ISSN 1009-3044
Computer Knowledge and Technology 电脑知识与技术
Vol.6, No.36, December 2010,pp.10328-10331

E-mail: xsjl@cccc.net.cn
http://www.dnzs.net.cn
Tel:+86-551-5690963 5690964

基于 VB.NET 实现全局键盘鼠标钩子

吴林[1],刘晔[2]

(1.重庆通信学院 机动作战通信系,重庆 400035;2.重庆通信学院 基础部,重庆 400035)

摘要:文章通过对 Windows 系统钩子技术的研究,利用 VB.NET 的委托类型且不依赖于 DLL 实现了全局键盘鼠标钩子。

关键词:VB.NET;全局;钩子

中图分类号:TP311 文献标识码:A 文章编号:1009-3044(2010)36-10328-04

Implementation of Global Keyboard and Mouse Hook Based on VB.NET

WU Lin[1], LIU Ye[2]

(1.Mobile Warfare Communications Department, Chongqing Institute of Communication, Chongqing 400035, China; 2.Basic Department, Chongqing Institute of Communication, Chongqing 400035, China)

Abstract: Researching on windows hook technology, this paper implement global keyboard and mouse hook by using delegate style of VB.NET and not depending on DLL.

Key words: VB.NET; global; hook

在实际应用中,开发一个应用程序,可能不允许响应回车键和空格键。为此需要截获所有的消息,然后判断。如果是回车或空格按键消息,则屏蔽,即不让这样的消息继续传递。另一种典型的情况如开发一个安装过程,为防止影响安装过程,也需要截获这两类消息[1]。

为了实现这一功能,可以安装一个 Hook 子程,称为"钩子子程"。钩子(Hook),是 Windows 消息处理机制的一个平台,应用程序可以在上面设置子程以监视指定窗口的某种消息,而且所监视的窗口可以是其它进程所创建的。钩子机制允许应用程序截获处理 window 消息或特定事件[2]。

目前很多文章和书籍均大量介绍基于 C/C++或 Delphi 语言来实现钩子,而且全局钩子的实现还必须封装在动态链接库(DLL)中才可以使用[3],实现过程较为繁琐。基于此,文章另辟蹊径,介绍了一种基于 VB.NET 的且不依赖于 DLL 的全局键盘鼠标钩子的实现方法。

1 钩子介绍

1.1 运行机制

钩子实际上是一个处理消息的程序段,通过系统调用,把它挂入系统。每当特定的消息发出,在没有到达目的窗口之前,钩子程序就先捕获该消息,亦即钩子函数先得到控制权。这时钩子函数即可以加工处理(改变)该消息,也可以不作处理而继续传递该消息,还可以强制结束消息的传递。

任何一个钩子都由系统来维护一个指针列表(钩子链表),其指针指向钩子的各个处理函数,对同一事件消息可安装多个钩子处理过程。最近安装的钩子放在链表的开始,最早安装的钩子则放在链表的最后。当钩子监视的消息出现时,操作系统调用链表开始处的第一个钩子处理函数进行处理,即最后加入的钩子优先获得控制权。这里提到的钩子处理函数必须是一个回调函数(call-back function),而且不能定义为类的成员函数,必须定义为普通的 C 函数。钩子会消耗消息处理时间,降低系统性能,只有在必要时才安装钩子,并且在使用后要及时卸载[4]。

1.2 钩子分类

按事件分类,钩子函数的常用类型为键盘钩子和低级键盘钩子、鼠标钩子和低级鼠标钩子,以及外壳钩子(可以监视各种 shell 事件消息)等。比如启动和关闭应用程序时,日志钩子可以记录从系统消息队列中取出的各种事件消息;窗口过程钩子监视从所有系统消息队列发往目标窗口的消息。

按使用范围可分为线程钩子和全局钩子。线程钩子监视指定线程的事件消息;全局钩子监视系统中的所有线程的事件消息。

2 实现步骤

钩子程序的实现过程可分为三个步骤,分别是定义钩子函数、安装钩子及卸载钩子。

2.1 定义钩子函数

钩子函数必须是回调函数,也称为"钩子子程",其函数原型定义如下:

