# Sử dụng API trong Visual Basic



NoBi
Quản trị

## Sử dụng API trong Visual Basic

T.Sáu 28/03/2008 11:50 am

**Tên bài viết:** Sử dụng API trong Visual Basic
**Tác giả:** NoBi
**Cấp độ bài viết:** Dành cho người đã có căn bản về VB
**Tóm tắt:** Khái niệm API, Handle, Class name, các ví dụ mẫu

## API là gì?

API là viết tắt của Application Programming Interface (giao diện lập trình ứng dụng). Nó là 1 giao tiếp phần mềm được dùng bởi các ứng dụng khác nhau. Cũng giống như bàn phím là một thiết bị giao tiếp giữa người dùng và máy tính, API là 1 giao tiếp phần mềm chẳng hạn như giữa chương trình và hệ điều hành (HĐH). Bộ API của từng HĐH là khác nhau, làm cho các HĐH khác nhau và thường không tương thích với nhau. Ví dụ những phần mềm trên máy Macintosh không thể chạy được trên máy Windows bởi vì Macintosh và Windows có các API hòan tòan khác nhau.

Windows API quản lý mọi vấn đề làm ra Windows. Tuy nhiên nó đồng thời cũng cung cấp hầu hết các tính năng thông dụng (Open, Save As, Choose Font,...), các thiết lập HĐH, các sự kiện,... Các ứng dụng trên nền Windows dùng Windows API hầu như trong mỗi tác vụ. Thậm chí nếu bạn không dùng API trong khi lập trình thì ngôn ngữ lập trình cũng sẽ gọi các API trong chương trình được tạo ra để quản lý nhiều tác vụ khác nhau.

Hầu hết các hàm API được chứa trong các file DLL (Dynamic Link Library – thư viện liên kết động) chứa trong thư mục hệ thống của Windows. Các file DLL cho phép các chương trình bên ngòai dùng các hàm của chúng.

## Khai báo hàm:

Trước khi 1 hàm API có thể được dùng trong Visual Basic, nó phải được khai báo. Bằng việc khai báo hàm, bạn báo cho Visual Basic biết phải tìm hàm đó ở đâu. Phần khai báo đặc tả tên của hàm, file .dll chứa nó, các tham số hàm và kiểu dữ liệu trả về (nếu có).
Phát biểu *Declare* trong Visual Basic dùng để khai báo 1 hàm. Phát biểu Declare chỉ có thể xuất hiện trong phần khai báo của 1 Form hay 1 Module. Nếu nó xuất hiện trong form, phần khai báo phải là Private, điều này làm cho hàm chỉ truy xuất được trong form. Nếu nó xuất hiện trong 1 module, phần khai báo có thể là Public hoặc Private. Từ khóa Public làm cho hàm có thể được dùng trong tòan bộ chương trình, còn từ khóa Private giới hạn nó chỉ được dùng trong moodule mà thôi.

*Các hàm API có 2 dạng*: hàm (Function) có trị trả về và thủ tục (Sub) không có trị trả về:

Khai báo cho hàm có trị trả về như sau:
**Declare Function function_name Lib "DLL_filename" [Alias "function_alias" (argument_list) As**

**data_type**

Khai báo cho các thủ tục:
**Declare Sub function_name Lib "DLL_filename" [Alias "function_alias" (argument_list)**

**function_name:** Là tên của hàm API. Đây là tên mà Visual Basic tham chiếu đến hàm mỗi khi nó được gọi.

**DLL_filename:** Tên của file .dl chứa hàm. Các file thư viện này phải có đầy đủ tên cùng phần mở rộng, riêng đối với 3 thư viện USER, KERNEL, và GUI thì không cần phải có phần mở rộng. Tên này là một String nên cần phải báo trong dấu "".

*Nếu không khai báo đường dẫn đầy đủ, VisualBasic sẽ tìm lần lượt trong:*
1. Thư mục chứa file thực thi .exe
2. Thư mục hiện hành
3. \Windows\System32
4. \Windows\System
5. \Windows
6. Đường dẫn đã được khai báo trước

*Sau đây là các file thư viện phổ biến :*
Advapi32.dll : Advanced API services library supporting numerous APIs including many security and Registry calls
Comdlg32.dll : Common dialog API library
Gdi32.dll : Graphics Device Interface API library (các hàm giao tiếp thiết bị đồ họa)
Kernel32.dll : Core Windows 32-bit base API support (các hàm của HĐH)
Lz32.dll : 32-bit compression routines
Mpr.dll : Multiple Provider Router library
Netapi32.dll : 32-bit Network API library
Shell32.dll : 32-bit Shell API library (các hàm hệ vỏ Windows)
User32.dll : Library for user interface routines (các hàm giao diện người dùng)
Version.dll : Version library
Winmm.dll : Windows multimedia library

*Bốn thư viện chính của Windows :*
KERNEL32
The main DLL, Kernel32, handles memory management, multitasking of the programs that are running, and most other functions which directly affect how Windows actually runs.
USER32
Windows management library. Contains functions which deal with menus, timers, communications, files and many other non-display areas of Windows.
GDI32
Graphics Device Interface. Provides the functions necessary to draw things on the screen, as well as checking which areas of forms need to be redrawn.
WINMM
Provides multimedia functions for dealing with sound, music, real-time video, sampling and more. This is a 32-bit only DLL. The 16 bit equivalent is called MMSYSTEM.

**function_alias** *(tùy chọn, có hay không cũng được):* Là tên thật sự của hàm lưu trong file .dll. Điều này là quan trọng bởi vì hầu hết các hàm nhận tham số kiểu String đều có 2 phiên bản: 1 phiên bản ANSI chuẩn dùng để xử lý các ký tự không phải Unicode và 1 phiên bản để xử lý ký tự Unicode. Nếu cả 2 phiên bản cùng được chứa trong 1 file .dll thì phiên bản không Unicode sẽ thêm hậu tố A và phiên bản Unicode sẽ thêm hậu tố W. Ví dụ: hàm CompareString có 2 phiên bản là CompareStringA và CompareStringW.

**argument_list:** Danh sách các đối số mà hàm yêu cầu. Xác định có bao nhiêu đối số và kiểu tương ứng của mỗi đối số được truyền cho hàm. Cú pháp như sau:

**[{ByVal | ByRef}] argument_name As data_type, ...**

*argument_name:* tên của đối số.
*data_type:* kiểu dữ liệu riêng của mỗi đối số.

Một số điểm cần lưu ý khi dùng ByVal và ByRef:

- Các String luôn truyền kiểu ByVal.
- Các cấu trúc (structure) đều được truyền bằng ByRef.
- Các array luôn truyền bằng ByRef khi tòan bộ array được truyền cho 1 hàm. Cú pháp của Visual Basic chỉ cho phép bạn truyền phần tử đầu tiên của mảng cho 1 hàm. Tuy nhiên khi bạn truyền bằng ByRef, Visual Basic sẽ hiểu là bạn muốn truyền tòan bộ mảng.
- Các biến kiểu số có thể truyền bằng ByVal hay ByRef tùy theo cách dùng của bạn.

**data_type:** Kiểu dữ liệu mà hàm trả về. Thường là kiểu Long, mắc dù 1 số hàm có thể trả về String. Hàm API chỉ hỗ trợ 1 số giới hạn các kiểu dữ liệu, sau đây là các kiểu dữ liệu có thể dùng trong các hàm API:

- Byte : 1 số nguyên 8-bit
- Integer : 1 số nguyên 16-bit
- Long : 1 số nguyên 32-bit
- String

Ngòai các kiểu dữ liệu trên, bất kỳ cấu trúc (structure) nào của API cũng có thể được dùng.

Để biết cách khai báo hàm API bạn cần khởi động tiện ích **API Text Viewer** được cung cấp kèm theo Visual Basic hoặc download ApiViewer 2004 và API-Guide

Sưu tầm và tổng hợp

Share on Facebook
Share on Twitter
Share on Google+

**NoBi**
Quản trị

## Thế nào là Handle?

T.Sáu 28/03/2008 11:52 am

# Thế nào là Handle - What is a Handle???

A variable that identifies an object; an indirect reference to an operating system resource. In plain English a handle is variable of type long which uniquely identifies any object like forms,desktop,menus or in other words a handle is a unique id for each of these objects. Every window in the Windows operating system is identified by a handle. The desktop window has a handle, a Visual Basic form displayed in an application has a handle, and even the controls on a form, which are themselves actually windows, have handles. You can gather a lot of information about the windows in your application after you get the handle of the window that interests you.

**Handle : (cán) tạm gọi là địa chỉ**
Là một biến kiểu Long có giá trị nhận biết duy nhất dùng để định nghĩa một đối tượng. Và trong Windows thì mỗi đối tượng (control) sẽ được gắn cho 1 địa chỉ riêng, giống như số CMND của mình vậy đó, và khi chúng ta muốn làm việc với đối tượng nào thì phải trỏ tới địa chỉ của đối tượng đó, cái địa chỉ đó được gọi là Handle của mỗi đối tượng. Mỗi cửa sổ trong HĐH Windows thì được định nghĩa bởi một handle. Bạn có thể lấy được tất cả các thông tin về một đối tượng sau khi bạn lấy được handle của nó.

**Bây giờ chúng ta sẽ thử làm cho cửa sổ ứng dụng nhấp nháy (Flashing a Window) :**

1. Tạo 1 project mới trong VB
2. Khai báo hàm API FlashWindow trong phần khai báo tổng quát của Form :

**MÃ: CHỌN HẾT**

```
1. Private Declare Function FlashWindow Lib "user32" Alias "FlashWindow" (ByVal hWnd As Long, ByVal
   bInvert As Long) As Long
```

3. Thêm vào trên Form 1 điều khiển Timer. Đặt thuộc tính Interval cho Timer =10 (Timer sẽ thực thi đoạn code trong 10 mili giây một lần).
4. Double-click vào điều khiển Timer. Nhập vào đoạn code sau :

**MÃ: CHỌN HẾT**

```
1. Private Sub Timer1_Timer()
2. Dim nReturnValue As Long
3. nReturnValue = FlashWindow(Form1.hWnd, True)
4. End Sub
```

5. Chạy thử chương trình, bạn sẽ thấy cửa sổ của ứng dụng nhấp nháy.

Private Declare Function FlashWindow Lib "user32" Alias "FlashWindow" (ByVal hWnd As Long, ByVal bInvert As Long) As Long

Phần khai báo sau từ khóa Lib là thư viện chứa hàm FlashWindow, cụ thể ở đây là User32 DLL.
Alias (bí danh): cho VB biết tên thật sự của hàm bên trong thư viện là gì - cái này có thể khác với cái tên mà chúng ta đã gán cho nó trước từ khóa Lib.
Private Declare Function FlashWindow Lib "user32" (ByVal hWnd As Long, ByVal bInvert As Long) As Long
Trường hợp này thì Alias sẽ có cùng tên với tên hàm API (FlashWindow)

(ByVal hWnd As Long, ByVal bInvert As Long) As Long
Tham số đầu tiên (hWnd) : là handle (cán, địa chỉ) của cửa sổ mà chúng ta muốn nó nhấp nháy.
Tham số thứ hai (bInvert) : bật hay tắt tính năng nhấp nháy. Nếu là True, khi ta gọi câu lệnh FlashWindow(Form1.hWnd, True) thì cửa sổ Form1 sẽ nhấp nháy. Khi muốn trả về trạng thái bình thường thì gọi câu lệnh trên với giá trị là False.

nReturnValue = FlashWindow(Form1.hWnd, True)
nReturnValue : biến lưu trử giá trị trả về khi ta gọi hàm FlashWindow. Hầu hết mỗi hàm API đều trả về con số mã báo lỗi, cho dù bạn không sử dụng gì tới giá trị này nhưng bạn nên gán cho nó 1 biến. Nếu không gán giá trị trả về vô 1 biến, sẽ rất nguy hiểm khi trong đoạn code bạn gọi lại hàm đó lần thứ 2, nó có thể gây treo máy. Nói chung tốt nhất bạn nên lưu trử cái giá trị trả về này vô 1 biến mặc dù sau đó bạn không bao giờ sử dụng tới nó.

**NoBi**
Quản trị



## Hằng số sử dụng trong các hàm API

T.Sáu 28/03/2008 11:53 am

Như chúng ta đã biết, các hằng số dùng để lưu một giá trị không đổi giúp cho chúng ta dể nhớ, tiện lợi cho quá trình viết code.
Vd: Const PI = 3.14
Vậy thì các hằng số dùng trong các hàm API cũng là các hằng số như bình thường vậy thôi. Chỉ khác là những hằng số đó đã được lấy những cái tên "chuẩn chung" để cho tất cả người lập trình như chúng ta khi xem code đều có thể hiểu được liền. Những tên hằng đó được viết tắc ngắn gọn và mang tính gợi nhớ để những người "trong nghề" khi đọc tới thì ít nhiều cũng đoán ra được ý nghĩa của nó. Dỉ nhiên là bạn có

thể đặt lại tên khác cho các hằng số đó miễn sao vẫn giữ nguyên giá trị của các hằng số đó. Nhưng có lẻ chẳng ai làm điều này vì làm như vậy thì chỉ có bạn mới hiểu được mình làm gì ...
Để xem các hằng số trên, bạn sử dụng tiện ích API Text Viewer được cung cấp kèm theo Visual Basic. Chọn Constants, nó sẽ trình bày cho bạn danh sách các hằng số đã được người ta "định nghĩa" sẵn. Ví dụ bạn đọc 1 sourcecode của 1 chương trình nào đó, thấy nó dùng hằng số WM_MOUSEMOVE = &H200, bạn có thể hiểu được rằng đây là hằng số được dùng trong xử lý Window Message(các thông điệp trong Windows) sự kiện di chuyển chuột. Giả sử bây giờ bạn lại không muốn dùng MOUSEMOVE nữa mà muốn dùng sự kiện MOUSEWHEEL, bạn có thể dể dàng tìm thấy hằng số WM_MOUSEWHEEL = &H20A bằng tiện ích API Text Viewer. Như vậy bạn có thấy rằng để tìm được hằng số cần dùng thì tùy thuộc rất nhiều vào kinh nghiệm của mình không ?. Và chỉ có đụng chạm nhiều với nó bạn mới "rút tỉa" ra được những kinh nghiệm cho riêng mình !.

