VB窗口的子类化技术及其应用

金培权 （合肥中国科技大学计算机科学技术系 230026）

摘要：VB语言本身的消息处理能力不是很强。对于VB中一类涉及消息处理的应用，可以使用VB窗口子类化技术来实现。本文介绍了VB窗口子类化的概念和技术原理，并给出了两个应用实例。实践证明，适当地应用窗口子类化技术，可以大大增强VB应用程序的功能。

关键词：VB 子类化 消息

1 引言

VB应用程序是基于Microsoft Windows的。由于Windows是多任务多线程的工作方式，因此VB程序运行时是受Windows控制的。具体讲，一个VB程序运行时Windows的进程管理模块就会给它创建一个进程，进程与Windows之间的通信通过消息传递来实现：进程发送一个消息给Windows并等待，Windows处理完此消息后将结果会传给进程。由于Windows支持多线程，因此一个进程可以在内部创建多个线程。

VB程序在运行时，通过对其所含的每个控件都创建一个线程来并行地处理控件的事件。这个线程就是该控件的窗口处理函数。在VB中，只要具有hWnd属性（Windows中有个专门的术语称之为窗口句柄）的控件，在运行时Windows都会创建相应的窗口处理函数。这些窗口处理函数也同样通过消息与Windows进行通信。但这些函数都是Windows自身所带，一般情况下用户无法对其进行控制。虽然大多数情况下，Windows的窗口处理函数都是合适的，如单击窗体的最大化按钮时就最大化窗体，但有的时候Windows的窗口处理结果可能并不是用户所想要的，例如当用户准备在文本框上定制一个弹出菜单时，由于文本框对应的Windows窗口处理函数在接收到“鼠标右击”这个消息时总会弹出一个缺省的菜单，所以此时用户就很难实现自己的功能，另外有时候用户可能想让自己程序最小化时缩到任务拦上……还有很多类似的情况。所有这些问题可以归结到一个问题：如何控制VB与Windows之间的消息处理机制？

可以通过VB窗口的子类化技术来实现VB与Windows消息机制之间的交互。本文第二节主要介绍VB中窗口子类化技术的概念与技术原理，第三节给出了两个应用实例。

2 子类化技术

WINDOWS运行的基础是“消息机制”，所谓的“消息”是一个唯一的值，这个值会被一个窗体或操作系统收到，它能告诉什么事件发生了以及需要采用什么样的动作来响应。这与我们人类的神经系统将感知的信息传递给大脑，而大脑发出指令给我们的身体非常相似。VB应用程序的每一个窗口都具有一个消息句柄，这个机制使得所有发自于WINDOWS操作系统的消息都能被窗口接收到。VB中的窗体、按钮、文本框、图片框等都具有这样的消息句柄。WINDOWS操作系统会跟踪这些消息句柄。消息句柄是Windows窗口类结构中的一个参数。

应用程序为了登记一个窗口类，首先要填写好一个WNDCLASS结构，其中的结构参数lpfnWndProc就是该类窗口函数的地址，接着调用RegisterClass()函数向Windows系统申请登记这个窗口类。这时Windows会为其分配一块内存来存放该类的全部信息，这个内存块称为窗口类内存块。

当应用程序要创建一个属于某一已登记窗口类的窗口时，Windows便为这个窗口分配一块内存，即窗口内存块，用来存放与该窗口有关的专用信息。这些信息一部分来自传递给窗口创建函数CreateWindow()或CreateWindowEx()的参数信息，另一部分则来自所属窗口类的窗口类内存块，其中参数lpfnWndProc便被Windows从窗口类内存块复制到为新创建窗口分配的窗口内存块中。当有消息被发送到这个窗口时，Windows检查该窗口内存块中的窗口函数地址（lpfnWndProc），并调用该地址上的函数来处理这些消息。

所谓窗口子类化，实际上就是改变窗口内存块中的有关参数。由于这种修改只涉及到一个窗口的窗口内存块，因此它不会影响到属于同一窗口类的其他窗口的功

能和表现。窗口子类化中最常见的是修改窗口内存块中的窗口函数地址(lpfnWndProc),使其指向一个新的窗口函数,从而改变原窗口函数的处理方法,改进其功能。其基本步骤如下:

(1) 编写子类化窗口函数。该函数必须为标准的窗口函数格式即:

Function WndProc(ByVal hwnd As Long, ByVal Msg As Long, ByVal wParam As Long, ByVal lParam As Long) As Long

此函数名和参数名都可以自定义。各参数的含义如表1所示:

表1

HWnd	窗口句柄
Msg	窗口消息
Wparam	用户定义
LParam	用户定义

在这个函数中对感兴趣的消息进行处理,而把未处理或者需要原窗口函数进一步处理的消息传送给原窗口函数;

(2) 利用待子类化窗口的句柄hWnd,调用GetWindowLong(hWnd,GWL_WNDPROC)函数获得原窗口函数的地址并保存起来;

(3) 调用SetWIndowLong(hWnd,GWL_WNDPROC,AddressOf WndProc)把窗口函数设置成子类化窗口函数,完成窗口子类化。

通过窗体子类化,可以改变响应消息的顺序,也就是说:可以截获某些感兴趣的消息并用子类化窗口函数进行处理,把其他消息传递到Windows默认的窗口处理函数上而不立即响应。下面的逻辑过程表示了子类化窗口函数的一般形式:

```
Public Function WindowProc(hWnd, Msg, wParam,_ lParam) As
Long
  Select Case Msg '检测消息
    Case  interested_message
      Do something here to handle the message
    Case  other messages
      Pass the message to the old default process of Windows
  End Select
End Function
```

3 实例研究

下面给出了两个实例,以说明如何应用子类化技术解决实际问题。为节省篇幅,下面用到的API函数和常量均不给出声明,详细的声明可以用VB自带的API VIEWER获取。

3.1 自定义文本框的弹出菜单

VB中的文本框控件在运行时单击右键会弹出一个缺省的菜单,即使你自定义了一个弹出式菜单并在文本框的Mouse_Down事件中放置了相应的代码,这个缺省的弹出式菜单也总是会显示。如何屏蔽掉文本框的缺省菜单,显示自定义的弹出式菜单呢?下面就通过使用窗口子类化技术来解决这一问题。

当右击文本框时,Windows传递消息WM_RBUTTONDOWN给默认的窗口处理函数。由于这个默认的窗口处理函数总是会弹出缺省的菜单,所以需要通过子类化技术,使用子类化窗口函数替代文本框默认的窗口处理函数,使得当接收到WM_RBUTTONDOWN消息时,可以弹出自己定义的菜单。下面分步骤进行说明:

(1) 编写子类化窗口函数WND_NewProc。

```
Function WND_NewProc(ByVal hWnd As Long, ByVal Msg
As Long, ByVal wParam As Long, ByVal lParam As Long) As
Long
  '检测消息
  Select Case Msg
    Case WM_RBUTTONDOWN '检测鼠标击键消息,如果
是单击右键
      Form1.PopupMenu Form1! MyPopMenu '弹出自定
义的菜单 MyPopMenu
    Case Else '其他消息仍交给原来的窗口函数处理
      WND_NewProc = CallWindowProc(WND_PrevProc,
hWnd, Msg, wParam,_ lParam)
  End Select
End Function
```

(2) 在程序启动时,加入以下代码,对文本框(txtTest)进行窗口子类化。其中WND_PrevProc保存了文本框原来的窗口处理函数

WND_PrevProc = GetWindowLong(txtTest.hWnd, GWL_WNDPROC) '取原来的窗口处理函数

'设置窗口处理函数为WND_NewProc

SetWindowLong hWnd, GWL_WNDPROC, AddressOf WND_NewProc

(3) 在退出程序时,加入以下代码,恢复文本框原来的窗口处理函数,取消窗口子类化。

If WND_PrevProc = 0 Then Exit Sub

SetWindowLong txtTest.hWnd, GWL_WNDPROC,

WND_PrevProc

以上的例子虽然简单，却体现了VB中窗口子类化的核心思想。如果不使用子类化技术，在VB中是很难实现以上功能的。

3.2 任务栏编程的消息处理

所谓任务拦编程主要指对任务栏最右边的提示区进行编程。Windows系统允许用户在任务栏提示区里放置自己的应用程序图标并定制自己需要的操作。关于如何将应用程序的图标放置到任务拦上，以及修改、删除任务栏图标的技术可以参考文献[2]，这里不加细述。用户将应用程序图标放到任务栏上后，还必须能控制应用程序。例如鼠标左击时恢复到窗口形式，或者鼠标右击时弹出一个菜单等等。但这些功能都是需要用户自己去实现，如果没有相应的处理，当最小化应用程序后，用户就再也无法对程序进行操纵。如何实现对上述的任务栏程序的控制？使用一般的VB编程技术是无法实现的。但借助窗口的子类化技术则可以很容易地实现。