LRESULT CALLBACK HookProc (int nCode,WPARAM wParam,LPARAM lParam)

HookProc 是应用程序自己定义的函数名称。nCode 是钩子代码,钩子进程使用钩子代码去决定是否执行,而钩子代码的值是依靠钩子的种类来定的,每种钩子种类都有他们自己一系列特性的代码。参数 wParam 和 lParam 的值依赖于 Hook 代码。wParam 表示键盘敲打所产生的键盘消息,包含键盘按键的虚拟代码。lParam 则包含了消息细节[5]。

收稿日期:2010-10-25

作者简介:吴林,男,重庆通信学院通信战术教研室,助教。

钩子子程的机制在 VB.NET 中由委托(Delegate)机制[6]实现,在 VB.NET 中钩子子程定义如下:

```
Private Delegate Function HookProc(ByVal nCode As Integer, ByVal wParam As Integer, ByVal lParam As IntPtr) As Integer
Private Function KeyboardHookProc(ByVal nCode As Integer, ByVal wParam As Integer, ByVal lParam As IntPtr) As Integer
...
End Function
```

钩子链表回调钩子子程的过程如下:

```
Private Shared KeyboardHookProcedure As HookProc = New HookProc(AddressOf KeyboardHookProc)
```

2.2 安装钩子

使用 SetWindowsHookEx 函数把应用程序定义的钩子子程安装到钩子链表中去。该函数总是在钩子链表的开头安装钩子子程。当指定类型的钩子监视的事件发生时,系统就调用与这个钩子关联的钩子链表开头的钩子子程。每一个钩子链表中的钩子子程都决定是否把这个事件传递到下一个钩子子程。钩子子程传递事件到下一个钩子子程时需要调用 CallNextHookEx 函数。

使用 VB.NET 编写程序时,函数 SetWindowsHookEx 定义如下:

```
Private Declare Function SetWindowsHookEx Lib "user32" Alias "SetWindowsHookExA" (ByVal idHook As Integer, ByVal lpfn As HookProc, ByVal hMod As IntPtr, ByVal dwThreadId As Integer) As Integer
```

参数 idHook 指定将要安装的钩子子程的类型,实际上是它处理的消息的类型,其取值如表 1 所示。参数 lpfn 是一个委托字段,相当于 C/C++中的函数指针,作用是向函数 SetWindowsHookEx 传递钩子子程。参数 hInstance 是应用程序实例的句柄。参数 dwThreadId 指定与钩子子程相关的线程标识,如果其值为 0,那么安装的钩子子程将与桌面上运行的所有线程都相关。函数 SetWindowsHookEx 成功安装钩子则返回钩子子程的句柄,失败则返回 0[7]。

使用 VB.NET 编写程序时,函数 CallNextHookEx 定义如下:

```
Private Declare Function CallNextHookEx Lib "user32" (ByVal idHook As Integer, ByVal
nCode As Integer, ByVal wParam As Integer, ByVal lParam As IntPtr) As Integer
```

参数 idHook 是由函数 SetWindowsHookEx 返回的钩子子程句柄。参数 nCode 是传给钩子子程的事件代码。参数 wParam 和 lParam 分别是传给钩子子程的参数值,其具体含义与设置的钩子类型有关。函数 CallNextHookEx 返回位于钩子链表中的下一个钩子的句柄,返回值类型视所设置的钩子类型而定[8]。

表 1 部分 idHook 类型标识定义

类型	使用范围	值	说明
WH_CALLWNDPROC	线程	4	在操作系统将消息发送到目标窗口处理过程之前监视该消息
WH_CALLWNDPROCRET	线程	12	监视已被目标窗口过程处理的消息
WH_FOREGROUNDIDLE	线程	11	当应用程序的前台线程即将进入空闲状态时被调用,它有助于在空闲时间内执行低优先级的任务
WH_GETMESSAGE	线程	3	监视发送到消息队列的消息
WH_KEYBOARD	线程	2	监视键盘按键信息
WH_KEYBOARD_LL	系统	13	监视底层的键盘输入事件
WH_MOUSE	线程	7	监视鼠标消息
WH_MOUSE_LL	系统	14	监视底层的鼠标输入事件