Hằng số được quy ước luôn viết bằng chữ IN HOA

Hãy xét ví dụ sau:
**Ví dụ : làm cho cửa sổ luôn ở trên các cửa sổ khác (always on top)**

```
'Khai báo các hằng số cần thiết
Const HWND_TOPMOST = -1
Const HWND_NOTOPMOST = -2
Const SWP_NOSIZE = &H1
Const SWP_NOMOVE = &H2
Const SWP_NOACTIVATE = &H10
Const SWP_SHOWWINDOW = &H40

'Hàm API cần sử dụng
Private Declare Sub SetWindowPos Lib "User32" (ByVal hWnd As Long, ByVal hWndInsertAfter As Long, ByVal X As Long, ByVal Y As Long, ByVal cx As Long, ByVal cy As Long, ByVal wFlags As Long)
```

**MÃ: CHỌN HẾT**

```
1. Private Sub Command1_Click()
2.         'Set the window position to topmost
3.      SetWindowPos Me.hWnd, HWND_TOPMOST, 0, 0, 0, 0, SWP_NOACTIVATE Or SWP_SHOWWINDOW Or SWP_NOMOVE Or
   SWP_NOSIZE
4. End Sub
```

· hWnd
Handle của cửa sổ cần thực hiện.

· hWndInsertAfter
Gồm 1 trong các giá trị sau :
HWND_BOTTOM : Đưa cửa sổ xuống vị trí cuối cùng

HWND_NOTOPMOST : Đưa cửa sổ lên trên tất cả các cửa sổ notopmost nhưng bên dưới topmost.
HWND_TOP : Đưa cửa sổ lên vị trí trên cùng (top).
HWND_TOPMOST : Đưa cửa sổ lên vị trí cao nhất (topmost), ngay cả khi nó không được kích hoạt (deactived).

· X
Vị trí mới bên trái (left).

· Y
Vị trí mới phía trên (top).

· cx

Độ rộng mới của cửa sổ (width), đơn vị là pixel.

· cy
Độ cao mới của cửa sổ (height), đơn vị là pixel.

· uFlags
Cờ định vị trí và kích thước. Có thể sử dụng kết hợp các tham số :

SWP_NOACTIVATE : không kích hoạt cửa sổ
Does not activate the window. If this flag is not set, the window is activated and moved to the top of either the topmost or non-topmost group (depending on the setting of the hWndInsertAfter parameter).
SWP_NOMOVE : giữ nguyên vị trí hiện tại
Retains the current position (ignores the X and Y parameters).
SWP_NOSIZE : giữ nguyên kích thước hiện tại
Retains the current size (ignores the cx and cy parameters).
SWP_SHOWWINDOW : hiển thị cửa sổ
Displays the window.

Để có những hướng dẫn chi tiết hơn, các bạn vô http://www.allAPI.net download về chương trình API-Guide và APIViewer, đây là 2 chương trình rất cần thiết cho những ai muốn tìm hiểu về API.

**NoBi**
Quản trị

## Ví dụ mẫu

T.Sáu 28/03/2008 11:54 am

# Lấy handle của một đối tượng

Chúng ta hãy cùng khảo sát hàm **WindowFromPoint** dùng để lấy handle của đối tượng ở vị trí được chỉ định (xPoint, yPoint).

```
Declare Function WindowFromPoint Lib "user32" (ByVal xPoint As Long, ByVal yPoint As Long) As Long

Declare Function GetCursorPos Lib "user32" (lpPoint As POINTAPI) As Long
```

Và đi cặp với nó là :

**GetCursorPos** : lấy vị trí hiện tại của con trỏ. Vị trí này được lưu giữ trong lpPoint, có 2 giá trị là tọa độ X và Y (kiểu Long). Vì vậy để lấy được giá trị của biến lpPoint, chúng ta cần phải có 1 biến có cùng cấu trúc (structure) kiểu dữ liệu với nó.

**MÃ: CHỌN HẾT**

```
1.
2. Private Type POINTAPI
3.     X As Long
4.     Y As Long
5. End Type
6.
7. Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
8. Dim Pt As POINTAPI, mWnd As Long
9.
10.     'Lấy vị trí hiện tại của chuột
11.     GetCursorPos Pt
12.
13.     'Lấy handle của cửa sổ tại vị trí chuột
```

**NoBi**
Quản trị



## Class name là gì?

T.Sáu 28/03/2008 11:55 am

# Class name ???

Lần trước chúng ta đã được giới thiệu hàm API WindowFromPoint để lấy handle của 1 cửa sổ bất kỳ, lần này tui xin giới thiệu với các bạn hàm **FindWindow**. Nhưng trước hết chúng ta cùng tìm hiểu Class name là gì ?.

**Class name:** là tên lớp đại diện cho 1 số cửa sổ (hay đối tượng). Ví dụ như class name của tất cả cửa sổ trình soạn thảo Notepad là "Notepad"; của các cửa sổ trong CommonDialog là "#32770"; của màn hình ngoài Desktop là "SysListView32"; của các nút lệnh trong cửa sổ Msgbox là "Button"; của label hiển thị thông báo trong Msgbox là "Static" .....
Và để lấy Class name của 1 cửa sổ (hay đối tượng), ta dùng hàm API **GetClassName**:

```
Declare Function GetClassName Lib "user32" Alias "GetClassNameA" (ByVal hwnd As Long, ByVal lpClassName As String, ByVal nMaxCount As Long) As Long
```

Lấy tên lớp của cửa sổ
hwnd : handle của cửa sổ (hay đối tượng) cần lấy tên lớp
lpClassName : biến nhận lấy chuỗi tên lớp
nMaxCount : qui định chiều dài của tên lớp, nếu tên lớp dài hơn thì nó sẽ bị cắt bớt

---

Bây giờ hãy tìm hiểu thêm với hàm **FindWindow** nhé :

```
Declare Function FindWindow Lib "user32" Alias "FindWindowA" (ByVal lpClassName As String, ByVal lpWindowName As String) As Long
```

Lấy handle của cửa sổ có tên (title) hoặc class name trong chuỗi được chỉ định.

lpClassName : tên lớp (Class name) cần lấy handle
lpWindowName : tiêu đề (title) của cửa sổ cần lấy handle

**MÃ: CHỌN HẾT**

```
1. Vd 1: lấy handle của cửa sổ soạn thảo văn bản Notepad
2. Private Sub Command1_Click()
3. Dim n As Long
4.     n = FindWindow("Notepad", vbNullString )
5.     MsgBox "handle = " & n
6. End Sub
```

Nếu chúng ta có nhiều cửa sổ ứng dụng Notepad đang mở thì n sẽ lần lượt là handle của từng cửa sổ, giá trị n sẽ là handle của cửa sổ được tìm thấy cuối cùng. (vbNullString : không cho biết tên cụ thể, hàm FindWindow sẽ tìm hết tất cả cửa sổ có class name là Notepad)

**MÃ: CHỌN HẾT**

```
1. Vd 2: lấy handle của cửa sổ có tiêu đề là Untitled - Notepad
2. Private Sub Command1_Click()
3. Dim n As Long
4.     n = FindWindow(vbNullString, "Untitled - Notepad")
```

```
5.      MsgBox "handle = " & n
6. End Sub
```

Lần này hàm FindWindow sẽ trả về handle của tất cả cửa sổ nào có tiêu đề là Untitled – Notepad. (vbNullString : không cho biết class name cụ thể, hàm FindWindow sẽ tìm hết trong tất cả các cửa sổ có trên màn hình)
Bạn thử tạo 1 Form có Caption = "Untitled – Notepad". Khi đó n sẽ = handle của Form đó.
Bật chương trình soạn thảo Notepad. Khi đó n sẽ = handle của cửa sổ chương trình Notepad.

**MÃ: CHỌN HẾT**

```
1. Vd 3: Lấy hanlde của trình soạn thảo Notepad có tiêu đề là caulacbovb.txt - Notepad
2. Private Sub Command1_Click()
3. Dim n As Long
4.      n = FindWindow("Notepad", "caulacbovb.txt - Notepad")
5.      MsgBox "handle = " & n
6. End Sub
```

Lần này rỏ ràng bạn thấy là chúng ta có 2 ràng buộc : chỉ lấy handle của trình soạn thảo Notepad nào có tiêu đề thỏa mãn yêu cầu.
Bạn tạo 1 file caulacbovb.txt
Tạo thêm vài file .txt khác nữa, tạo 1 Form có Caption = "caulacbovb. txt - Notepad"
Mở tất cả mấy file đó, bây giờ n sẽ chỉ là handle của cửa sổ chương trình Notepad có tiêu đề là caulacbovb. txt - Notepad mà thôi.

Qua 3 ví dụ trên, bạn đã thấy rỏ cách làm việc của hàm FindWindow, nhưng nó chỉ tìm lấy handle của cửa sổ cha mà thôi. Vậy muốn lấy handle của các nút lệnh, label hiển thị thông báo trong Msgbox để Việt hóa nó thì ta phải làm sao ?. Mời bạn tham khảo hàm **FindWindowEx**

Đoạn code bên dưới demo cho việc lấy tên lớp của 1 cửa sổ. Bạn nhập chính xác tiêu đề của 1 cửa sổ vào, chương trình sẽ trả lại tên lớp của cửa sổ đó:

**MÃ: CHỌN HẾT**

```
1. Private Declare Function FindWindow Lib "user32" Alias "FindWindowA" (ByVal lpClassName As String,
   ByVal lpWindowName As String) As Long
2. Private Declare Function GetClassName Lib "user32" Alias "GetClassNameA" (ByVal hwnd As Long, ByVal
   lpClassName As String, ByVal nMaxCount As Long) As Long
3.
4. Const gcClassnameMSWord = "OpusApp"
5. Const gcClassnameMSExcel = "XLMAIN"
6. Const gcClassnameMSIExplorer = "IEFrame"
7. Const gcClassnameMSVBasic = "wndclass_desked_gsk"
8. Const gcClassnameNotePad = "Notepad"
9. Const gcClassnameMyVBApp = "ThunderForm"
10.
11. Private Sub Form_Load()
```

**NoBi**
Quản trị


## Ví dụ mẫu

T.Sáu 28/03/2008 11:56 am

# FindWindowEx

Declare Function FindWindowEx Lib "user32" Alias "FindWindowExA" (ByVal hWnd1 As Long, ByVal hWnd2 As Long, ByVal lpsz1 As String, ByVal lpsz2 As String) As Long

Hàm **FindWindowEx** giống với hàm **FindWindow**, chỉ khác ở chổ là cho lấy handle của "cửa sổ con" (các đối tượng như nút lệnh, label... trong 1 cửa sổ thì được gọi là cửa sổ con)

- hWnd1 : (hwndParent ) handle của cửa sổ cha. Nếu nó = Null thì FindWindowEx sẽ lấy cửa sổ Desktop làm cửa sổ cha và bắt đầu tìm các cửa sổ con trong cửa sổ Desktop (tức là các cửa sổ có trên màn hình)
- hWnd2 : (hwndChildAfter ) chỉ định cửa sổ con mà FindWindowEx sẽ bắt đầu tìm từ cửa sổ kế đó trở đi. Nếu nó = Null thì sẽ bắt đầu tìm từ cửa sổ con đầu tiên
  (Note that if both hwndParent and hwndChildAfter are NULL, the function searches all top-level windows).
- lpsz1 : (lpszClass) chỉ định class name của cửa sổ con cần tìm
- lpsz2 : (lpszWindow) chỉ định caption của cửa sổ con cần tìm

Bạn hãy chạy chương trình nào có xuất hiện hộp thoại Msgbox hay Dialogbox (vì 2 cái này có chung classname). Chạy chương trình bên dưới, bạn sẽ thấy sự thay đổi ...

**MÃ: CHỌN HẾT**

```
Private Declare Function FindWindowEx Lib "user32" Alias "FindWindowExA" (ByVal hWnd1 As Long, ByVal hWnd2 As Long, ByVal lpsz1 As String, ByVal lpsz2 As String) As Long
Private Declare Function FindWindow Lib "user32" Alias "FindWindowA" (ByVal lpClassName As String, ByVal lpWindowName As String) As Long
Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hwnd As Long, ByVal wMsg As Long, ByVal wParam As Long, lParam As Any) As Long
Private Const WM_SETTEXT = &HC
Dim hDlg As Long, hButton As Long

Private Sub Command1_Click()
    'Lấy handle của cửa sổ Dialog bất kỳ có trên màn hình
    hDlg = FindWindow("#32770", vbNullString)
    'Đổi tiêu đề của Dialog đó
```

Nếu bạn chỉ muốn đổi tên nút lệnh OK không thôi thì dùng :

**MÃ: CHỌN HẾT**

```
hButton = FindWindowEx(hDlg, ByVal 0&, "Button", "OK")
SendMessage hButton, WM_SETTEXT, 0&, ByVal "Dong y"
```

Tương tự bạn có thể tùy ý thay đổi tên nút lệnh hay label nào mà mình muốn ...

Các bạn có thể tham khảo thêm sourcecode của các chương trình Việt hóa. Vì cũng dùng cách này để đổi tên tiếng Anh thành tiếng Việt.

Để hiểu rỏ classname là gì, mời bạn xem lại bài viết trên.

Chúc các bạn thành công !!!.

**NoBi**
Quản trị



## Ví dụ tổng hợp kết thúc bài viết

T.Sáu 28/03/2008 11:58 am

# Lấy handle, classname và text của 1 đối tượng

Chép đoạn code sau vào Form, thêm vào 3 TextBox. Nhấn F5 để chạy. Click chuột vào Form và giữ nguyên, kéo chuột tới cửa sổ hoặc control bất kỳ, thả chuột ra, bạn sẽ lấy được handle, classname và text của nó.

