {

            MouseCurrentSpeedToSide = 0

            MouseCurrentSpeedToSide += %Temp%

            Goto EndMouseCurrentSpeedToSideCalculation

        }

    }

    if Temp >= 1

    {

        if Temp < 2

        {

            MouseCurrentSpeedToSide = 1

            Temp -= 1

            MouseCurrentSpeedToSide -= %Temp%

            Goto EndMouseCurrentSpeedToSideCalculation

        }

    }

}
```

```
        }

    }

    if Temp >= 2

    {

        if Temp < 3

        {

            MouseCurrentSpeedToSide = 0

            Temp -= 2

            MouseCurrentSpeedToSide -= %Temp%

            Goto EndMouseCurrentSpeedToSideCalculation

        }

    }

    if Temp >= 3

    {

        if Temp < 4

        {

            MouseCurrentSpeedToSide = -1

            Temp -= 3

            MouseCurrentSpeedToSide += %Temp%

            Goto EndMouseCurrentSpeedToSideCalculation

        }

    }

}

EndMouseCurrentSpeedToSideCalculation:

MouseCurrentSpeedToDirection *= %MouseCurrentSpeed%

MouseCurrentSpeedToSide *= %MouseCurrentSpeed%


Temp = %MouseRotationAnglePart%
```

```
Transform, Temp, Mod, %Temp%, 2

If Button = NumPadUp
{
    if Temp = 1
    {
        MouseCurrentSpeedToSide *= 2
        MouseCurrentSpeedToDirection *= 2
    }

    MouseCurrentSpeedToDirection *= -1
    MouseMove, %MouseCurrentSpeedToSide%, %MouseCurrentSpeedToDirection%, 0,
R
}

else if Button = NumPadDown
{
    if Temp = 1
    {
        MouseCurrentSpeedToSide *= 2
        MouseCurrentSpeedToDirection *= 2
    }

    MouseCurrentSpeedToSide *= -1
    MouseMove, %MouseCurrentSpeedToSide%, %MouseCurrentSpeedToDirection%, 0,
R
}

else if Button = NumPadLeft
{
    if Temp = 1
    {
```

```
MouseCurrentSpeedToSide *= 2

MouseCurrentSpeedToDirection *= 2

}

MouseCurrentSpeedToSide *= -1

MouseCurrentSpeedToDirection *= -1

MouseMove, %MouseCurrentSpeedToDirection%, %MouseCurrentSpeedToSide%, 0,
R

}

else if Button = NumPadRight

{

    if Temp = 1

    {

        MouseCurrentSpeedToSide *= 2

        MouseCurrentSpeedToDirection *= 2

    }

MouseMove, %MouseCurrentSpeedToDirection%, %MouseCurrentSpeedToSide%, 0,
R

}

else if Button = NumPadHome

{

    Temp = %MouseCurrentSpeedToDirection%

    Temp -= %MouseCurrentSpeedToSide%

    Temp *= -1

    Temp2 = %MouseCurrentSpeedToDirection%

    Temp2 += %MouseCurrentSpeedToSide%

    Temp2 *= -1

    MouseMove, %Temp%, %Temp2%, 0, R
```

```
}

else if Button = NumPadPgUp

{

    Temp = %MouseCurrentSpeedToDirection%

    Temp += %MouseCurrentSpeedToSide%

    Temp2 = %MouseCurrentSpeedToDirection%

    Temp2 -= %MouseCurrentSpeedToSide%

    Temp2 *= -1

    MouseMove, %Temp%, %Temp2%, 0, R

}

else if Button = NumPadEnd

{

    Temp = %MouseCurrentSpeedToDirection%

    Temp += %MouseCurrentSpeedToSide%

    Temp *= -1

    Temp2 = %MouseCurrentSpeedToDirection%

    Temp2 -= %MouseCurrentSpeedToSide%

    MouseMove, %Temp%, %Temp2%, 0, R

}

else if Button = NumPadPgDn

{

    Temp = %MouseCurrentSpeedToDirection%

    Temp -= %MouseCurrentSpeedToSide%

    Temp2 *= -1

    Temp2 = %MouseCurrentSpeedToDirection%

    Temp2 += %MouseCurrentSpeedToSide%

    MouseMove, %Temp%, %Temp2%, 0, R

}
```

```
SetTimer, ButtonAccelerationEnd, 10
return

ButtonAccelerationEnd:
GetKeyState, kstate, %Button%, P
if kstate = D
    Goto ButtonAccelerationStart