2.3 卸载钩子

由于安装钩子对系统的性能有一定的影响,所以在钩子使用完毕后应及时将其卸载以释放其所占资源。释放钩子的函数为 UnhookWindowsHookEx,使用 VB.NET 编写程序时,函数 UnhookWindowsHookEx 定义如下:

```
Private Declare Function UnhookWindowsHookEx Lib "user32" (ByVal idHook As Integer) As Integer
```

释放钩子的过程实现起来比较简单,参数 idHook 表示此前由 SetWindowsHookEx 函数所返回的钩子句柄。函数 UnHookWindowsHookEx 成功释放钩子返回 true,失败返回 false[9]。

3 全局钩子应用实例

文章将以一个全局键盘鼠标钩子的安装和使用的实例来介绍基于 VB.NET 如何实现全局钩子技术。全局键盘钩子安装后将返回当前按键信息,同时安装的全局鼠标钩子将返回当前鼠标所在位置的坐标。

1)新建 VB.NET 项目 SystemHook。

2)引用下列命名空间:

```
Imports System.Reflection
Imports System.Threading
Imports System.ComponentModel
Imports System.Runtime.InteropServices
```

3)在类 SystemHook 的声明部分加入如下声明:

```
#Region "API 声明导入"
    Private Declare Function SetWindowsHookEx Lib "user32" Alias "SetWindowsHookExA" (ByVal idHook As Integer, ByVal lpfn As HookProc, ByVal hMod As IntPtr, ByVal dwThreadId As Integer) As Integer
    Private Declare Function UnhookWindowsHookEx Lib "user32" (ByVal idHook As Integer) As Integer
Private Declare Function CallNextHookEx Lib "user32" (ByVal idHook As Integer, ByVal nCode As Integer, ByVal wParam As Integer, ByVal lParam As IntPtr) As Integer
#End Region
#Region "定义结构体"
    Private Structure MouseHookStruct
        Dim PT As Point
        Dim Hwnd As Integer 'Message.HWnd 属性,获取(或设置)消息的窗口的句柄.
        Dim WHitTestCode As Integer
```

```
        Dim DwExtraInfo As Integer
    End Structure
    Private Structure MsLLHookStruct
        Dim PT As Point
        Dim MouseData As Integer
        Dim Flags As Integer
        Dim Time As Integer
        Dim DwExtraInfo As Integer
    End Structure
    Private Structure KbDllHookStruct
        Dim vkCode As Integer
        Dim ScanCode As Integer
        Dim Flags As Integer
        Dim Time As Integer
        Dim DwExtraInfo As Integer
    End Structure
#End Region
#Region "常量声明"
Private Const WH_MOUSE_LL = 14
Private Const WH_KEYBOARD_LL = 13
    Private Const WM_KEYDOWN = &H100
    Private Const WM_SYSKEYDOWN = &H104
#End Region
#Region "声明委托"
    Private Delegate Function HookProc (ByVal nCode As Integer, ByVal wParam As Integer, ByVal lParam As IntPtr) As Integer ′对
应于回调函数的类型.
#End Region
#Region "声明全局变量"
    Private hMouseHook As Integer = 0 ′指示鼠标全局钩子是否安装成功.
    Private hKeyboardHook As Integer = 0 ′指示键盘全局钩子是否安装成功.
Private Shared MouseHookProcedure As HookProc ′定义委托.
Private Shared KeyboardHookProcedure As HookProc ′定义委托.
#End Region
```