**MÃ: CHỌN HẾT**

```
Option Explicit

Private Const WM_GETTEXT = &HD
Private Const WM_GETTEXTLENGTH = &HE

Private Type POINTAPI
    X As Long
    Y As Long
End Type

Private Declare Function WindowFromPoint Lib "user32" (ByVal xPoint As Long, ByVal yPoint As Long) As Long
Private Declare Function GetCursorPos Lib "user32" (lpPoint As POINTAPI) As Long
```

## Kiểm tra tất cả cửa sổ chương trình Notepad hiện có, cái nào đang Minimize thì hiện nó lên

Mở thử vài cửa sổ Notepad, thu nhỏ chúng rồi chạy đoạn code bên dưới:

**MÃ: CHỌN HẾT**

```
Private Declare Function FindWindowExA Lib "user32" (ByVal hWnd1 As Long, ByVal hWnd2 As Long, ByVal lpsz1 As String, ByVal lpsz2 As String) As Long
Private Declare Function ShowWindow Lib "user32" (ByVal hwnd As Long, ByVal nCmdShow As Long) As Long
Private Declare Function IsIconic Lib "user32" (ByVal hwnd As Long) As Long
Private Const SW_NORMAL = 1

Private Sub Command1_Click()
    Dim hCurrent As Long
    Do
        hCurrent = FindWindowExA(0&, hCurrent, vbNullString, "Untitled - Notepad")
        If IsIconic(hCurrent) <> 0 Then ShowWindow hCurrent, SW_NORMAL
    Loop Until hCurrent = 0
```

**truongphu**
VIP



## 55 hàm API liên quan đến cửa sổ

T.Sáu 20/03/2009 1:15 pm

http://congdong.saccauvong.net/showthread.php/c-c-h-m-api-li-n-6696.html
Dưới đây là nội dung Copy từ địa chỉ trên. Không thấy tên tác giả

### Các hàm API liên quan đến cửa sổ

### Phần 1: Các hàm API liên quan đến cửa sổ

Để xem xét quan hệ của một cửa sổ (Tạm gọi là cửa sổ khai báo) với các cửa sổ khác ta nghiên cứu các mối quan hệ sau:

**1. Declare Function AnyPopup Lib "user32" Alias "AnyPopup" () As Long**

Công dụng: Đưa ra chỉ số cửa sổ popup hiện đang tồn tại trên màn hình.
Trị trả về: Integer ~ True (Khác zero) nếu có cửa sổ popup.

**2. Declare Function AdjustWindowRect Lib "user32" Alias "AdjustWindowRect" (lpRect As RECT, ByVal dwStyle As Long, ByVal bMenu As Long) As Long**

**3. Declare Function AdjustWindowRectEx Lib "user32" Alias "AdjustWindowRectEx" (lpRect As RECT, ByVal dsStyle As Long, ByVal bMenu As Long, ByVal dwEsStyle As Long) As Long**

Công dụng: Điều chỉnh cửa sổ khi có vùng làm việc client (Không tính kích thước của thanh tiêu đề, đường viền và các phần thêm) được khai báo, khi biết kiểu cửa sổ.
Tham số kèm:
LpRect Hình chữ nhật chứa vùng làm việc client.
DwStyle Kiểu cửa sổ.
BMenu Đưa giá trị True (Khác zero) nếu cửa sổ có trình đơn
DwEsStyle kiểu cửa sổ mở rộng.

**4. Declare Function ArrangeIconicWindows Lib "user32" Alias "ArrangeIconicWindows" (ByVal hwnd As Long) As Long**

Công dụng: Xếp các biểu tượng cửa sổ trong một cửa sổ chứa (Mức Parent).
Trị trả về: Integer chiều cao của hàng biểu tượng. Zero nếu thất bại.
Tham số kèm:
HWnd Cán của cửa sổ chứa (Mức Parent).

**5. Declare Function BeginDeferWindowPos Lib "user32" Alias "BeginDeferWindowPos" (ByVal nNumWindows As Long) As Long**

Công dụng: Bắt đầu xây dựng danh sách vị trí các cửa sổ mới thành cấu trúc bản đồ nội bộ chứa vị trí các cửa sổ.
Trị trả về: Integer - cán của cấu trúc bản đồ. Zero nếu thất bại.
Tham số kèm:
NNum Windows Số cửa sổ ban đầu để cấp phát chỗ trống.

**6. Declare Function DeferWindowPos Lib "user32" Alias "DeferWindowPos" (ByVal hWinPosInfo As Long, ByVal hwnd As Long, ByVal hWndInsertAfter As Long, ByVal x As Long, ByVal y As Long, ByVal cx As Long, ByVal cy As Long, ByVal wFlags As Long) As Long**

Công dụng: Đinh nghĩa vị trí của cửa sổ mới qua cửa sổ khai báo và đưa vào cấu trúc bản đồ nội bộ chứa vị trí các cửa sổ.
Trị trả về: Integer - Cán mới đối với cấu trúc bản đồ chứa thông tin cập nhật vị trí. Zero nếu thất bại.
Tham số kèm:
HWinPosInfo Cán của cấu trúc bản đồ.
HWnd Cửa sổ cần định vị.
HWndInsertAfter Cán cửa sổ mà cửa sổ hWnd đặt sau nó trong danh sách. Nó có thể là một trong các hằng sau:
HWnd_BOTTOM: Đặt về cuối danh sách.
HWnd_TOP: Đặt cửa sổ ở đầu danh sách
HWnd_TPMOST: Đặt cửa sổ ở đầu danh sách lên trên cùng nhìn thấy được.
X Hoành độ của cửa sổ hWnd theo toạ độ của cửa sổ chứa (Mức Parent) nó.
Y Tung độ của cửa sổ hWnd theo toạ độ cửa sổ chứa (Mức Parent) nó.
cx Chiều rộng cửa sổ mới.
cy Chiều cao cửa sổ mới.
Flags Một số nguyên là một trong các hằng sau:
SWP_DRAWFRAME: Vẽ khung bao quanh cửa sổ.
SWp-HIDEWINDOW: Giấu cửa sổ.
SWP_NOACTIVE: Không kích hoạt cửa sổ.
SWP_NOMOVE: Giữ nguyên vị trí hiện tại.
SWP_NOREDRAW: Không vẽ lại tự động.
SWp_NOSIZE: Giữ nguyên kích thước.
SWp_NOZORDER: Giữ nguyên vị trí hiện hành trong danh sách.

**7. Declare Function SetWindowPos Lib "user32" Alias "SetWindowPos" (ByVal hwnd As Long, ByVal hWndInsertAfter As Long, ByVal x As Long, ByVal y As Long, ByVal cx As Long, ByVal cy As Long, ByVal wFlags As Long) As Long**

Công dụng: Thiết đặt vị trí và trạng thái cửa sổ.
Tham số kèm:
HWnd Cán của cửa sổ cần định vị
HWndInsertAfter Như hàm trên.

**8. Declare Function EndDeferWindowPos Lib "user32" Alias "EndDeferWindowPos" (ByVal hWinPosInfo As Long) As Long**

Công dụng: Cập nhật các vị trí và tình trạng của tất cả các cửa sổ.
Tham số kèm:
HWinPosInfo Cán của cấu trúc bản đồ lấy từ lệnh DerefWindowPos gần nhất.

**9. Declare Function BringWindowToTop Lib "user32" Alias "BringWindowToTop" (ByVal hwnd As Long) As Long**

Công dụng: Chuyển cửa sổ lên đầu danh sách làm lộ ra nếu bị khuất.
Tham số kèm:
HWndCán của cửa sổ cần tác động.

**10. Declare Function ChildWindowFromPoint Lib "user32" Alias "ChildWindowFromPoint" (ByVal hWnd As Long, ByVal xPoint As Long, ByVal yPoint As Long) As Long**

**11. Declare Function ChildWindowFromPoint Lib "user32" Alias "ChildWindowFromPoint" (ByVal hWndParent As Long, ByVal pt As POINTAPI) As Long**

Công dụng: Lấy cán của cửa sổ con (Mức Child) khi đưa điểm của cửa sổ chứa (Mức Parent) nó.
Trị trả về: Integer - Cán của cửa sổ con (Mức Child) đầu tiên thoả mãn. Nếu không thấy cửa sổ con (Mức Child) nào trả về cán của cửa sổ chứa (Mức Parent). Zero nếu điểm nằm ngoài cửa sổ chứa (Mức Parent).
Tham số kèm:
HWnd Cán của cửa sổ chứa (Mức Parent).
Pt Trị của điểm.
XPoint Hoành độ của điểm.
YPoint Tung độ của điểm.

**12. Declare Function ClientToScreen Lib "user32" Alias "ClientToScreen" (ByVal hwnd As Long, lpPoint As POINTAPI) As Long**

Công dụng: Chuyển toạ độ theo cửa sổ sang toạ độ theo màn hình.
Tham số kèm:
HWnd Cán của cửa sổ làm căn cứ xác định toạ độ.
LpPoint Ddieemr tính theo toạ độ cửa sổ

---

**o0o--truongphu--o0o**

.....
Ghé thăm:
**Chuyện Linh Tinh**

**truongphu**
VIP

## 55 hàm API liên quan đến cửa sổ

T.Sáu 20/03/2009 4:00 pm

**13. Declare Function CloseWindow Lib "user32" Alias "CloseWindow" (ByVal hwnd As Long) As Long**
**Công dụng: Thu nhỏ cửa sổ.**

Tham số kèm:
HWnd Cán của cửa sổ cần thu.

**14. Declare Function CopyRect Lib "user32" Alias "CopyRect" (lpDestRect As RECT, lpSourceRect As RECT) As Long**

Công dụng: Sao nội dung hình chữ nhật.
Tham số kèm:
lpDestRect Hình chữ nhật đích sẽ nhận kết quả.
LpSourceRect Hình chữ nhật nguồn bị copy.

**15. Declare Function DestroyWindow Lib "user32" Alias "DestroyWindow" (ByVal hwnd As Long) As Long**

Công dụng: Phá huỷ cửa sổ (Kể cả các cửa sổ con (Mức Child) của nó).
Trị trả về: Integer khác 0 sẽ thành công. Zero nếu thất bại.
Tham số kèm:
HWnd Cán của cửa sổ sẽ phá huỷ.

**16. Declare Function EnableWindow Lib "user32" Alias "EnableWindow" (ByVal hwnd As Long, ByVal fEnable As Long) As Long**

Công dụng: Cho hiệu lực hay vô hiệu hoá mọi dữ liệu nhập vào cửa sổ từ bàn phím hoặc chuột.
Trị trả về: Integer True (Khác zero) nếu trước đó cửa sổ được phép. Zero nếu bị vô hiệu hoá.
Tham số kèm:
HWnd Cán của cửa sổ
FEnable Giá trị logic. Nếu là True, thì Window sẽ có hiệu lực Enable. Còn False, sẽ không có hiệu lực Disable.

**17. Declare Function EnumChildWindows Lib "user32" Alias "EnumChildWindows" (ByVal hWndParent As Long, ByVal lpEnumFunc As Long, ByVal lParam As Long) As Long**

Công dụng: Liệt kê các cửa sổ con (Mức Child) của một cửa sổ chứa (Mức Parent). Phải có Custom Control CBK.VBX mới sử dụng được.
Trị trả về: Integer True (Khác zero) nếu thành công. Zero nếu thất bại.
Tham số kèm:
HWndParent Cán của cửa sổ chứa (Mức Parent) cần liệt kê
LpEnumFunc Biến trỏ chỉ đến hàm để gọi đối với mỗi cửa sổ con (Mức Child). Sử dụng tính chất ProcAddress của Custon Control CBK.VBX để nhận hàm biến trỏ (function pointer) để gọi lại (callbacks).
LParam Trị chuyển đến cho sự kiện EnumWindows của Custom Control trong lúc liệt kê. Ý nghĩa của trị này do lập trình viên xác định.

**18. Declare Function EnumWindowStations Lib "user32" Alias "EnumWindowStationsA" (ByVal lpEnumFunc As Long, ByVal lParam As Long) As Long**

Công dụng: Liệt kê danh sách cửa sổ cấp trên, chứa cửa sổ khai báo. Phải có Custom Control CBK.VBX mới sử dụng được.
Trị trả về: Integer True (Khác zero) nếu thành công.
Tham số kèm:
LpEnumFuncBiến trỏ chỉ đến hàm để gọi đối với mỗi cửa sổ con (Mức Child). Sử dụng tính chất ProcAddress của Custon Control CBK.VBX để nhận hàm biến trỏ (function pointer) để gọi lại (callbacks)
LParam Trị chuyển đến cho sự kiện EnumWindows của Custom Control trong lúc liệt kê. Ý nghĩa của trị này do lập trình viên xác định.
**19. Declare Function EqualRect Lib "user32" Alias "EqualRect" (lpRect1 As RECT, lpRect2 As**

**RECT) As Long**

Công dụng: So sánh 2 cấu trúc hình chữ nhật.
Trị trả về: Integer True (Khác zero) Nếu các toạ độ góc trái trên và góc phải dưới của 2 hình bằng nhau. Zero nếu khác.
Tham số kèm:
LpRec1, lpRec2: Hai hình chữ nhật cần so sánh.

**20. Declare Function FindWindow Lib "user32" Alias "FindWindowA" (ByVal lpClassName As String, ByVal lpWindowName As String) As Long**

Công dụng: Tìm cửa sổ đầu tiên trong danh sách cửa sổ thoả mãn điều kiện.
Trị trả về: Integer - Cán của cửa sổ thoả mãn. Zero nếu không có cửa sổ nào.
Tham số kèm:
LpClassName Biến trỏ chỉ đến chuỗi kết thúc bằng null chứa tên lớp đối tượng đối với cửa sổ. Nếu bằng zero chấp nhận bất cứ lớp nào.
LpWindowName Biến trỏ chỉ đến chuỗi kết thúc bằng null chứa tên tiêu đề cửa sổ. Nếu bằng 0, chấp nhận bất cứ tiêu đề nào.