3.2.1 Shell_NotifyIcon函数

该函数给Windows发送添加、修改、删除任务栏提示区图标的消息,Windows根据发送的消息进行相应的处理。该函数具有两个参数，参数意义如下:

(1) 参数dwMessage (ByVal dwMessage As Long)

该参数通知系统进行何种操作，可以取值NIM_ADD、NIM_MODIFY或NIM_DELETE。

(2) 参数pnid (pnid As NOTIFYICONDATA)

存储图标特性数据。该参数是子类化时的关键。NOTIFYICONDATA定义如下：

```
Private Type NOTIFYICONDATA
CbSize As Long  '该数据结构的大小
hWnd As Long  '处理图标通知消息的窗口句柄
uID As Long  '应用程序自定义的图标ID
uFlags As Long  '用来设置uCallbackMessage、hIcon、szTip等三个栏目是否有效，一般取组合NIF_ICON Or NIF_TIP Or NIF_MESSAGE,表示全部有效
uCallbackMessage As Long  '消息编号，将来当使用者在图标上按下鼠标时就会以消息通知子类化窗口函数
hIcon As Long  '图标句柄
szTip As String*64 '提示消息
End Type
```

3.2.2 使用子类化技术处理任务栏的消息

本例中的窗体名为frmCaution。

(1) 编写子类化窗口函数，本函数实现当用鼠标左击或右击任务栏上的图标时弹出菜单mnuaa。

在本例中，子类化窗口函数的四个参数含义如下：

HWnd	窗口句柄
Msg	等于当初调用Shell_NotifyIcon时所设置的uCallbackMessage的值
Wparam	等于当初调用Shell_NotifyIcon时所设置的uID的值
Lparam	等于鼠标消息，例如WM_LBUTTONDOWN（按下鼠标左键）等

```
Function WndProc(ByVal hwnd As Long, ByVal Msg As Long, ByVal wParam As Long, ByVal lParam As Long) As Long
  If Msg = WM_USER + 100 Then  "该消息等于调用Shell_NotifyIcon时定义的uCallbackMessage
    If lParam = WM_LBUTTONDOWN Or lParam = WM_RBUTTONDOWN Then
      '当按下鼠标左键或右键时弹出菜单
      frmCaution.PopupMenu frmCaution.mnuaa
    End If
  End If
  '其他的消息交给系统处理，prevWndProc为原来的窗口消息处理函数的句柄
  WndProc = CallWindowProc(prevWndProc, hwnd, Msg, wParam, lParam)
End Function
```

(2) 在Form_Load过程中加入以下代码，对窗体进行子类化，prevWndProc保存原来的窗口处理函数：

```
prevWndProc = GetWindowLong(Me.hwnd, GWL_WNDPROC) '获取系统确省的窗口消息处理函数句柄
SetWindowLong Me.hwnd, GWL_WNDPROC, AddressOf WndProc '指定WndProc为新的消息处理函数
```

(3) 在Form_Unload过程中取消子类化：

```
If WND_PrevProc <> 0 Then
  SetWindowLong txtTest.hWnd, GWL_WNDPROC, WND_PrevProc
End If
```

以上代码实现了当用户左击或右击任务栏上的应用程序图标时弹出一个自定义的菜单。用户可以在菜单中定义自己想要的功能，如退出、显示版本信息等等。另外也可以加入更多的消息处理，使自己的应用程序的功能更强大。

4 结束语

Visual Basic是一个功能十分强大的开发系统，虽然它在消息收发方面并不十分强大，但还是可以通过子类化技术增强VB对消息的处理能力。但在VB使用子类化技术也存在不少限制：① 子类化函数必须放到标准的模块（Module）中，而不能放在类模块或窗体代码中；② AddressOf运算符只能用于自定义的过程、函数或属性，不能将其用于Declare语句声明的外部函数，也不能用于类型库中的函数；③ 写在AddressOf后面的过程、函数和属性必须与有关的声明和过程在同一个工程中；④ 由于子类化函数要与系统直接交互，所以调试十分困难，而且不能在子类化函数中放置Stop或End语句，否则就会引起崩溃。如果子类化函数本身出现错误，也会致使程序崩溃。由于这些限制，VB子类化技术的消息处理能力无法和VC相比，但VB有着VC不能相比的其他优势，所以如果能在实际开发中充分结合VB自身的强大功能和子类化技术，可以大大增强应用程序的功能。■

参考文献

1 *王国荣，Visual Basic 6.0与Windows API讲座，北京人民邮电出版社，1999。*

2 *金培权，“Windows任务栏编程的实现技术”，微计算机应用，2001，No.4。*