SetTimer, ButtonAccelerationEnd, off
MouseCurrentAccelerationSpeed = 0
MouseCurrentSpeed = %MouseSpeed%
Button = 0
return

;Mouse wheel movement support

ButtonWheelSpeedUp:
MouseWheelSpeed++
RegRead, MouseWheelSpeedMultiplier, HKCU, Control Panel\Desktop, WheelScrollLines
If MouseWheelSpeedMultiplier <= 0
    MouseWheelSpeedMultiplier = 1
MouseWheelSpeedReal = %MouseWheelSpeed%
MouseWheelSpeedReal *= %MouseWheelSpeedMultiplier%
ToolTip, Mouse wheel speed: %MouseWheelSpeedReal% lines
SetTimer, RemoveToolTip, 1000
return

ButtonWheelSpeedDown:
RegRead, MouseWheelSpeedMultiplier, HKCU, Control Panel\Desktop, WheelScrollLines
If MouseWheelSpeedMultiplier <= 0
```

```
MouseWheelSpeedMultiplier = 1

If MouseWheelSpeedReal > %MouseWheelSpeedMultiplier%
{
    MouseWheelSpeed--
    MouseWheelSpeedReal = %MouseWheelSpeed%
    MouseWheelSpeedReal *= %MouseWheelSpeedMultiplier%
}

If MouseWheelSpeedReal = 1
    ToolTip, Mouse wheel speed: %MouseWheelSpeedReal% line
else
    ToolTip, Mouse wheel speed: %MouseWheelSpeedReal% lines
SetTimer, RemoveToolTip, 1000
return

ButtonWheelAccelerationSpeedUp:
MouseWheelAccelerationSpeed++
RegRead, MouseWheelSpeedMultiplier, HKCU, Control Panel\Desktop, WheelScrollLines
If MouseWheelSpeedMultiplier <= 0
    MouseWheelSpeedMultiplier = 1
MouseWheelAccelerationSpeedReal = %MouseWheelAccelerationSpeed%
MouseWheelAccelerationSpeedReal *= %MouseWheelSpeedMultiplier%
ToolTip, Mouse wheel acceleration speed: %MouseWheelAccelerationSpeedReal% lines
SetTimer, RemoveToolTip, 1000
return

ButtonWheelAccelerationSpeedDown:
RegRead, MouseWheelSpeedMultiplier, HKCU, Control Panel\Desktop, WheelScrollLines
If MouseWheelSpeedMultiplier <= 0
    MouseWheelSpeedMultiplier = 1
If MouseWheelAccelerationSpeed > 1
```

```

{

    MouseWheelAccelerationSpeed--

    MouseWheelAccelerationSpeedReal = %MouseWheelAccelerationSpeed%

    MouseWheelAccelerationSpeedReal *= %MouseWheelSpeedMultiplier%


}

If MouseWheelAccelerationSpeedReal = 1

    ToolTip, Mouse wheel acceleration speed: %MouseWheelAccelerationSpeedReal% line
else

    ToolTip, Mouse wheel acceleration speed: %MouseWheelAccelerationSpeedReal%
lines

SetTimer, RemoveToolTip, 1000

return


ButtonWheelMaxSpeedUp:

MouseWheelMaxSpeed++

RegRead, MouseWheelSpeedMultiplier, HKCU, Control Panel\Desktop, WheelScrollLines

If MouseWheelSpeedMultiplier <= 0

    MouseWheelSpeedMultiplier = 1

MouseWheelMaxSpeedReal = %MouseWheelMaxSpeed%

MouseWheelMaxSpeedReal *= %MouseWheelSpeedMultiplier%


ToolTip, Mouse wheel maximum speed: %MouseWheelMaxSpeedReal% lines

SetTimer, RemoveToolTip, 1000

return


ButtonWheelMaxSpeedDown:

RegRead, MouseWheelSpeedMultiplier, HKCU, Control Panel\Desktop, WheelScrollLines

If MouseWheelSpeedMultiplier <= 0

    MouseWheelSpeedMultiplier = 1

If MouseWheelMaxSpeed > 1

{

```

```
MouseWheelMaxSpeed--  
  
MouseWheelMaxSpeedReal = %MouseWheelMaxSpeed%  
  
MouseWheelMaxSpeedReal *= %MouseWheelSpeedMultiplier%  
  
}  
  
If MouseWheelMaxSpeedReal = 1  
  
    ToolTip, Mouse wheel maximum speed: %MouseWheelMaxSpeedReal% line  
  
else  
  
    ToolTip, Mouse wheel maximum speed: %MouseWheelMaxSpeedReal% lines  
  
SetTimer, RemoveToolTip, 1000  
  
return  
  
  
ButtonWheelUp:  
  