**21. Declare Function FindWindowEx Lib "user32" Alias "FindWindowExA" (ByVal hWnd1 As Long, ByVal hWnd2 As Long, ByVal lpsz1 As String, ByVal lpsz2 As String) As Long**

Công dụng: Tìm cửa sổ đầu tiên trong danh sách cửa sổ thoả mãn điều kiện.
Trị trả về: Integer - Cán của cửa sổ thoả mãn. Zero nếu không có cửa sổ nào.
Tham số kèm:
hwndParent
Cán cuả cửa sổ chứa (Cấp Parent) có các cửa sổ con để tìm.
Nếu hwndParent là NULL, hàm sẽ sử dụng desktop như cửa sổ chứa parent. Hàm sẽ tìm trong số các cửa sổ là cửa sổ con (Cấp Child) của desktop.
Windows 2000 trở lên: Nếu hwndParent là HWND_MESSAGE, hàm sẽ tìm tất cả cửa sổ dạng message-only windows.
hwndChildAfter
Là cán của cửa sổ con (cấp child). Tìm kiếm bắt đầu từ cửa sổ con kế tiếp theo thứ tự trục Z. Cửa sổ con phải là một cấp kế tiếp của hwndParent, không thể cấp thấp hơn.
Nếu hwndChildAfter là NULL, tìm kiếm sẽ bắt đầu với cửa sổ con đầu tiên(Cấp child) của hwndParent.
Nhớ rằng nếu cả hai hwndParent và hwndChildAfter là NULL, hàm sẽ tìm tất cả mức top của dạng message-only windows.
lpszClass Lớp cần tìm kiếm.
lpszWindow
Tiêu đề của cửa sổ cần tìm. Nếu là NULL, Tìm tất cả.

**22. Declare Function FlashWindow Lib "user32" Alias "FlashWindow" (ByVal hwnd As Long, ByVal bInvert As Long) As Long**

Công dụng: Chiếu sáng cửa sổ, ngay cả khi nó chưa được kích hoạt (inactive)
Trị trả về: Integer True (Khác zero) nếu cửa sổ đã được kích hoạt trước khi gọi.
Tham số kèm:
HWnd Cán của cửa sổ cần chiếu sáng.
BInvert Integer - True (Khác zero) nếu bật, False để quay lại trạng thái trước

**23. Declare Function GetActiveWindow Lib "user32" Alias "GetActiveWindow" () As Long**

Công dụng: Nhận cán của cửa sổ đang kích hoạt.
Trị trả về: Integer - Cán của cửa sổ đang kích hoạt. Zero nếu không có.

**24. Declare Function GetClassInfo Lib "user32" Alias "GetClassInfoA" (ByVal hInstance As Long, ByVal lpClassName As String, lpWndClass As WNDCLASS) As Long**
Công dụng: Nhận bản sao cấu trúc Wndclass chứa thông tin về lớp khai báo.

Trị trả về: Integer - True (Khác zero) khi thành công. Zero nếu không thấy lớp thoả mãn.
Tham số kèm:
hInstance Cán của đối tượng sở hữu lớp. Dùng NULL để nhận thông tin về các lớp Windows chuẩn.
LpClassName Tên của lớp cần tìm. Có thể dùng ID resource.
LpWndClass WndCLASS - Cấu trúc để chứa kết quả.

---

**o0o--truongphu--o0o**

.....
Ghé thăm:
**Chuyện Linh Tinh**

truongphu
VIP

## 55 hàm API liên quan đến cửa sổ

T.Sáu 20/03/2009 4:05 pm

**25. Declare Function GetClassLong Lib "user32" Alias "GetClassLongA" (ByVal hwnd As Long, ByVal nIndex As Long) As Long**

Công dụng: Lấy thông tin lớp.
Trị trả về: Tuỳ theo yêu cầu.
Tham số kèm:
HWnd Cán của cửa sổ để nhận thông tin đối với lớp chứa nó.
NIndex Thông tin cần nhận. Nếu là GLC_MENUNAME lấy tên hay resource ID đối với trình đơn của lớp. Nếu là GLC_WNDPROC để nhận vị trí của hàm cửa sổ lớp (Hàm đờ phôn đối với các cửa sổ trong lớp).

**26. Declare Function GetWindowLong Lib "user32" Alias "GetWindowLongA" (ByVal hwnd As Long, ByVal nIndex As Long) As Long**

Công dụng: Lấy thông tin từ cấu trúc cửa sổ.
Trị trả về: Theo yêu cầu.
Tham số kèm:
HWnd Cán của cửa sổ cần lấy thông tin.
NIndex Thông tin cần lấy, tuỳ thuộc vào các hằng sau:

GWL_EXSTYLE: Kiểu cửa sổ mở rộng.
GWL_STYLE: Kiểu cửa sổ.
GWL_WNDPROC: Vị trí của hàm xử lý cửa sổ này.
DWL_MSGRESULT: Trị được trả về bởi thông báo bên trong hàm đối thoại.
DWL_DLGPROC: Vị trí của hàm xử lý khung đối thoại đối với cửa sổ này.
DWL_USER: Được định nghĩa bởi ứng dụng.

**27. Declare Function SetWindowLong Lib "user32" Alias "SetWindowLongA" (ByVal hwnd As Long, ByVal nIndex As Long, ByVal dwNewLong As Long) As Long**

Công dụng: Thiết đặt thông tin trong cấu trúc cửa sổ.
Trị trả về: Integer - Trị trước đó của dữ kiện cần đặt giá trị.
Tham số kèm:
HWnd Cán của cửa sổ để đặt thông tin.
NIndex Thông tin cần đặt. Xem hàm trên.
DwNewLong Trị mới cần đặt.

**28. Declare Function GetWindowText Lib "user32" Alias "GetWindowTextA" (ByVal hwnd As Long, ByVal lpString As String, ByVal cch As Long) As Long**

Công dụng: Lấy tiêu đề của cửa sổ hay nội dung của ô điều khiển.

Trị trả về: Integer - chiều dài chuỗi được lấy không tính ký tự null đứng cuối.
Tham số kèm:
HWnd Cán của cửa sổ cần lấy.
LpString Biến lưu kết quả là tên chuỗi cần lấy. Phải khai báo tối thiểu aint+1. Dùng công thức sau lấy tên chuỗi: Chuỗi = Left (lpString, len (trim(lpString)).
Aint Chiều dài chuỗi lpString

**29. Declare Function GetWindowTextLength Lib "user32" Alias "GetWindowTextLengthA" (ByVal hwnd As Long) As Long**

Công dụng: Lấy chiều dài của tiêu đề cửa sổ hay nội dung của một ô điều khiển.
Trị trả về: Chiều dài chuỗi cửa sổ..
Tham số kèm:
HWnd Cán của cửa sổ cần lấy.

**30. Declare Function GetWindowWord Lib "user32" Alias "GetWindowWord" (ByVal hwnd As Long, ByVal nIndex As Long) As Integer**

Công dụng: Lấy thông tin từ cấu trúc của cửa sổ chỉ định.
Trị trả về: Theo yêu cầu.
Tham số kèm:
HWnd Cán của cửa sổ cần lấy.
NIndex Thông tin cần lấy, phụ thuộc vào một trong các hằng:
GWW_HINSTANCE: Cán của chủ cửa sổ.
GWW_HWNDPARENT: Cán cửa sổ chứa (Mức Parent) nó.
GWW_ID: Số ID của cửa sổ con (Mức Child) bên trong khung đối thoại.

**31. Declare Function SetWindowWord Lib "user32" Alias "SetWindowWord" (ByVal hwnd As Long, ByVal nIndex As Long, ByVal wNewWord As Long) As Long**

Công dụng: Đặt thông tin trong cấu trúc cửa sổ.
Trị trả về: Integer - Trị trước khi đặt của dữ liệu cần thay.
Tham số kèm:
HWnd Cán của cửa sổ cần đặt.
NIndex Như hàm trên.
DwNewWord - Trị mới cần đặt.

**32. Declare Function InflateRect Lib "user32" Alias "InflateRect" (lpRect As RECT, ByVal x As Long, ByVal y As Long) As Long**

Công dụng: Thay đổi kích thước của hình chữ nhật.
Tham số kèm:
LpRect Cấu trúc hình chữ nhật cần điều chỉnh
X Chiều rộng được tăng lên hay giảm đi.
Y Chiều cao tăng lên hay giảm đi.

**33. Declare Function IntersectRect Lib "user32" Alias "IntersectRect" (lpDestRect As RECT, lpSrc1Rect As RECT, lpSrc2Rect As RECT) As Long**

Công dụng: Nạp vào hình chữ nhật đích phần chung của 2 hình chữ nhật đơn.
Trị trả về: Integer (Khác zero)- Nếu hình chữ nhật đích không rỗng. Zero nếu rỗng.
Tham số kèm:
LpDestRect - Hình chữ nhật đích.
LpScr1Rect, lpSrc2Rect: Hai hình chữ nhật giao nhau.

**34. Declare Function InvalidateRect Lib "user32" Alias "InvalidateRect" (ByVal hwnd As Long, lpRect As RECT, ByVal bErase As Long) As Long**

Công dụng: Làm sai bất hợp lệ tất cả hay một phần vùng làm việc của một cửa sổ. Để vẽ lại đúng lúc, đúng chỗ.
Tham số kèm:
HWnd Cán của cửa sổ cần làm mất hợp lệ.
LpRect hình chữ nhật mô tả phần không hợp lệ.
BErase Cho về True (Khác zero) để xoá vùng chỉ định trước khi vẽ lại.

**35. Declare Function IsChild Lib "user32" Alias "IsChild" (ByVal hWndParent As Long, ByVal hwnd As Long) As Long**

Công dụng: Xác định cửa sổ cần xét có phải cửa sổ con (Mức Child) thuộc nhánh cửa sổ khác.
Trị trả về: Integer - True (Khác zero) nếu HWnd là cửa sổ con (Mức Child) hay hậu duệ của HWndParent
Tham số kèm:
HWnd Cán của cửa sổ cần kiểm tra
HWndParent Cán của cửa sổ chứa (Mức Parent).

**36. Declare Function IsIconic Lib "user32" Alias "IsIconic" (ByVal hwnd As Long) As Long**

Công dụng: Kiểm tra cửa sổ có phải đã thu nhỏ thành biểu tượng không.
Trị trả về: Integer - True (Khác zero) nếu bị thu nhỏ
Tham số kèm:
HWnd Cán của cửa sổ cần kiểm tra.

---

**o0o--truongphu--o0o**

.....
Ghé thăm:
**Chuyện Linh Tinh**

truongphu
VIP

## 55 hàm API liên quan đến cửa sổ

T.Sáu 20/03/2009 4:11 pm

**37. Declare Function IsRectEmpty Lib "user32" Alias "IsRectEmpty" (lpRect As RECT) As Long**

Công dụng: Kiểm tra xem hình chữ nhật có rỗng không.
Trị trả về: Integer - True (Khác zero) nếu rỗng. Zero nếu không rỗng.
Tham số kèm:
LpRect Hình chữ nhật cần kiểm tra.

**38. Declare Function IsWindow Lib "user32" Alias "IsWindow" (ByVal hwnd As Long) As Long**

Công dụng: Xác định xem có phải là cán cửa sổ không.
Trị trả về: Integer - True (Khác zero) nếu đúng là cán cửa sổ.
Tham số kèm:
HWnd Cán cần kiểm tra.

**39. Declare Function IsWindowEnabled Lib "user32" Alias "IsWindowEnabled" (ByVal hwnd As Long) As Long**

Công dụng: Kiểm tra cửa sổ có hiệu lực (enabled) không.
Trị trả về: Integer - True (Khác zero) nếu có hiệu lực.
Tham số kèm:
HWnd Cán của cửa sổ cần kiểm tra

**40. Declare Function IsWindowVisible Lib "user32" Alias "IsWindowVisible" (ByVal hwnd As**

**Long) As Long**

Công dụng: Kiểm tra xem cửa sổ xem có thể nhìn thấy nó trên màn hình, kể cả cửa sổ bị cửa sổ khác xếp chồng lên trên.
Trị trả về: Integer - True (Khác zero) nếu nhìn thấy được.
Tham số kèm:
HWnd Cán của cửa sổ cần kiểm tra.

**41. Declare Function IsZoomed Lib "user32" Alias "IsZoomed" (ByVal hwnd As Long) As Long**

Công dụng: Kiểm tra xem cửa sổ xem có phóng to toàn màn hình không.
Trị trả về: Integer - True (Khác zero) nếu phóng toàn màn hình.
Tham số kèm:
HWnd Cán của cửa sổ cần kiểm tra.

**42. Declare Function LockWindowUpdate Lib "user32" Alias "LockWindowUpdate" (ByVal hwndLock As Long) As Long**

Công dụng: Khoá cửa sổ, không cho cập nhật. Mỗi lần chỉ có 1 cửa sổ bị khoá.
Trị trả về: Integer - True (Khác zero) nếu thành công. Zero nếu đã có cửa sổ khác bị khoá.
Tham số kèm:
HWndLock Cán của cửa sổ cần khoá.

**43. Declare Function MapWindowPoints Lib "user32" Alias "MapWindowPoints" (ByVal hwndFrom As Long, ByVal hwndTo As Long, lppt As Any, ByVal cPoints As Long) As Long**

Công dụng: Chuyển đổi các điểm theo các toạ độ sử dụng (client) của một cửa sổ sang các toạ độ cuả cửa sổ khác.
Tham số kèm:
HWndFrom, HWndTo Cán của cửa sổ nguồn và đích. Nếu một cán là toạ độ theo màn hình thì chọn cán là cán của Desktop.
Lppt Điểm chốt POINTAPI của mảng chuyển đổi.
CPoints Số điểm chuyển đổi.