4)为类 SystemHook 添加函数 KeyboardHookProc 和 MouseHookProc 用作回调函数:

```
Private Function KeyboardHookProc(ByVal nCode As Integer, ByVal wParam As Integer, ByVal
lParam As IntPtr) As Integer ′键盘钩子子程.
    Static handled As Boolean = False ′定义静态局部变量.
    If nCode >= 0 Then
        Static MyKeyboardHookStruct As KbDllHookStruct
MyKeyboardHookStruct=DirectCast(Marshal.PtrToStructure(lParam,GetType(KbDllHo
okStruct)), KbDllHookStruct)
If wParam = WM_KEYDOWN OrElse wParam = WM_SYSKEYDOWN Then
Dim e As New _ System.Windows.Forms.KeyEventArgs(MyKeyboardHookStruct.vkCode)
            rmMain.Text = "您输入的是:" & chr(e.KeyValue) ′在窗体标题栏显示字符(此处仅能检测常规按键字符).
            handled = handled Or e.Handled ′是否取消下一个钩子.
        End If
        If handled = True Then′取消或者激活下一个钩子.
Return 1
        Else
            Return CallNextHookEx(hKeyboardHook, nCode, wParam, lParam)
        End If
End If
End Function
Private Function MouseHookProc(ByVal nCode As Integer, ByVal wParam As Integer, ByVal lParam As IntPtr) As Integer ′鼠标钩子
子程.
    If nCode >= 0 Then
        Static mouseHookStruct As MsLLHookStruct
mouseHookStruct=DirectCast(Marshal.PtrToStructure(lParam,GetType(MsLLHookStru      ct)), MsLLHookStruct)
        Static moubut As MouseButtons : moubut = MouseButtons.None ′鼠标按键.
        Static mouseDelta As Integer : mouseDelta = 0 ′滚轮值.
        Static clickCount As Integer : clickCount = 0 ′单击次数.
Dim e As New _ MouseEventArgs(moubut,clickCount,mouseHookStruct.PT.X,mouseHookStruct.PT.Y,mouseDelta)
        frmMain.Text = "X=" & e.X & " Y=" & e.Y′在窗体标题栏显示鼠标位置.
```

```
        End If
        Return CallNextHookEx(hMouseHook, nCode, wParam, lParam) ′激活下一个钩子.
    End Function
```

5)安装键盘鼠标钩子:

```
    Public Sub StartHook (Optional ByVal InstallKeyboardHook As Boolean = True, Optional ByVal InstallMouseHook As Boolean = False)
    If InstallKeyboardHook = True AndAlso hKeyboardHook = 0 Then′注册键盘钩子.
    KeyboardHookProcedure = New HookProc (AddressOf KeyboardHookProc)        hKeyboardHook=SetWindowsHookEx(WH_KEYBOARD_LL,KeyboardHookProcedure, Marshal.GetHINSTANCE(Assembly.GetExecutingAssembly.GetModules()(0)), 0)
        If hKeyboardHook = 0 Then ′检测是否注册完成.
    UnHook(True, False) ′在这里反注册.
            Throw New Win32Exception(Marshal.GetLastWin32Error) ′报告错误.
        End If
        End If
    If InstallMouseHook = True AndAlso hMouseHook = 0 Then′注册鼠标钩子.
    MouseHookProcedure = New HookProc (AddressOf MouseHookProc)        hMouseHook =SetWindowsHookEx (WH_MOUSE_LL, MouseHookProcedure,Marshal.GetHINSTANCE(Assembly.GetExecutingAssembly.GetModules()(0)), 0)
        If hMouseHook = 0 Then
        UnHook(False, True)
            Throw New Win32Exception(Marshal.GetLastWin32Error)
        End If
    End If
    End Sub
```

6)卸载键盘鼠标钩子:

```
    Public Sub UnHook (Optional ByVal UninstallKeyboardHook As Boolean = True, Optional ByVal UninstallMouseHook As Boolean = True, Optional ByVal ThrowExceptions As Boolean = False)
    If hKeyboardHook <> 0 AndAlso UninstallKeyboardHook = True Then′卸载键盘钩子.
    Dim retKeyboard As Integer = UnhookWindowsHookEx(hKeyboardHook)
        hKeyboardHook = 0 ′恢复为初始值.
        If ThrowExceptions AndAlso retKeyboard = 0 Then
    Throw New Win32Exception(Marshal.GetLastWin32Error)
    End If
        End If
        If hMouseHook <> 0 AndAlso UninstallMouseHook = True Then′卸载鼠标钩子.
            Dim retMouse As Integer = UnhookWindowsHookEx(hMouseHook)
            hMouseHook = 0 ′恢复为初始值.
            If ThrowExceptions AndAlso retMouse = 0 Then
                Throw New Win32Exception(Marshal.GetLastWin32Error)
            End If
        End If
    End Sub
```