ButtonWheelDown:  
  
  
If Button <> 0  
  
{  
  
    If Button <> %A_ThisHotkey%  
  
    {  
  
        MouseWheelCurrentAccelerationSpeed = 0  
  
        MouseWheelCurrentSpeed = %MouseWheelSpeed%  
  
    }  
  
}  
  
StringReplace, Button, A_ThisHotkey, *  
  
  
ButtonWheelAccelerationStart:  
  
If MouseWheelAccelerationSpeed >= 1  
  
{  
  
    If MouseWheelMaxSpeed > %MouseWheelCurrentSpeed%  
  
    {
```

```
Temp = 0.001

Temp *= %MouseWheelAccelerationSpeed%

MouseWheelCurrentAccelerationSpeed += %Temp%

MouseWheelCurrentSpeed += %MouseWheelCurrentAccelerationSpeed%


}

}

If Button = NumPadSub

    MouseClick, wheelup,,, %MouseWheelCurrentSpeed%, 0, D

else if Button = NumPadAdd

    MouseClick, wheeldown,,, %MouseWheelCurrentSpeed%, 0, D

SetTimer, ButtonWheelAccelerationEnd, 100

return


ButtonWheelAccelerationEnd:

GetKeyState, kstate, %Button%, P

if kstate = D

    Goto ButtonWheelAccelerationStart


MouseWheelCurrentAccelerationSpeed = 0

MouseWheelCurrentSpeed = %MouseWheelSpeed%

Button = 0

return


RemoveToolTip:

SetTimer, RemoveToolTip, Off

ToolTip

return
```

30.33 ToolTip Mouse Menu (requires XP/2k/NT) -- by Rajat

This script displays a popup menu in response to briefly holding down the middle mouse button. Select a menu item by left-clicking it. Cancel the menu by left-clicking outside of it. A recent improvement is that the contents of the menu can change depending on which type of window is active (Notepad and Word are used as examples here).

[Download This Script](#) | [Other Sample Scripts\(See 13.\)](#) | [Home](#)

; You can set any title here for the menu:

```
MenuTitle = =====
```

; This is how long the mouse button must be held to cause the menu to appear:

```
UMDelay = 20
```

```
SetFormat, float, 0.0
```

```
SetBatchLines, 10ms
```

```
SetTitleMatchMode, 2
```

```
#SingleInstance
```

```
;
```

```
;_____Menu Definitions_____
```

; Create / Edit Menu Items here.

; You can't use spaces in keys/values/section names.

; Don't worry about the order, the menu will be sorted.

```
MenuItems = Notepad/Calculator/Section 3/Section 4/Section 5
```

```
;_____  
;_____Dynamic menuitems here_____
```

; Syntax:

```
;      Dyn# = MenuItem|Window title
```

```
Dyn1 = MS Word|- Microsoft Word
```

```
Dyn2 = Notepad II|- Notepad
```

```
;_____
```

Exit

```
;_____  
;_____Menu Sections_____
```

; Create / Edit Menu Sections here.

Notepad:

```
Run, Notepad.exe
```

Return

Calculator:

```
Run, Calc
```

Return

Section3:

MsgBox, You selected 3

Return

Section4:

MsgBox, You selected 4

Return

Section5:

MsgBox, You selected 5

Return

MSWord:

msgbox, this is a dynamic entry (word)

Return

NotepadII:

msgbox, this is a dynamic entry (notepad)

Return

;

;Hotkey Section

~MButton::

HowLong = 0

Loop

{

 HowLong ++

```
Sleep, 10

GetKeyState, MButton, MButton, P

IfEqual, MButton, U, Break

}

IfLess, HowLong, %UMDelay%, Return

;prepares dynamic menu

DynMenu =

Loop

{

    IfEqual, Dyn%a_index%%,, Break

    StringGetPos, ppos, dyn%a_index%, |

    StringLeft, item, dyn%a_index%, %ppos%

    ppos += 2

    StringMid, win, dyn%a_index%, %ppos%, 1000

    IfWinActive, %win%,

        DynMenu = %DynMenu%/%item%


}

;Joins sorted main menu and dynamic menu

Sort, MenuItems, D/

TempMenu = %MenuItems% %DynMenu%


;clears earlier entries
```

```
Loop
{
    IfEqual, MenuItem%a_index%,,
        Break
    MenuItem%a_index% =
}

;creates new entries
Loop, Parse, TempMenu, /
{
    MenuItem%a_index% = %a_loopfield%
}