**44. Declare Function MoveWindow Lib "user32" Alias "MoveWindow" (ByVal hwnd As Long, ByVal x As Long, ByVal y As Long, ByVal nWidth As Long, ByVal nHeight As Long, ByVal bRepaint As Long) As Long**

Công dụng: Di chuyển và định lại kích thước cửa sổ.
Tham số kèm:
HWnd Cán của cửa sổ cần di chuyển.
X,y Toạ độ mới của đỉnh trái cửa sổ.
NWidth, nHeight Chiều rộng và chiều cao mới của cửa sổ.
BRepaint Integer - True (Khác zero) nếu muốn cửa sổ vẽ lại tự động sau khi di chuyển. False (zero) nếu ứng dụng tự vẽ lại.

**45. Declare Function OffsetRect Lib "user32" Alias "OffsetRect" (lpRect As RECT, ByVal x As Long, ByVal y As Long) As Long**

Công dụng: Di chuyển và thay đổi kích thước một vùng hình chữ nhật. Lưu ý Các chiều kích thước mới không quá 72767 đơn vị.
Tham số kèm:
LpRect - Hình chữ nhật cần di chuyển và thay đổi kích thước.
X - Khoảng cách dịch chuyển cho góc trái trên hình chữ nhật.
Y - Khoảng cách dịch chuyển cho góc phải dưới hình chữ nhật.

**46. Declare Function PostMessage Lib "user32" Alias "PostMessageA" (ByVal hwnd As Long, ByVal wMsg As Long, ByVal wParam As Long, ByVal lParam As Long) As Long**

Công dụng: Gửi một chỉ lệnh vào hàng đợi message queue của một cửa sổ. Các chỉ lệnh này sẽ được xử lý theo tuần tự.
Trị trả về: Integer - True (Khác zero) nếu thành công.
Tham số kèm:
HWnd Cán của cửa sổ nhận chỉ lệnh.
WMsg Hằng số ID của chỉ lệnh. (Xin tra công dụng của các hằng ở bảng khác)
WParam, lParam Các tham số tuỳ thuộc vào chỉ lệnh.

**47. Declare Function PtInRect Lib "user32" Alias "PtInRect" (lpRect As RECT, pt As POINTAPI) As Long**

Công dụng: Kiểm tra điểm có nằm trong hình chữ nhật không.
Trị trả về: Integer - True (Khác zero)nếu nằm trong. Zero nếu ngoài.
Tham số kèm:
LpRect Hình chữ nhật để kiểm tra.
pt DDieemr cần kiểm tra.

**48. Declare Function RedrawWindow Lib "user32" Alias "RedrawWindow" (ByVal hwnd As Long, lprcUpdate As RECT, ByVal hrgnUpdate As Long, ByVal fuRedraw As Long) As Long**

Công dụng: Vẽ lại cửa sổ.
Trị trả về: Integer - True (Khác zero)nếu thành công. Zero nếu thất bại.
Tham số kèm:
HWnd Cán của cửa sổ để vẽ lại.
LprcUpdate - Hình chữ nhật bên trong cửa sổ cần vẽ lại.
HrgnUpdate - Cán của miền mô tả khu vực cần vẽ lại.
FuRedraw - Cờ yêu cầu vẽ lại, là một trong các hằng sau:
RDW_ERASE - Nền phẫn vẽ lại phải xoá trước khi vẽ.
RDW_FRAME - Cập nhật khung vẽ lại, nếu khung vẽ trùm lên tiêu đề, thực đơn, dòng trạng thái....
RDW_INTERNALPAINT - Gửi chỉ lệnh WM_PAINT cho cửa sổ.
RDW_INVALIDATE - Yêu cầu vẽ lại khu vực khung HrgnUpdate.
RDW_NOERASE - Không xoá nền của khung cần vẽ lại.
RDW_NOFRAME - Không cập nhật nếu khung vẽ lại trùm lên tiêu đề, thực đơn, dòng trạng thái.
RDW_NOINTERNALPAINT - Cấm các chỉ lệnh WM_PAINT đối với cửa sổ.
RDW_VALIDATE - Thừa nhận khung vẽ lại hợp lệ.
RDW_ERASENOW - Xoá ngay khung vẽ lại.
RDW_UPDATENOW - Cập nhật ngay khung vẽ lại.
RDW_ALLCHIDREN - Thao tác vẽ lại thực hiện luôn trên cả các cửa sổ con (Mức Child) nằm trong khung vẽ lại.
RDW_NOCHIDREN - Không vẽ lại các cửa sổ con (Mức Child), nếu nó nằm trong khung vẽ lại.

**49. Declare Function ScreenToClient Lib "user32" Alias "ScreenToClient" (ByVal hwnd As Long, lpPoint As POINTAPI) As Long**

Công dụng: Chuyển toạ độ một điểm trên màn hình thành toạ độ tương đối của cửa sổ.
Tham số kèm:
HWnd Cán của cửa sổ làm căn cứ toạ độ.
LpPoint Điểm cần chuyển

**50. Declare Function ShowWindow Lib "user32" Alias "ShowWindow" (ByVal hwnd As Long, ByVal nCmdShow As Long) As Long**

Công dụng: Điều khiển hiện cửa sổ.
Trị trả về: Integer - Nếu cr được nhìn thấy trước đó. Zero nếu ngược lại.
Tham số kèm:
HWnd Cán của cửa sổ cần điều khiển.
NCmdShow - Integer - Là các chỉ lệnh hằng sau:

SW_HIDE: Giấu cửa sổ.
SW_MINIMIZE: Thu nhỏ thành biểu tượng.
SW_RESTORE: Hiện lại như lúc ban đầu, kích hoạt.
SW_SHOW: Hiện lại như lúc chưa giấu, kích hoạt
SW_SHOWMAXIMIZED: Hiện mở rộng tối đa, kích hoạt.
SW_SHOWMINIMIZED: Hiện như biểu tượng, kích hoạt.
SW_SHOWMINNOACTIVE: Thu nhỏ cửa sổ, không làm thay đổi cửa sổ đang kích hoạt.
SW_SHOWNA: Hiện một cửa sổ ở kích thước và vị trí hiện tại, không làm thay đổi cửa sổ đang kích hoạt.
SW_SHOWNOACTIVE: Hiện cửa sổ như trước khi giấu, không làm thay đổi cửa sổ đang kích hoạt.
SW_SHOWNORMAL: Hiện ra bình thường.

**51. Declare Function SubtractRect Lib "user32" Alias "SubtractRect" (lprcDst As RECT, lprcSrc1 As RECT, lprcSrc2 As RECT) As Long**

Công dụng: Nạp vào cửa sổ đích phần trừ của 2 cửa sổ khác.
Trị trả về: Integer - True (Khác zero) nếu thành công. Zero nếu thất bại.
Tham số kèm:
LpDestRect - Hình chữ nhật đích.
lprcSrc1, lprcSrc2: Hai hình chữ nhật nguồn trừ nhau.

**52. Declare Function UnionRect Lib "user32" Alias "UnionRect" (lpDestRect As RECT, lpSrc1Rect As RECT, lpSrc2Rect As RECT) As Long**

Công dụng: Nạp vào cửa sổ đích phần cộng của 2 cửa sổ khác.
Trị trả về: Integer - True (Khác zero) nếu thành công. Zero nếu thất bại.
Tham số kèm:
LpDestRect - Hình chữ nhật đích.
lprcSrc1Rect, lprcSrc2Rect: Hai hình chữ nhật nguồn cần cộng.

**53. Declare Function UpdateWindow Lib "user32" Alias "UpdateWindow" (ByVal hwnd As Long) As Long**

Công dụng: Cập nhật ngay cửa sổ.
Tham số kèm:
HWnd Cán của cửa sổ cần cập nhật.

**54. Declare Function ValidateRect Lib "user32" Alias "ValidateRect" (ByVal hwnd As Long, lpRect As RECT) As Long**

Công dụng: Hợp lệ hoá cửa sổ, để không cần vẽ lại.
Tham số kèm:
HWnd Cán của cửa sổ cần hợp lệ hoá.
LpRect - Hình chữ nhật cần hợp lệ hoá. Nếu đặt zero thì hợp lệ toàn bộ cửa sổ.

**55. Declare Function WindowFromPoint Lib "user32" Alias "WindowFromPoint" (ByVal xPoint As Long, ByVal yPoint As Long) As Long**

Công dụng: Lấy cán cửa sổ chứa (Mức Parent) điểm cần khai báo.
Trị trả về: Integer - Cán của cửa sổ chứa (Mức Parent) điểm. Zero nếu không có cửa sổ nào.
Tham số kèm:
XPoint, Ypoint: Điểm theo toạ độ màn hình.

---

**o0o--truongphu--o0o**

.....
Ghé thăm:
**Chuyện Linh Tinh**

**truongphu**
VIP

## WINDOWS API

T.Sáu 20/03/2009 4:21 pm

**WINDOWS API**
**Khám phá từ A đến Z**

Bản chất của Windows API.
*(Bài sau đây là Copy từ địa chỉ trên)*

Trong lập trình Visual Basic độc lập hoặc Visual Basic for Application, Microsoft đã cung cấp cho chúng ta một bộ các hàm lập sẵn, hàng trăm hàm API (Aplication Programming Interface) được lưu trong các tệp thư viện liên kết động (Tệp đuôi *.DLL - Dynamic Link Library). Đó là công cụ tuyệt vời cho phép bạn phát triển ứng dụng cực mạnh, tại sao bạn lại bỏ qua và không sử dụng nó?
Tôi sẽ cùng bạn khám phá những gì mà Microsoft cung cấp các hàm Windows API trong bộ Visual Studio. Tuy nhiên vì khuôn khổ cũng như kích thước của bài viết, ta chỉ đi vào các nét chính căn bản nhất, bạn có thể tham khảo trong Help hoặc các bài viết của Nguyễn Hồ Thiên Đăng, Nguyễn Thị Thanh Phương tại WebsiteLH.

**I. Hàm API** - Nhìn từ góc độ người ít có điều kiện học Tin học

Nếu bạn chưa từng lập trình những chương trình lớn, bạn sẽ phát hoảng khi đọc khai báo (rắc rối và kỳ cục!!!) của API:

**MÃ: CHỌN HẾT**

```
1. Private Declare Function CallNextHookEx Lib "user32" Alias "CallNextHookEx" (ByVal hHook As Long,
   ByVal ncode As Long, ByVal wParam As Long, lParam As Any) As Long
```

Tuy nhiên bạn có thể chép phần khai báo trên thật đơn giản bằng Text API Viewer. Theo các chuyên gia về Tin học thì đừng bao giờ sử dụng thẳng hàm API trong thủ tục thiết kế chính của mình. Thay vào đó ta thiết kế một hàm hay thủ tục Visual Basic thay thế hàm API để đơn giản hoá (Gọi là wrapper - Tôi không tìm được từ tiếng Việt tương ứng để nói đủ bản chất của nó.)
Ví dụ một wrapper sau:

**MÃ: CHỌN HẾT**

```
1. Public Sub ThoatVaTatMay ()
2. 'Thoát và tắt máy
3. Dim thoat
4. Thoat = ExitWindowsE 2,0)
5. End Sub
```

Khi đó, trong chương trình Visual Basic của ta khi cần thoát và tắt máy chỉ việc gọi:

**MÃ: CHỌN HẾT**

```
1. ThoatVaTatMay
```

hoặc

**MÃ: CHỌN HẾT**

```
1. Call ThoatVaTatMay
```

Là máy tính thực hiện thoát và tắt máy.
Chính vì thế, khi học viên dân tộc được học phổ cập API tại Trung tâm Dạy nghề và Phổ cập Tin học Miền núi ABC của chúng tôi đã gọi chức năng tạo Wrapper chẳng khác dán nhãn Tiếng Việt cho từng loại thuốc tây API vậy. Nên khi lập trình ta nên tạo các Wrapper tương ứng với các chức năng mà mình muốn sử dụng. Đó cũng là lời khuyên của Bill Gate cho chúng ta.
Để tạo các wrapper bạn hãy chèn một Module vào Project. Nếu máy của bạn không cài Visual Basic thì bạn phải copy hay đánh thật chính xác những dòng khai báo dạng như khai báo trên, thật khổ sở nếu như phần tiếng Anh của bạn không thạo lắm, vì một sai sót nhỏ có thể dẫn tới lỗi nặng cho máy. Nếu máy có cài Visual Basic thì quá tốt, chỉ việc khởi động API Viewer để hiện bảng giao tiếp như thế này:
Bạn phải mở các tệp TXT trong thư mục API của thư mục cài Visual Basic trong máy bạn. Ví dụ tệp Win32api.TXT. Máy sẽ hỏi có chuyển thành dạng cơ sở dữ liệu không, thì hãy chọn có để sử dụng API thuận lợi và nhanh chóng hơn. Khi đã copy vào clipboard, bạn có thể dán vào module của mình. Nó sẽ thành dạng tương tự như thế này:

**MÃ: CHỌN HẾT**

```
1. Declare Function CallNextHookEx Lib "user32" Alias "CallNextHookEx" (ByVal hHook As Long, ByVal ncode
   As Long, ByVal wParam As Long, lParam As Any) As Long
```

Bạn phải đánh thêm vào trước khai báo trên cụm từ Private để được:

**MÃ: CHỌN HẾT**

```
1. Private Declare Function CallNextHookEx Lib "user32" Alias "CallNextHookEx" (ByVal hHook As Long,
   ByVal ncode As Long, ByVal wParam As Long, lParam As Any) As Long
```

Tôi xin phép được nhắc lại 2 từ khoá khai báo trong Visual Basic là Private và Public.
**Private** - Khai báo dùng riêng trong Module. Có nghĩa là bạn chỉ sử dụng được nó trong Module này. Nếu chèn Module khác sẽ không sử dụng được nó.
**Public** - Khai báo dùng chung, bạn có thể dùng nó ở bất cứ Module nào.
Bạn biết đấy, ta khai báo các hàm API thì lại dùng Private, còn khai báo các Wrapper lại dùng Public. Đó chính là mẹo mà chúng ta đã học trộm được của Microsoft để tránh lỗi hệ thống.