图 1 全局键盘鼠标钩子运行示意图

7)为窗体类 frmMain 定义一个 SystemHook 类的实例:

```
    Friend MyHK As New SystemHook(True, True) ′表示全局键盘、鼠标钩子均有效。
```

8)运行测试:

将以上代码编译运行,当用户按下按键时,第二个文本框能显示该按键的字符;当用户移动鼠标时,窗体标题栏也能显示鼠标当前所在位置的坐标。最终我们基于 VB.NET 实现了全局键盘钩子和全局鼠标钩子。程序运行效果如图 1 所示。

4 结束语

文章在充分阐述了钩子技术的运行机制和分类的基础上,介绍了钩子技术的实现方法,最终通过一个实例详细的讲解了全局键盘钩子和全局鼠标钩子基于 VB.NET 的实现方法。为读者将全局钩子技术应用于其它方面提供了思路。

参考文献:

[1] 杨友东,汪琛琛.Visual C++程序设计全程指南[M].北京:电子工业出版社,2009:361.
[2] hbtsg.钩子[EB/OL].http://baike.baidu.com/view/802026.htm,2010.
[3] TShengWei.系统钩子[EB/OL].http://baike.baidu.com/view/3673455.htm,2010.
[4] 杨友东,汪琛琛.Visual C++程序设计全程指南[M].北京:电子工业出版社,2009:363.
[5] tecmybook.钩子回调函数[EB/OL].http://hi.baidu.com/tecmybook/blog/item/0110de1d30_a120c3a78669bc.html,2010.
[6] 郝春强.Visual Basic 2005 基础实例与教程[M].北京:中国电力出版社,2007:103-106.
[7] 曾昭琳.SetWindowsHookEx[EB/OL].http://baike.baidu.com/view/1208620.htm,2010.
[8] blookme.CallNextHookEx[EB/OL].http://baike.baidu.com/view/1690990.htm,2008.
[9] M4orz3r.UnhookWindowsHookEx[EB/OL].http://hi.baidu.com/m4orz3r/blog/item/3c31ba_06ad343c7102088182.html,2009.

基于VB.NET实现全局键盘鼠标钩子

作者： 吴林， 刘晔， WU Lin， LIU ye
作者单位： 吴林,WU Lin(重庆通信学院,机动作战通信系,重庆,400035)， 刘晔,LIU ye(重庆通信学院,基础部,重庆,400035)
刊名： 电脑知识与技术
英文刊名： COMPUTER KNOWLEDGE AND TECHNOLOGY
年，卷(期)： 2010,06(36)

参考文献(18条)

1. 杨友东.汪琛琛 Visual C++程序设计全程指南 2009
2. 杨友东;汪琛琛 Visual C++程序设计全程指南 2009
3. hbtsg 钩子
4. hbtsg 钩子
5. TShengWei 系统钩子 2010
6. TShengWei 系统钩子 2010
7. 杨友东.汪琛琛 Visual C++程序设计全程指南 2009
8. 杨友东;汪琛琛 Visual C++程序设计全程指南 2009
9. tecmybook 钩子回调函数 2010
10. tecmybook 钩子回调函数 2010
11. 郝春强 Visual Basic 2005基础实例与教程 2007
12. 郝春强 Visual Basic 2005基础实例与教程 2007
13. 曾昭琳 SetWindowsHookEx 2010
14. 曾昭琳 SetWindowsHookEx 2010
15. blookme CallNextHookEx 2008
16. blookme CallNextHookEx 2008
17. M4Orz3r UnhookWindowsHookEx 2009
18. M4Orz3r UnhookWindowsHookEx 2009