;creates the menu
Menu = %MenuTitle%
Loop
{
    IfEqual, MenuItem%a_index%,,
        Break
    numItems ++
    StringTrimLeft, MenuText, MenuItem%a_index%, 0
    Menu = %Menu%`n%MenuText%
}

MouseGetPos, mX, mY
HotKey, ~LButton, MenuClick
HotKey, ~LButton, On
ToolTip, %Menu%, %mX%, %mY%
WinActivate, %MenuTitle%
Return
```

```

MenuClick:
HotKey, ~LButton, Off
IfWinNotActive, %MenuTitle%
{
    ToolTip
    Return
}

MouseGetPos, mX, mY
ToolTip
mY -= 3           ;space after which first line starts
mY /= 13         ;space taken by each line
IfLess, mY, 1, Return
IfGreater, mY, %numItems%, Return
StringTrimLeft, TargetSection, MenuItem%mY%, 0
StringReplace, TargetSection, TargetSection, %a_space%,,
Gosub, %TargetSection%
Return

```

30.34 Volume On-Screen-Display (OSD) -- by Rajat

This script assigns hotkeys of your choice to raise and lower the master and/or wave volume. Both volumes are displayed as different color bar graphs.

[Download This Script](#) | [Other Sample Scripts](#)(See 13.) | [Home](#)

```

;_____
;_____User Settings_____
; Make customisation only in this area or hotkey area only!!

```

```
; The percentage by which to raise or lower the volume each time:  
vol_Step = 4  
  
; How long to display the volume level bar graphs:  
vol_DisplayTime = 2000  
  
; Master Volume Bar color (see the help file to use more  
; precise shades):  
vol_CBM = Red  
  
; Wave Volume Bar color  
vol_CBW = Blue  
  
; Background color  
vol_CW = Silver  
  
; Bar's screen position. Use -1 to center the bar in that dimension:  
vol_PosX = -1  
vol_PosY = -1  
vol_Width = 150 ; width of bar  
vol_Thick = 12 ; thickness of bar  
  
; If your keyboard has multimedia buttons for Volume, you can  
; try changing the below hotkeys to use them by specifying  
; Volume_Up, ^Volume_Up, Volume_Down, and ^Volume_Down:  
HotKey, #Up, vol_MasterUp      ; Win+UpArrow  
HotKey, #Down, vol_MasterDown  
HotKey, +#Up, vol_WaveUp       ; Shift+Win+UpArrow  
HotKey, +#Down, vol_WaveDown
```

```

;_____
;_____Auto Execute Section_____
; DON'T CHANGE ANYTHING HERE (unless you know what you're doing).

vol_BarOptionsMaster = 1:B ZH%vol_Thickness% ZX0 ZY0 W%vol_Width% CB%vol_CBM%
CW%vol_CW%

vol_BarOptionsWave    = 2:B ZH%vol_Thickness% ZX0 ZY0 W%vol_Width% CB%vol_CBW%
CW%vol_CW%


; If the X position has been specified, add it to the options.
; Otherwise, omit it to center the bar horizontally:
if vol_PosX >= 0
{
    vol_BarOptionsMaster = %vol_BarOptionsMaster% X%vol_PosX%
    vol_BarOptionsWave   = %vol_BarOptionsWave% X%vol_PosX%
}

; If the Y position has been specified, add it to the options.
; Otherwise, omit it to have it calculated later:
if vol_PosY >= 0
{
    vol_BarOptionsMaster = %vol_BarOptionsMaster% Y%vol_PosY%
    vol_PosY_wave = %vol_PosY%
    vol_PosY_wave += %vol_Thickness%
    vol_BarOptionsWave = %vol_BarOptionsWave% Y%vol_PosY_wave%
}

```

```
#SingleInstance  
SetBatchLines, 10ms  
Return  
  
;  
  
vol_WaveUp:  
SoundSet, +%vol_Step%, Wave  
Gosub, vol_ShowBars  
return  
  
vol_WaveDown:  
SoundSet, -%vol_Step%, Wave  
Gosub, vol_ShowBars  
return  
  
vol_MasterUp:  
SoundSet, +%vol_Step%  
Gosub, vol_ShowBars  
return  
  
vol_MasterDown:  
SoundSet, -%vol_Step%  
Gosub, vol_ShowBars  
return  
  
vol_ShowBars:  
; To prevent the "flashing" effect, only create the bar window if it
```

```
; doesn't already exist:

IfWinNotExist, vol_Wave

    Progress, %vol_BarOptionsWave%, , , vol_Wave

IfWinNotExist, vol_Master

{

; Calculate position here in case screen resolution changes while

; the script is running:

if vol_PosY < 0

{

; Create the Wave bar just above the Master bar:

WinGetPos, , vol_Wave_Posy, , , vol_Wave

vol_Wave_Posy -= %vol_Thick%

Progress, %vol_BarOptionsMaster% Y%vol_Wave_Posy%, , , vol_Master

}

else

    Progress, %vol_BarOptionsMaster%, , , vol_Master

}

; Get both volumes in case the user or an external program changed them:

SoundGet, vol_Master, Master

SoundGet, vol_Wave, Wave

Progress, 1:%vol_Master%

Progress, 2:%vol_Wave%

SetTimer, vol_BarOff, %vol_DisplayTime%

return


vol_BarOff:

SetTimer, vol_BarOff, off

Progress, 1:Off

Progress, 2:Off
```

```
return
```

30.35 Window Shading (roll up a window to its title bar) -- by Rajat

This script reduces a window to its title bar and then back to its original size by pressing a single hotkey. Any number of windows can be reduced in this fashion (the script remembers each). If the script exits for any reason, all "rolled up" windows will be automatically restored to their original heights.

[Download This Script](#) | [Other Sample Scripts](#)(See 13.) | [Home](#)

```
; Set the height of a rolled up window here. The operating system
; probably won't allow the title bar to be hidden regardless of
; how low this number is:
ws_MinHeight = 25

; This line will unroll any rolled up windows if the script exits
; for any reason:
OnExit, ExitSub

return ; End of auto-execute section

#z:: ; Change this line to pick a different hotkey.

; Below this point, no changes should be made unless you want to
; alter the script's basic functionality.

; Uncomment this next line if this subroutine is to be converted
; into a custom menu item rather than a hotkey. The delay allows
; the active window that was deactivated by the displayed menu to
; become active again:
;Sleep, 200

WinGet, ws_ID, ID, A
Loop, Parse, ws_IDList, |
```

```

{

IfEqual, A_LoopField, %ws_ID%

{
    ; Match found, so this window should be restored (unrolled):

StringTrimRight, ws_Height, ws_Window%ws_ID%, 0

WinMove, ahk_id %ws_ID%,,,, %ws_Height%

StringReplace, ws_IDList, ws_IDList, |%ws_ID%

return

}

}

WinGetPos,,, ws_Height, A

ws_Window%ws_ID% = %ws_Height%

WinMove, ahk_id %ws_ID%,,,, %ws_MinHeight%

ws_IDList = %ws_IDList%|%ws_ID%

return

}

ExitSub:

Loop, Parse, ws_IDList, |

{
    if A_LoopField = ; First field in list is normally blank.

        continue      ; So skip it.

    StringTrimRight, ws_Height, ws_Window%A_LoopField%, 0

    WinMove, ahk_id %A_LoopField%,,,, %ws_Height%

}

ExitApp ; Must do this for the OnExit subroutine to actually Exit the script.

```

30.36 WinLIRC Client

This script receives notifications from [WinLIRC](#) whenever you press a button on your remote control. It can be used to automate Winamp, Windows Media Player, etc. It's easy to configure. For example, if WinLIRC recognizes a button named "VolUp" on your remote control, create a

label named VolUp and beneath it use the command "SoundSet +5" to increase the soundcard's volume by 5%.

[Download This Script](#) | [Other Sample Scripts](#)(See 13.) | [Home](#)

```
; Here are the steps to use this script:  
;  
; 1) Configure WinLIRC to recognize your remote control and its buttons.  
;  
;     WinLIRC is at http://winlirc.sourceforge.net  
;  
; 2) Edit the WinLIRC path, address, and port in the CONFIG section below.  
;  
; 3) Launch this script. It will start the WinLIRC server if needed.  
;  
; 4) Press some buttons on your remote control. A small window will  
;  
;     appear showing the name of each button as you press it.  
;  
; 5) Configure your buttons to send keystrokes and mouse clicks to  
;  
;     windows such as Winamp, Media Player, etc. See the examples below.  
  
;  
; This script requires AutoHotkey 1.0.38.04 or later.  
;  
; HISTORY OF CHANGES  
;  
; March 2, 2007:  
;  
; - Improved reliability via "Critical" in ReceiveData().  
;  
; October 5, 2005:  
;  
; - Eliminated Winsock warning dialog "10054" upon system shutdown/logoff.  
;  
; - Added option "DelayBetweenButtonRepeats" to throttle the repeat speed.  
  