Các hàm mà ta thiết kế trong Visual Basic có điều khác với các hàm API. Tại sao? Vì hàm ta thiết kế (Tạm gọi là hàm Visual Basic) thường chỉ có một kết quả trả về của hàm làm căn cứ xử lý. Còn hàm API không phải chỉ có một kết quả trả về mà nó còn trả về tiếp các giá trị vào các biến mà ta truyền cho nó. Nghĩa là có gọi nó là hàm thì bản chất nó là một thủ tục, ta gọi hàm để thực hiện và kiểm tra xem thủ tục trong hàm đó có thực hiện được thành công hay không mà thôi.
Điều này chỉ hơi giống như khi bạn lập trình với hàm Visual Basic mà bạn khai báo Public Static - Biến tĩnh dùng chung, để làm biến đổi nó trong hàm của bạn. Ta nghiên cứu một cách tổng quát như sau để hiểu cặn kẽ:
Giả sử ở một Wrapper bạn dùng công thức:

**MÃ: CHỌN HẾT**

```
1. TENBIEN=TEN_HAM_API(Bien1, Bien2, Bien3)
```

Thực ra đây là một thủ tục. Nếu TENBIEN<>0 thì thủ tục này thành công.
Khi đó các Bien1, Bien2, Bien3 truyền vào, sẽ có một giá trị mới. Lập trình API lúc này không chỉ xử lý TENBIEN, mà bạn có thể xử lý các Bien1, Bien2, Bien3. Đó mới thực sự là sức mạnh của API.
Bản chất lập trình của biến là một vùng các ô nhớ trong RAM được đặt tên để tiện sử dụng. Tại một thời điểm biến chỉ có một giá trị duy nhất. Người ta có thể dùng biến này làm giá trị định vị hoặc kích thước lưu trữ cho biến kia. Các hàm API có thể được gọi nhiều lần để sử dụng kết quả trả về của các biến truyền Bien1, Bien2, Bien3 để xử lý theo quy luật xác định nào đó.

**II. Xác định mục đích khi sử dụng WinAPI.**

Trong lập trình API có thể phân làm nhiều mục đích sử dụng, tôi chưa từng được học Tin học một cách chính thống, nên có thể khả năng phân tích và tổng hợp những bài viết trên INTERNET của Microsoft khác với những bài học của những học viên học Tin học chính quy. Nhưng tôi nghĩ chúng ta nắm bản chất của vấn đề mới là điều quan trọng.
Có 3 vấn đề chính khi sử dụng và khai thác WinAPI đó là:
a. Kỹ thuật Subclass: Để cải tổ các đối tượng Visual Basic.
b. Kỹ thuật Hook: Câu móc từ chương trình Visual Basic với các chương trình khác. Lấy giá trị nhập vào các chương trình khác của người sử dụng đưa vào chương trình của mình để xử lý.
c. Kỹ thuật Multicasting: Dùng một đối tượng tạo lập để theo dõi, chi phối các đối tượng khác của Visual Basic.
Bạn có thể sử dụng từng Kỹ thuật hoặc cả 3. Tuy nhiên bước đầu chưa thạo, bạn hãy thực hiện từng Kỹ thuật một cho thành thạo. Sau khi kiểm soát được khả năng của mình, bạn sẽ đủ trình độ để sử dụng WinAPI để cải tổ Windows và máy tính.
*Xin đọc tất cả các bài viết của Nguyễn Hồ Thiên Đăng, Nguyễn Thị Thanh Phương và Nguyễn Phương Thảo về lập trình Visual Basic trên WebsiteLH.*

Khi nắm rõ bản chất của Windows API để lập trình sâu với hệ thống, ta cũng cần hiểu biết sơ bộ về Hệ điều hành Windows cách thức điều khiển của Hệ điều hành đối với ứng dụng để có thể can thiệp như bổ sung chức năng thậm chí biến đổi nó, bắt thực hiện theo hướng của mình, ngay cả khi hướng này ngược hẳn với công dụng truyền thống !!!
Các chương trình ứng dụng trong Windows có thể có nhiều cửa sổ phục vụ cho nó. Cửa sổ có thể là Form thậm chí là Dialog. Mỗi cửa sổ này đều có một handle (Cán) để hệ thống nhận biết do chính hệ điều hành Windows tạo ra. Cán cửa sổ này là chỉ số duy nhất.

Hệ điều hành và chương trình ứng dụng đều duy trì các hàng đợi các chỉ lệnh cần thực hiện. Mỗi ứng dụng đều có hàng đợi (Message Queue). Khi người sử dụng ra lệnh hoặc có một biến cố, các chương trình điều khiển thiết bị nhập (INPUT) sẽ chuyển các thông tin vào thành chỉ lệnh và đặt chỉ lệnh này vào hàng đợi hệ thống (System Message Queue). Hệ điều hành lấy lần lượt các chỉ lệnh trong hàng đợi hệ thống kiểm tra để xác định cửa sổ nào sẽ tiếp nhận thi sẽ đặt vào hàng đợi của nó (thread message) một chỉ lệnh tương ứng. Các chương trình ứng dụng căn cứ vào chỉ lệnh này để thực hiện cũng như xử lý chúng.
Các cửa sổ giống như một động cơ tự động chạy theo một vòng lặp. Tiếp "nhiên liệu" cho các "động cơ" này là hệ điều hành Windows. Hệ điều hành Windows nhận các chỉ lệnh (message) từ hàng đợi của hệ điều hành, dùng một hàm dạng API (Như kỳ 1 - Hàm này bản chất là một thủ tục) để cung cấp chỉ lệnh tới cửa sổ thông qua cán (handle) của cửa sổ.
Có nghĩa là bản thân trong mỗi cửa sổ luôn có một hàm gọi là WinProc (Đôi khi gọi là WinMain()). Hàm này là cốt lõi xử lý của cửa sổ. Trong hàm, nó lặp đi lặp lại liên miên 2 dòng lệnh sau thông qua cấu trúc:

> “
> Do While 0 <>GetMessage (message, 0, 0,0)
> TranslateMessage message
> DispatchMessage message
> Loop

Trong đó message là chỉ lệnh mà Hệ điều hành cung cấp, thông qua cán (handle) của cửa sổ. Đương nhiên, nếu chỉ lệnh có giá trị WM_QUIT thì hàm WinProc trong cửa sổ chấm dứt vòng lặp.
Còn nếu chỉ lệnh message khác giá trị trên, thì 2 dòng lệnh trên sẽ thực hiện. Cụ thể:
TranslateMessage message -> Dịch chỉ lệnh thành dạng dữ liệu khác đặt kết quả này vào hàng đợi của ứng dụng.
DispatchMessage message ->Nhận chỉ lệnh từ hàm GetMessage và gửi cho hệ thống. Hệ thống sẽ đưa chỉ lệnh cho ứng dụng.
Windows có hàng ngàn chỉ lệnh khác nhau đó là các hằng dạng WM_* (Windows đặt tên cho tiện gọi thôi, vì bản chất các hằng này là một con số - Rất khó nhớ. Nếu phân tích chi tiết ra, những con số này lại là dãy số 0 và 1, tức là bật tắt ấy mà).
Một hàm WinProc luôn nhận vào trong nó các biến theo khuôn mẫu sau để xử lý:

**MÃ: CHỌN HẾT**

```
1. Function WinProc(hwnd as Long, wc as WNDCLASSEX, message as MSG, wParam as Long, lparam as Long)
```

Nếu hàm WinProc không xử lý các chỉ lệnh, nó phải đưa trả chỉ lệnh cho hệ điều hành xử lý thông qua hàm DefWindowProc. Hàm DefWindowProc gởi lại chỉ lệnh WM_CLOSE cho WinProc. Hàm WinProc sẽ lại gởi trả WM_CLOSE cho DefWindowProc một lần nữa như mô tả ở trên.
Bạn có thể thấy, kết quả và cơ chế xử lý rất lằng nhằng. Ta có thể tóm lại sơ bộ như sau:

Các chỉ lệnh đưa tới ngăn chờ trên thông thường từ các nguồn sau:
1. Hệ thống đặt vào
2. Chương trình khác đặt vào
3. Chính chương trình của mình đặt vào thông qua các hàm SendMessage() và PostMessage().
Tuy nhiên nếu bạn chọn sử dụng hàm SendMessage() thì sau khi chỉ lệnh được WinProc lấy ra xử lý thì chương trình mới tiếp tục chạy tiếp lệnh kế sau. Còn bạn dùng PostMessage() chỉ có tác dụng đặt chỉ lệnh vào hàng đợi và thực hiện ngay lệnh kế tiếp.
Từ đây ta nhận thấy việc xử lý hệ thống của Windows thông qua cơ chế trên thì giản đơn đi rất nhiều. Ta sẽ có các hướng giải quyết tiếp theo như sau:
1. Nếu ta thiết kế một hàm WinProc() mới, Ví dụ NewWinProc(), thay thế hàm WinProc() truyền thống
- Rồi ta đổi địa chỉ của WinProc() gốc sang địa chỉ của WinProc() mà ta thiết kế. Trong thủ tục này sử dụng cấu trúc Select Case để tuỳ vào chỉ lệnh message mà xử lý theo ý mình.
- Trường hợp chỉ lệnh message nào không thể viết được thủ tục thi hành (Không cần thay đổi hoặc khó quá) thì ta gọi thủ tục DefWinProc() của Windows xử lý. (Tức là dễ thì làm, khó trả lại ấy mà). Đây chính là Kỹ thuật Subclass.
- Tuy nhiên nếu như hàm WinProc() cũ, xử lý tốt một số tính năng nào đó thì ta có thể tận dụng thủ tục bằng cách gọi lại bằng hàm CallWindowProc().
- Thậm chí ta có thể lồng WinProc() vào trong NewWinProc() để xử lý
(Xem các bài ví dụ của Nguyễn Phương Thảo có sử dụng API).
2. Trường hợp không thể thay được hàm WinProc() bằng NewWinProc() (Ví dụ như bạn lập trình với các chương trình ứng dụng khác như Winword, Excel...) bạn phải chặn các chỉ lệnh trước khi nó được lấy ra khỏi hàng đợi. Đó chính là Kỹ thuật Hooking, một Kỹ thuật cực kỳ mạnh để làm việc với Windows.
Trước khi chỉ lệnh được hàm SendMessage() lấy từ hàng đợi gửi đi, hoặc cũng có thể chỉ lệnh được lấy bằng hàm PickMessage() hay GetMessage(), ta có thể đăng ký với Windows để sử dụng bộ lọc HookFilter. Khi đó những chỉ lệnh cần xử lý đã đăng ký, đều qua HOOK Filter. Ta chỉ việc viết một thủ tục dùng hàm WinProc lấy chỉ lệnh từ HOOK Filter để xử lý. *(Xem bài viết chặn các thủ tục in và nhập dữ liệu của Nguyễn Phương Thảo).*
3. Các đặc điểm của lớp cửa sổ:
- Mỗi ứng dụng có thể tạo ra nhiều cửa sổ, thông thường các cửa sổ này có những đặc điểm giống nhau và được phân theo từng lớp CLASS. Khi lập trình, bao giờ ta cũng đăng ký lớp với hệ thống thông qua một hàm là RegisterClassEx(). Khi lớp đã được đăng ký (Chỉ 1 lần) thì các thông tin window và địa chỉ hàm WinProc sẽ được lưu trong suốt thời gian mà nó tồn tại.
- Ta có thể thay đổi các thông tin trong bộ đăng ký lớp, khi đó nó sẽ ảnh hưởng đến toàn bộ cửa sổ trong lớp này.
Bạn thân mến! Như vậy bạn đã nắm chắc về cơ chế làm việc của Windows và các thủ tục hệ thống của nó. Hi vọng bạn hãy đọc thật kỹ và hiểu rõ về nó, để những bài viết sau chúng ta sẽ mổ xẻ giải phẫu từ những hàm API cơ sở được dễ dàng hơn.

---

**o0o--truongphu--o0o**

.....
Ghé thăm:
**Chuyện Linh Tinh**

truongphu
VIP

## Xử lý của API khi làm việc với phần cứng và hệ thống

T.Sáu 20/03/2009 4:34 pm

*(Bài sau đây là Copy từ địa chỉ trên)*

Bạn thân mến!
Để thuận lợi cũng như để mọi người cùng học tập và tham gia xây dựng đề án "Xây dựng bộ gõ tiếng Việt tăng tốc" chúng ta hãy làm quen với các hàm API liên quan đến vấn đề này. Mong rằng đây sẽ là cơ sở cơ bản để chúng ta cùng tiếp cận tìm hiểu cơ chế xây dựng một bộ gõ tiếng Việt đơn giản tại Website Lê Hoàn. Bạn có thể xem các hàm liên quan đến UNICODE hoặc các bài học cơ bản ban đầu Visual Basic tại trang http://www.bangden.com/soncuoc. Chi tiết và ví dụ các hàm bạn có thể tìm ở trang này.

**1. Các hàm với chuột, con trỏ**

Visual Basic cung cấp sự yểm trợ cho chuột và trỏ chuột (Trỏ chịu sự tác động của chuột thường để điều khiển) cũng như trỏ thanh (Trỏ chịu tác động của bàn phím thường để nhập liệu), tại một thời điểm chỉ có duy nhất một trỏ chuột và một trỏ thanh. Theo ngầm định, vị trí của trỏ chuột và trỏ thanh tính theo toạ độ của màn hình.
Windows cung cấp khả năng hạn chế con chuột vào một vùng khai báo trên màn hình gọi là clipping. Có hai hàm cơ bản là:

**1. Declare Function ClipCursor Lib "user32" Alias "ClipCursor" (lpRect As Any) As Long**

Công dụng: Giới hạn trỏ chuột đối với vùng chỉ định.
Tham số kèm:
LpREct - Vùng trỏ chuột định vị.

**2. Declare Function GetClipCursor Lib "user32" Alias "GetClipCursor" (lprc As RECT) As Long**

Công dụng: Nhận hình chữ nhật làm vùng làm việc cho trỏ chuột.
Tham số kèm:
Lprc - Hình chữ nhật nhận vùng làm việc.

Để nhận vị trí trỏ chuột, hay đặt trỏ chuột vào một vị trí ta sử dụng hai hàm:

**3. Declare Function GetCursorPos Lib "user32" Alias "GetCursorPos" (lpPoint As POINTAPI) As Long**

Công dụng: Xác định vị trí hiện tại của con trỏ chuột.
Tham số kèm:
LpPoint - Cấu trúc tiếp nhận toạ độ con trỏ trên màn hình.

**4. Declare Function SetCursorPos Lib "user32" Alias "SetCursorPos" (ByVal x As Long, ByVal y As Long) As Long**

Công dụng: Đặt con trỏ chuột vào một vị trí.
Tham số kèm:
X, Y Toạ độ màn hình cần đặt vị trí.

Các con trỏ theo ý muốn có thể được tạo từ cặp bitmap đơn sắc, kích thước có thể tới 32x32 pixel, nó có một cán 16 bit quản lý.
Khi ở một vị trí nào đó, ta thao tác chuột như click, d-click, hoặc drag. Các thao tác này sẽ ảnh hưởng đến một cửa sổ X nào đó. Ngay sau đó cửa sổ X này sẽ được nhận focus ngay cả khi trước đó nó đã mất focus. Vì vậy, cửa sổ này còn được gọi là capture (Hay cửa sổ đón chặn hay cửa sổ chộp).

**1. Các hàm liên quan đến trỏ chuột cơ bản gồm:**

**5. Declare Function CopyCursor Lib "user32" Alias "CopyCursor" (ByVal hcur As Long) As Long**

Công dụng: Copy thêm một trỏ chuột.
Tham số kèm:
hCur Cán của trỏ chuột cần sao chép.

**6. Declare Function GetCapture Lib "user32" Alias "GetCapture" () As Long**

Công dụng: Xác định cửa sổ đón chặn các tình huống chuột.
Trị trả về: Integer - Cán cửa sổ chặn tình huống chuột. Zero nếu không có cửa sổ nào.

**7. Declare Function GetCursor Lib "user32" Alias "GetCursor" () As Long**

Công dụng: Nhận cán của trỏ chuột hiện tại.
Trị trả về: Integer cán của trỏ chuột hiện tại. Zero nếu không có trỏ chuột hiện tại.

**8. Declare Function GetDoubleClickTime Lib "user32" Alias "GetDoubleClickTime" () As Long**

Công dụng: Xác định thời gian 2 lần nháy của thủ tục Dclick.
Trị trả về: Thời gian tính theo mili giây.

**9. Declare Function LoadCursor Lib "user32" Alias "LoadCursorA" (ByVal hInstance As Long, ByVal lpCursorName As String) As Long**

Công dụng: Tải một trỏ chuột.
Tham số kèm:
HInstance Cán của chương trình mô tả trỏ chuột
LpCursorName - Chỉ số ID nguồn. Căn cứ vào chỉ số này mà trỏ sẽ có hình dáng khác nhau. Đây là những hình dáng ngầm định của hệ thống.
IDC_APPSTARTING - Trỏ khi một ứng dụng khởi động
IDC_NO - Trỏ khi không thực hiện việc nào.
IDC_SIZE - Trỏ khi thay đổi cỡ một đối tượng.
IDC_SIZEALL - Trỏ khi thay đổi tất cả cỡ của các đối tượng
IDC_SIZENESW - Trỏ là hình mũi tên 2 chiều chéo góc xuống dốc
IDC_SIZENS - Trỏ là hình mũi tên 2 chiều dọc.
IDC_SIZENWSE - Trỏ là hình mũi tên 2 chiều chéo lên dốc
IDC_SIZEWE - Trỏ là hình mũi tên 2 chiều ngang
IDC_UPARROW - Trỏ là hình mũi tên lên
IDC_WAIT - Trỏ là đồng hồ cát
IDC_ARROW: Trỏ hình mũi tên.
IDC_CROSS: Trỏ hình dấu thập.
IDC_IBEAM: Trỏ hình thanh chữ I
IDC_ICON: Biểu tượng rỗng.

**10. Declare Function LoadCursorFromFile Lib "user32" Alias "LoadCursorFromFileA" (ByVal lpFileName As String) As Long**

Công dụng: Tải một hình dáng trỏ chuột từ tệp bên ngoài được khai báo
Trị trả về: Integer - Zero nếu thành công.
Tham số kèm:
LpFileName - Cán của tệp mô tả hình dáng trỏ.

**11. Declare Function ReleaseCapture Lib "user32" Alias "ReleaseCapture" () As Long**

Công dụng: Tải bỏ việc chặn đối với chuột.

**12. Declare Function SetCapture Lib "user32" Alias "SetCapture" (ByVal hwnd As Long) As Long**

Công dụng: Thiết lập việc chặn con chuột đối với cửa sổ.
Trị trả về: Integer là cán của cửa sổ trước đó đang chặn chuột. Zero nếu không có cửa sổ nào chặn trước đó.
Tham số kèm:
hwnd - Cán của cửa sổ cần thiết lập để chặn.

**13. Declare Function SetCursor Lib "user32" Alias "SetCursor" (ByVal hCursor As Long) As Long**

Công dụng: Đặt con trỏ hiện tại.
Trị trả về: Integer - Trị số mô tả trỏ trước đó.
Tham số kèm:
HCursor - Cán của con trỏ cần đặt làm hiện tại.

**14. Declare Function SetDoubleClickTime Lib "user32" Alias "SetDoubleClickTime" (ByVal wCount As Long) As Long**

Công dụng: Đặt thời gian giữa hai lần bấm của thao tác Dclick.
Tham số kèm:
WCount - Thời gian cần đặt, tính theo miligiây.

**15. Declare Function ShowCursor Lib "user32" Alias "ShowCursor" (ByVal bShow As Long) As Long**

Công dụng: Hiện hay giấu con trỏ trên màn hình.
Trị trả về: Số nguyên DP báo số lần hiển thị. Mỗi lần sử dụng hàm này để hiện trỏ DP sẽ được cộng 1. Mỗi lần giấu đi DP sẽ trừ 1. Nếu DP âm trỏ không được hiện.
Tham số kèm:
BShow - Trị số điều khiển. Nếu dương, để hiện. Số 0 để dấu.

**16. Declare Function SwapMouseButton Lib "user32" Alias "SwapMouseButton" (ByVal bSwap As Long) As Long**

Công dụng: Kiểm tra sự tráo đổi chức năng 2 phím chuột.
Trị trả về: Integer - True nếu đã bị hoán đổi. False nếu vẫn chưa bị đổi.
Tham số kèm:
BSwap - Đặt là True để đổi. Đặt là False để không đổi.

**2. Các mã quét và phím ảo:**

Các phím trên bàn phím tạo ra các mã quét (scan code) tuỳ thuộc vào vị trí của chúng. Khi ta tác động vào phím, bàn phím sẽ gửi mã quét của phím đó cho máy tính. Do có nhiều loại bàn phím khác nhau nên Windows đưa ra khái niệm phím ảo (virtual key) là phím không phụ thuộc vào vị trí của từng loại bàn phím, giúp cho việc thống nhất xử lý. Ví như bạn nhấn phím Enter, ở bất cứ bàn phím nào, ở vị trí nào vẫn là phím đó, chứ không liên quan đến vị trí như mã quét.
Do đó ta có thể hiểu vấn đề này như sau:
Khi một phím được nhấn, Windows sẽ nhận được tín hiệu và chuyển đổi thông tin mã quét của bàn phím thành một mã phím ảo. Phần xử lý phím ảo sẽ chịu trách nhiệm chuyển tiếp hình ảnh của ký tự được nhấn, sau khi đã căn cứ vào trạng thái của phím Shift.

**Các hàm xử lý bàn phím bao gồm:**

**17. Declare Function GetAsyncKeyState Lib "user32" Alias "GetAsyncKeyState" (ByVal vKey As Long) As Integer**

Công dụng: Nhận tình trạng của phím ảo được khai báo.
Trị trả về: Integer - Bit 0 là 1 nếu phím đã được nhấn kể từ lần gọi hàm này gần đây nhất. Ngược lại là Zero. Bit 15 là 1 nếu phím đang nhấn xuống, hay zero nếu thả ra.
Tham số kèm:
VKey Mã của phím cần kiểm tra (Xem bảng mã key code các phím)

**18. Declare Function GetKBCodePage Lib "user32" Alias "GetKBCodePage" () As Long**

Công dụng: Nhận trang mã Windows cần chuyển đổi giữa các bộ ký tự.
Trị trả về: Integer - Là các trị số chỉ trang mã.

**19. Declare Function GetKeyboardState Lib "user32" Alias "GetKeyboardState" (pbKeyState As Byte) As Long**

Công dụng: Nhận trạng thái phím ảo của bàn phím.
Trị trả về: Đọc trạng thái ghi vào biến cần lưu.
Tham số kèm:
PbKeyState - Biến dạng byte (Là chuỗi có chiều dài có thể lưu trữ 256 ký tự) nhằm lưu giá trị để nhận tình trạng các phím. Giá trị này phải được hiểu là nhị phân, căn cứ vào bit 0 của các đoạn tương ứng với các phím Caplock, NumLock, Scroll Lock, nếu là 1 sẽ là đang bật, còn bit 7 các phím thường, nếu nhấn xuống là 1, giá trị 0 là thả ra.

**20. Declare Function GetKeyboardType Lib "user32" Alias "GetKeyboardType" (ByVal nTypeFlag As Long) As Long**

Công dụng: Nhận kiểu bàn phím đang dùng.
Trị trả về: Integer - Zero nếu có lỗi. Các số khác chỉ loại bàn phím đăng ký.
Nếu cờ bằng 0, trị trả về:
1 - Tương ứng bàn phím 83 phím
2 - Tương ứng 102 phím
3 - Tương ứng 84 phím
4 - Tương ứng 101 -102 phím (IBM)
5 - Tương ứng Nokia 1050
6 - Tương ứng Nokia 9140
7 - Tương ứng tiếng Nhật.
Nếu cờ bằng 1, trị trả về tuỳ thuộc hãng sản xuất.
Nếu cờ bằng 2, trị trả về:
1, 3, 5 - Tương ứng bàn phím 10 phím chức năng.
2 - Tương ứng bàn phím 12 hoặc 18 phím chức năng.
4 - Tương ứng bàn phím 12 phím chức năng.
6 - Tương ứng bàn phím 24 phím chức năng.
7 - Do hãng sản xuất quy định.
Tham số kèm:
NTypeFlag - Cờ xác định kiểu dữ liệu cần lấy.

**21. Declare Function GetKeyNameText Lib "user32" Alias "GetKeyNameTextA" (ByVal lParam As Long, ByVal lpBuffer As String, ByVal nSize As Long) As Long**

Công dụng: Nhận tên của phím.
Trị trả về: Chiều dài của tên phím nhận được trong lpBuffer
Tham số kèm:
LParam Là một số mà bit từ 0 - 5 đặt là 0. Bít 16 - 23 đặt mã quét của phím cần xác định tên, bít 24 - Bít mở rộng trên những bàn phím nâng cao. Bit 25 - Khi gán bằng 1, bỏ qua sự khác biệt giữa các phím trái, phải.
LpBuffer Chuỗi được gán trước để nhận tên phím. Nên có nSize + 1 bytes.
NSize Chiều dài tối đa của chuỗi.

**22. Declare Function GetKeyState Lib "user32" Alias "GetKeyState" (ByVal nVirtKey As Long) As Integer**

Công dụng: Nhận trạng thái phím ảo.
Trị trả về: Bit 0 là 1 khi các phím như Caplock, NumLock, ScrollLock là bật, nếu bit 0 là 0, các phím này tắt.
Bit 15 là 1, nếu phím đang bị nhấn, bit 15 là 0 nếu vừa nhả.
Tham số kèm:
NVirtKey - Mã phím ảo để kiểm tra

**23. Declare Function MapVirtualKey Lib "user32" Alias "MapVirtualKeyA" (ByVal wCode As Long, ByVal wMapType As Long) As Long**

Công dụng: Thực hiện những sự chuyển đổi mã quét và ký tự tuỳ thuộc vào kiểu ánh xạ cung cấp.
Trị trả về: Tuỳ thuộc vào kiểu ánh xạ yêu cầu, trị trả về theo bảng sau
WCode là
Giá trị của VMapType
0
1
2
Mã phím ảo
Mã quét tương ứng

Mã quét

Mã phím ảo tương ứng

Mã phím ảo

Trị ASCII tương ứng.

Tham số kèm:
WCode - Giá trị nguồn cần chuyển đổi.
WMapType - Điều khiển kiểu chuyển đổi.

**24. Declare Function MapVirtualKeyEx Lib "user32" Alias "MapVirtualKeyExA" (ByVal uCode As Long, ByVal uMapType As Long, ByVal dwhkl As Long) As Long**

Công dụng Hàm dịch mã phím ảo thành một mã quét hoặc giá trị ký tự, hoặc dịch một mã quét thành mã phím ảo. Chức năng dịch các mã dùng cho ngôn ngữ nhập và định dạng vị trí.
Tham số kèm:
Parameters

uCode
Mã phím ảo hoặc mã quét của một phím. Giá trị này phải tương ứng với uMapType.
uMapType
[in] Kiểu dữ liệu cần dịch. Giá trị này phải phù hợp với tham số uCode trên.
Các giá trị 0, 1, 2 như bảng trên.
Riêng giá trị bằng 3 chỉ sử dụng cho Windows NT/2000 trở lên: uCode là một mã quét và được dịch thành mã phím ảo để phân biệt giữa những phím bên tay trái và những phím bên tay phải. Nếu điều này không dịch, hàm trả về 0

dwhkl
[in] Định vị vị trí nhập để sử dụng cho việc dịch các mã đặc biệt. Tham số này có thể là bất kỳ giá trị định vị vị trí trước đó mà đã nhận được từ hàm LoadKeyboardLayout
Trị trả về

Hoặc là một ã quét, môt mã phím ảo hoặc một giá trị ký tự tuỳ thuộc vào uCode và uMapType. Nếu điều này không được dịch, hàm trả về zero.
Một ứng dụng có thể dùng MapVirtualKeyEx để dịch các mã quét thành các hằng mã phím ảo như VK_SHIFT, VK_CONTROL, và VK_MENU, và ngược lại. Quá trình dịch không do not phân biệt giữa các phím trá hay phải của các phím SHIFT, CTRL, hoặc ALT. Còn Windows NT/2000 trở lên thì phân biệt, nó sẽ chuyển được thành các hằng mã phím ảo như:
VK_LSHIFT

VK_RSHIFT
VK_LCONTROL
VK_RCONTROL
VK_LMENU
VK_RMENU
Sự phân biệt này ảnh hưởng tới các hàm GetKeyboardState, SetKeyboardState, GetAsyncKeyState, GetKeyState, MapVirtualKey, và chính MapVirtualKeyEx .
Xin xem bảng mã phím ảo Virtual-Key Codes.