;  
-----  
;  
; CONFIGURATION SECTION: Set your preferences here.  
;  
-----  
;  
; Some remote controls repeat the signal rapidly while you're holding down  
;  
; a button. This makes it difficult to get the remote to send only a single  
;  
; signal. The following setting solves this by ignoring repeated signals  
;  
; until the specified time has passed. 200 is often a good setting. Set it
```

```
; to 0 to disable this feature.

DelayBetweenButtonRepeats = 200

; Specify the path to WinLIRC, such as C:\WinLIRC\winlirc.exe

WinLIRC_Path = %A_ProgramFiles%\WinLIRC\winlirc.exe

; Specify WinLIRC's address and port. The most common are 127.0.0.1 (localhost) and
; 8765.

WinLIRC_Address = 127.0.0.1

WinLIRC_Port = 8765

; Do not change the following two lines. Skip them and continue below.

Gosub WinLIRC_Init

return

; -----
; ASSIGN ACTIONS TO THE BUTTONS ON YOUR REMOTE
; -----

; Configure your remote control's buttons below. Use WinLIRC's names
; for the buttons, which can be seen in your WinLIRC config file
; (.cf file) -- or you can press any button on your remote and the
; script will briefly display the button's name in a small window.

;

; Below are some examples. Feel free to revise or delete them to suit
; your preferences.

VolUp:

SoundSet +5 ; Increase master volume by 5%. On Vista, replace this line with: Send
{Volume_Up}

return
```

```
VolDown:  
  
SoundSet -5 ; Reduce master volume by 5%. On Vista, replace this line with: Send  
{Volume_Down}  
  
return  
  
  
ChUp:  
  
WinGetClass, ActiveClass, A  
  
if ActiveClass in Winamp v1.x,Winamp PE ; Winamp is active.  
  
    Send {right} ; Send a right-arrow keystroke.  
  
else ; Some other type of window is active.  
  
    Send {WheelUp} ; Rotate the mouse wheel up by one notch.  
  
return  
  
  
ChDown:  
  
WinGetClass, ActiveClass, A  
  
if ActiveClass in Winamp v1.x,Winamp PE ; Winamp is active.  
  
    Send {left} ; Send a left-arrow keystroke.  
  
else ; Some other type of window is active.  
  
    Send {WheelDown} ; Rotate the mouse wheel down by one notch.  
  
return  
  
  
Menu:  
  
IfWinExist, Untitled - Notepad  
{  
    WinActivate  
}  
else  
{
```

```

Run, Notepad

WinWait, Untitled - Notepad

WinActivate

}

Send Here are some keystrokes sent to Notepad.{Enter}

return

; The examples above give a feel for how to accomplish common tasks.

; To learn the basics of AutoHotkey, check out the Quick-start Tutorial
; at http://www.autohotkey.com/docs/Tutorial.htm

;

; -----  

; END OF CONFIGURATION SECTION  

;  

; -----  

; Do not make changes below this point unless you want to change the core
; functionality of the script.

WinLIRC_Init:

OnExit, ExitSub ; For connection cleanup purposes.

;  

; Launch WinLIRC if it isn't already running:  

Process, Exist, winlirc.exe

if not ErrorLevel ; No PID for WinLIRC was found.

{

    IfNotExist, %WinLIRC_Path%  

    {

        MsgBox The file "%WinLIRC_Path%" does not exist. Please edit this script to
specify its location.

        ExitApp
}

```

```
}

Run %WinLIRC_Path%

Sleep 200 ; Give WinLIRC a little time to initialize (probably never needed, just for
peace of mind).

}

; Connect to WinLIRC (or any type of server for that matter):

socket := ConnectToAddress(WinLIRC_Address, WinLIRC_Port)

if socket = -1 ; Connection failed (it already displayed the reason).

ExitApp

; Find this script's main window:

Process, Exist ; This sets ErrorLevel to this script's PID (it's done this way to support
compiled scripts).

DetectHiddenWindows On

ScriptMainWindowId := WinExist("ahk_class AutoHotkey ahk_pid " . ErrorLevel)

DetectHiddenWindows Off

; When the OS notifies the script that there is incoming data waiting to be received,
; the following causes a function to be launched to read the data:

NotificationMsg = 0x5555 ; An arbitrary message number, but should be greater than
0x1000.

OnMessage(NotificationMsg, "ReceiveData")

; Set up the connection to notify this script via message whenever new data has arrived.

; This avoids the need to poll the connection and thus cuts down on resource usage.

FD_READ = 1 ; Received when data is available to be read.

FD_CLOSE = 32 ; Received when connection has been closed.

if DllCall("Ws2_32\WSAAsyncSelect", "UInt", socket, "UInt", ScriptMainWindowId, "UInt",
NotificationMsg, "Int", FD_READ|FD_CLOSE)

{
```

```

    MsgBox % "WSAAsyncSelect() indicated Winsock error " .
DIICall("Ws2_32\WSAGetLastError")

    ExitApp

}

return

ConnectToAddress(IPAddress, Port)
; This can connect to most types of TCP servers, not just WinLIRC.

; Returns -1 (INVALID_SOCKET) upon failure or the socket ID upon success.

{

    VarSetCapacity(wsaData, 32) ; The struct is only about 14 in size, so 32 is
conservative.

    result := DIICall("Ws2_32\WSAStartup", "UShort", 0x0002, "UInt", &wsaData) ;
Request Winsock 2.0 (0x0002)

    ; Since WSAStartup() will likely be the first Winsock function called by this script,
    ; check ErrorLevel to see if the OS has Winsock 2.0 available:

    if ErrorLevel

    {

        MsgBox WSAStartup() could not be called due to error %ErrorLevel%. Winsock 2.0
or higher is required.

        return -1

    }

    if result ; Non-zero, which means it failed (most Winsock functions return 0 upon
success).

    {

        MsgBox % "WSAStartup() indicated Winsock error " .
DIICall("Ws2_32\WSAGetLastError")

        return -1

    }
}

```

```

AF_INET = 2

SOCK_STREAM = 1

IPPROTO_TCP = 6

socket := DllCall("Ws2_32\socket", "Int", AF_INET, "Int", SOCK_STREAM, "Int",
IPPROTO_TCP)

if socket = -1

{
    MsgBox % "socket() indicated Winsock error " .
    DllCall("Ws2_32\WSAGetLastError")

    return -1

}

; Prepare for connection:

SizeOfSocketAddress = 16

VarSetCapacity(SocketAddress, SizeOfSocketAddress)

InsertInteger(2, SocketAddress, 0, AF_INET) ; sin_family

InsertInteger(DllCall("Ws2_32\hton", "UShort", Port), SocketAddress, 2, 2) ; sin_port

InsertInteger(DllCall("Ws2_32\inet_addr", "Str", IPAddress), SocketAddress, 4, 4) ; sin_addr.s_addr

; Attempt connection:

if DllCall("Ws2_32\connect", "UInt", socket, "UInt", &SocketAddress, "Int",
SizeOfSocketAddress)

{
    MsgBox % "connect() indicated Winsock error " .
    DllCall("Ws2_32\WSAGetLastError") . ". Is WinLIRC running?"

    return -1

}

return socket ; Indicate success by returning a valid socket ID rather than -1.

```

```

}

ReceiveData(wParam, lParam)
; By means of OnMessage(), this function has been set up to be called automatically
whenever new data

; arrives on the connection. It reads the data from WinLIRC and takes appropriate action
depending

; on the contents.

{
    Critical ; Prevents another of the same message from being discarded due to
    thread-already-running.

    socket := wParam

    ReceivedDataSize = 4096 ; Large in case a lot of data gets buffered due to delay in
processing previous data.

    VarSetCapacity(ReceivedData, ReceivedDataSize, 0) ; 0 for last param terminates
string for use with recv().

    ReceivedDataLength := DllCall("Ws2_32\recv", "UInt", socket, "Str", ReceivedData,
"Int", ReceivedDataSize, "Int", 0)

    if ReceivedDataLength = 0 ; The connection was gracefully closed, probably due to
    exiting WinLIRC.

        ExitApp ; The OnExit routine will call WSACleanup() for us.

    if ReceivedDataLength = -1

    {
        WinsockError := DllCall("Ws2_32\WSAGetLastError")

        if WinsockError = 10035 ; WSAEWOULDBLOCK, which means "no more data to be
read".

            return 1

        if WinsockError <> 10054 ; WSAECONNRESET, which happens when WinLIRC
closes via system shutdown/logoff.

```

```

; Since it's an unexpected error, report it. Also exit to avoid infinite loop.

MsgBox % "recv() indicated Winsock error " . WinsocketError

ExitApp ; The OnExit routine will call WSACleanup() for us.

}

; Otherwise, process the data received. Testing shows that it's possible to get more
than one line

; at a time (even for explicitly-sent IR signals), which the following method handles
properly.

; Data received from WinLIRC looks like the following example (see the WinLIRC docs
for details):

; 0000000000eab154 00 NameOfButton NameOfRemote

Loop, parse, ReceivedData, `n, `r

{
    if A_LoopField in ,BEGIN,SIGHUP,END ; Ignore blank lines and WinLIRC's
start-up messages.

        continue

    ButtonName = ; Init to blank in case there are less than 3 fields found below.

    Loop, parse, A_LoopField, %A_Space% ; Extract the button name, which is the
third field.

        if A_Index = 3

            ButtonName := A_LoopField

        global DelayBetweenButtonRepeats ; Declare globals to make them available to
this function.

        static PrevButtonName, PrevButtonTime, RepeatCount ; These variables
remember their values between calls.

        if (ButtonName != PrevButtonName || A_TickCount - PrevButtonTime >
DelayBetweenButtonRepeats)

    {
        if IsLabel(ButtonName) ; There is a subroutine associated with this button.

            Gosub %ButtonName% ; Launch the subroutine.

        else ; Since there is no associated subroutine, briefly display which button was
pressed.

```

```

{
    if (ButtonName == PrevButtonName)
        RepeatCount += 1
    else
        RepeatCount = 1
    SplashTextOn, 150, 20, Button from WinLIRC, %ButtonName%
    (%RepeatCount%)
    SetTimer, SplashOff, 3000 ; This allows more signals to be processed
    while displaying the window.

}

PrevButtonName := ButtonName
PrevButtonTime := A_TickCount
}

}

return 1 ; Tell the program that no further processing of this message is needed.

}

SplashOff:
SplashTextOff
SetTimer, SplashOff, Off
return

InsertInteger(pInteger, ByRef pDest, pOffset = 0, pSize = 4)
; The caller must ensure that pDest has sufficient capacity. To preserve any existing
contents in pDest,
; only pSize number of bytes starting at pOffset are altered in it.

```

```
{
    Loop %pSize% ; Copy each byte in the integer into the structure as raw binary data.

        DllCall("RtlFillMemory", "UInt", &pDest + pOffset + A_Index-1, "UInt", 1, "UChar",
pInteger >> 8*(A_Index-1) & 0xFF)

}

ExitSub: ; This subroutine is called automatically when the script exits for any reason.

; MSDN: "Any sockets open when WSACleanup is called are reset and automatically
; deallocated as if closesocket was called."

DllCall("Ws2_32\WSACleanup")

ExitApp
```

30.37 AutoHotkey Tutorial - Launch a program or document (continued)

要使一个程序或文档最大化、最小化或隐藏地启动, 考虑下面的 [hotkey \(热键\)\(See 4.\)](#) 它指定 Win+Z 热键来启动两个记事本实例, 第一个是最大化的, 而第二个是最小化的:

```
#z:::
Run, Notepad, , max
Run, Notepad, , min
return
```

要使一个程序使用一个指定的文件夹作为它的工作目录, 考虑 这个创建一个命令提示符窗口到指定目录的 Win+C 热键:

```
#c:::Run, %comspec% /k, C:\My Documents
```

在上面的例子中, [comspec\(See 9.\)](#) 是一个在一个典型的系统中指向 C:\Windows\system32\cmd.exe 的内置 variable (变量)。

想要传递参数, 直接把它们添加到程序或文档的名字后面——正如这些例子:

```
Run, %comspec% /c dir >"Output File.txt", C:\My Documents
Run, Notepad.exe "C:\My Documents\Address List.txt"
```

```
Run, "%A_AhkPath%(See 9.)" "C:\Scripts\Test Script.ahk" param1 "param2 with  
spaces" param3
```

在上面的第二个和第三个例子中，带有空格的参数被放在引号里，这是一般来说最可靠的做法。相比之下，即使工作目录包含空格，也不需要被放在引号里，正如上面的第一个例子。

某些特定的被称为 **system verbs (系统保留字)** 的词语也是被支持。下面的第一个例子为指定的文件打开资源管理器的属性对话框。第二个例子打印指定的文档。

```
Run, properties "C:\Address List.txt"
```

```
Run, print "C:\Address List.txt"
```

最后，[RunWait\(See 24.5\)](#)设定内置的 [ErrorLevel\(See 30.8\)](#) 变量为它打开的那个程序的退出代码(它还会等候程序终止并关闭)。例如，由于 cmd.exe 在表明有问题发生了，故下面的例子会显示一个非零的 ErrorLevel：

```
RunWait, %comspec% /c dir c:\NonExistent.txt, , hide
```

```
MsgBox, %ErrorLevel%
```

更多关于启动程序和文档的内容，请看 [Run/RunWait\(See 24.5\)](#)。

[返回到指南目录\(See 2.\)](#)