**25. Declare Function OemKeyScan Lib "user32" Alias "OemKeyScan" (ByVal wOemChar As Long) As Long**

Công dụng: Nhận mã quét và trạng thái Shift đối với một ký tự ASCII trong bộ ký tự OEM.
Trị trả về: Từ thấp chứa mã quét. Từ cao đưa ra các cờ hiệu thông qua các bit. Bit 0 nếu bằng 1, phím Shift bị nhấn. Bit 1 là 1 cho biết Ctrl bị nhấn.
Tham số kèm:
WOemChar - Giá trị ASCII của ký tự cần chuyển đổi.

**26. Declare Function SetKeyboardState Lib "user32" Alias "SetKeyboardState" (lppbKeyState As Byte) As Long**

Công dụng: Đặt trạng thái phím ảo. Thường sử dụng để đặt trạng thái phím CaspLock, NumLock, ScrollLock.
Tham số kèm:
lppbKeyState - Biến con trỏ 256 byte chứa trạng thái bàn phím.

**27. Declare Function ToAscii Lib "user32" Alias "ToAscii" (ByVal uVirtKey As Long, ByVal uScanCode As Long, lpbKeyState As Byte, lpwTransKey As Long, ByVal fuState As Long) As Long**

Công dụng: Xác định trị ASCII của một phím ảo dựa trên các trạng thái phím Shift và phím điều khiển.
Trị trả về:
Nếu phím cần kiểm tra là phím chết, giá trị trả về là số âm. Trường hợp khác là một trong các giá trị sau:
0 - Phím ảo cần kiểm tra không được dịch trong trạng thái hiện hành của bàn phím.
1 - Một ký tự được copy vào buffer
2 - Hai ký tự được copy vào bufer, trường hợp này xảy ra khi một phím nhấn chết, và gõ một phím khác.
Tham số kèm:
UVirtKey - Phím ảo cần chuyển đổi.
UScanCode - Mã quét của phím hoặc phím cần chuyển đổi. Bit cao của giá trị được thiết lập nếu phím nhả.
LpbKeyState - Biến trỏ tới mảng 256 byte chỉ trạng thái bàn phím hiện tại. Mỗi byte trong mảng chứa trạng thái của một phím. Nếu bit cao của một byte được thiết lập là phím được nhấn xuống. Với bit thấp được thiết lập, tín hiệu được đảo là ON. Trường hợp này chỉ dùng cho CAPSLOCK. Còn SCROLLOCK và NUMLOCK bị bỏ qua.
LpwTransKey - Biến con trỏ chỉ tới buffer mà nhận giá trị kết quả sau khi dịch.
FuState - Cờ hiệu, trạng thái thực đơn được kích hoạt. Nếu là 1 là được kích hoạt, ngược lại 0.
28. Declare Function ToAsciiEx Lib "user32" Alias "ToAsciiEx" (ByVal uVirtKey As Long, ByVal uScanCode As Long, lpKeyState As Byte, lpChar As Integer, ByVal uFlags As Long, ByVal dwhkl As Long) As Long
Như trên riêng dwhkl dùng để định vị trí nhập, dùng cho chuyển mã.Tham số này nhận được bất cứ vị trí định vị nào trước đó đã được trả về từ hàm LoadKeyboardLayout

**29. Declare Function VkKeyScan Lib "user32" Alias "VkKeyScanA" (ByVal cChar As Byte) As Integer**

Công dụng: Dịch ký tự phím ảo tùy thuộc trạng thái phím shift hiện tại.
Trị trả về: Nếu hàm thành công, byte thấp của giá trị trả về chứa mã phím ảo, và byte cao chứa trạng thái Shift, căn cứ vào bảng sau
Bit

Nghĩa là
1
Bất cứ SHIFT key bị nhấn
2
Bất cứ CTRL key bị nhấn.
4
Bất cứ ALT key bị nhấn.
8
The Hankaku key bị nhấn
16
Dành riêng (Được định nghĩa bởi driver).
32
Dành riêng (Được định nghĩa bởi driver).

Nếu hàm không tìm thấy phím nào để dịch thì bỏ qua mã ký tựcả hai byte thấp và cao đều chứa –1.

**30. Declare Function VkKeyScanEx Lib "user32" Alias "VkKeyScanExA" (ByVal ch As Byte, ByVal dwhkl As Long) As Integer**

Như trên riêng dwhkl dùng để định vị trí nhập, dùng cho chuyển mã.Tham số này nhận được bất cứ vị trí định vị nào trước đó đã được trả về từ hàm LoadKeyboardLayout

**3. Tiếp theo là các hàm điều khiển nhập liệu:**

**31. Declare Function GetInputState Lib "user32" Alias "GetInputState" () As Long**

Công dụng: Nhận trạng thái nhập liệu, kiểm tra xem có tình huống chuột hoặc bàn phím đang chờ xử lý không.
Trị trả về: True - Nếu có trường hợp cần xử lý đang chờ.

**32. Declare Function GetQueueStatus Lib "user32" Alias "GetQueueStatus" (ByVal fuFlags As Long) As Long**

Công dụng: Nhận loại chỉ lệnh cần xử lý trong hàng đợi đợi ứng dụng.
Trị trả về: Từ cao là tập cờ 16 bit, khai báo các chỉ lệnh ở hàng đợi. Từ thấp cho biết các loại chỉ lệnh được thêm vào.
Tham số kèm:
FUFlags - Tập cờ yêu cầu cần kiểm tra chỉ lệnh là một từ thông qua các bit. Là một trong các hằng số sau:
QS_ALLEVENTS = Tất cả các mức (QS_INPUT hoặc QS_POSTMESSAGE hoặc QS_TIMER hoặc QS_PAINT hoặc QS_HOTKEY)
QS_ALLINPUT = Tấ cả phần nhập liệu (QS_SENDMESSAGE hoặc QS_PAINT hoặc QS_TIMER hoặc QS_POSTMESSAGE Hoặc QS_MOUSEBUTTON Hoặc QS_MOUSEMOVE Hoặc QS_HOTKEY Hoặc QS_KEY)
QS_HOTKEY - Phím nóng.
QS_INPUT - Phần nhập (QS_MOUSE Hoặc QS_KEY)
QS_KEY - Các chỉ lệnh phím.
QS_MOUSE = Các chỉ lệnh chuột (QS_MOUSEMOVE Hoặc QS_MOUSEBUTTON)
QS_MOUSEBUTTON - Chỉ lệnh liên quan đến nút chuột.
QS_MOUSEMOVE - Di chuyển bằng chuột
QS_PAINT - Chỉ lệnh vẽ.
QS_POSTMESSAGE - Chỉ lệnh được phát đi.
QS_SENDMESSAGE - Chỉ lệnh được chuyển từ ứng dụng khác.
QS_TIMER - Timer.

**33. Declare Function GetQueuedCompletionStatus Lib "kernel32" Alias "GetQueuedCompletionStatus" (ByVal CompletionPort As Long, lpNumberOfBytesTransferred As Long, lpCompletionKey As Long, lpOverlapped As Long, ByVal dwMilliseconds As Long) As**

**Long**

Công dụng: Nhận trạng thái chỉ lệnh đã hoàn thành xếp hàng trong hàng đợi hệ thống
Trị trả về: @
Tham số kèm:
CompletionPort - Cán cổng hoàn thành. Để tạo một cổng hoàn thành sử dụng hàm CreateIoCompletionPort
LpNumberOfBytesTransferred - Biến con trỏ chỉ vào một biến nhận về số byte trao đổi trong tác vụ vào ra đã hoàn thành.
LpCompletionKey - Biến con trỏ chỉ vào biến nhận về phím đã hoàn thành tác vụ vào ra. Một phím hoàn thành là một giá trị nhận từ hàm CreateIoCompletionPort
LpOverlapped - Biến con trỏ chỉ tới biến ghi lại vị trí của địa chỉ cấu trúc OVERLAPPED được thiết lapaj khi tác vụ hoàn thành được bắt đầu.
DwMilliseconds - Số mili giây gọi để chờ cho tác vụ hoàn thành.

**34. Declare Function IsDBCSLeadByte Lib "kernel32" Alias "IsDBCSLeadByte" (ByVal bTestChar As Byte) As LongDeclare**

**35. Function IsDBCSLeadByte Lib "kernel32" Alias "IsDBCSLeadByte" (ByVal TestChar As Byte) As Long**

Công dụng: Kiểm tra xem có phải là ký tự đầu tiên trong bộ ký tự 2 byte.
Trị trả về: True nếu là byte đầu của ký tự thuộc bộ ký tự 2 byte.
Tham số kèm:
BTestChar - Ký tự cần kiểm tra.

---

**o0o--truongphu--o0o**

.....
Ghé thăm:
**Chuyện Linh Tinh**

**truongphu**
VIP

## Cửa sổ - Khái niệm ôm đồm của Windows API

T.Sáu 20/03/2009 4:44 pm

**• Cửa sổ - Khái niệm ôm đồm của Windows API**

*(Bài sau đây là copy từ địa chỉ trên, có lược bỏ phần đầu)*
*(Bài API với Registry tôi cũng lược bỏ)*

Thật kỳ quặc khi chính Microsoft đưa ra khái niệm Window khi lập trình API, mà lẽ ra phải gọi đó là các Object hoặc Control. Khi nói đến cửa sổ xin bạn lưu ý nó không chỉ là Form, mà còn có thể là bất cứ một đối tượng độc lập nào trong hộp Tool Box như các thanh cuộn (Scroll Bar), các hộp Text Box... ngay cả chính các biểu tượng Icon cũng được Win API coi là các cửa sổ (!?!). Ngoài ra còn có loại cửa sổ âm thầm mà không nhìn thấy trên màn hình.
Tất cả các cửa sổ kiểu trên đều thuộc vào các lớp. Mỗi lớp đều có các tính chất riêng. Có nghĩa là khi ta đổi vị trí của một cửa sổ từ lớp này sang lớp khác ngoài các tính chất riêng đặc thù của nó, còn các tính chất chung theo lớp nó sẽ được tiếp nhận ngay các tính năng của lớp mới và giã từ những tính chất của lớp cũ mà nó phụ thuộc.
Tìm hiểu các lớp hệ thống thường được sử dụng và đã có sẵn trong Windows (Xin lưu ý tên lớp không có dấu cách)
**Tên lớp**
Mô tả
**BUTTON**
Dùng cho các nút lệnh (Command button), nút chọn (Option button), nút kiểm tra (Check box)

**COMBOBOX**
Dùng cho hộp Combo box
**EDIT**
Dùng cho các Edit Control
**LISTBOX**
Dùng cho các List Box
**SCROLLBAR**
Dùng cho các thanh cuộn
**STATIC**
Dùng cho các cửa sổ hiển thị văn bản
**MDICLIENT**
Dùng cho các cửa sổ giao diện nhiều tài liệu MDI

• Xin quan sát các tính chất của mỗi lớp:
Tính chất
Công dụng
**Class Style** (Kiểu lớp)
Thiết đặt các thuộc tính mẫu của mỗi loại cửa sổ trong lớp.
**Class Function**
(Hàm lớp)
Các hàm mặc định của lớp.
**Instance** (Thể hiện)
Mô tả phiên bản sở hữu lớp. Thường là lớp hệ thống hoặc thư viện.
**Icon** (Biểu tượng)
Mô tả biểu tượng mặc nhiên trên Desktop khi cửa sổ của lớp này thu nhỏ Minimize.
**Cursor** (Con trỏ)
Mô tả con trỏ mặc nhiên khi chuột được định vị trên cửa sổ lớp này.
**Background** (Nền)
Mô tả màu nền mặc nhiên đối với các cửa sổ trong lớp này.
**Menu** (Thực đơn)
Mô tả thực đơn mặc nhiên đối với cửa sổ thuộc lớp này.

Bạn có thể thay đổi các tính chất của một lớp trong chương trình của bạn khi chương trình thi hành. Bạn có thể điều chỉnh các tính chất này. Các giá trị tính chất ban đầu của lớp sẽ phục hồi lại tuỳ thuộc vào thời điểm bạn unload bỏ lớp mà bạn đã điều chỉnh. Xem sơ đồ sau:
Thời điểm bạn Unload bỏ lớp mà bạn đã thay đổi tuỳ thuộc vào lớp đó bị thay đổi bởi ứng dụng, thì bạn nên khôi phục nó về ban đầu. Còn các lớp do DLL định nghĩa thì thường được unload khi các ứng dụng sử dụng DLL kết thúc. Khi Windows nạp nó sẽ đưa về các lớp về tình trạng ban đầu.

**• Subclassing**

Xin bạn xem lại bài viết API khám phá từ A đến Z phần 2, tôi đã nói rõ việc xử lý của các hàm Window. Mỗi lớp có hàm Window mặc nhiên để sử dụng cho mọi cửa sổ trong lớp. Một cửa sổ không nhất thiết phải sử dụng hàm mặc nhiên này, ta có thể tạo một lớp con (Subclassing) của một cửa sổ. Xây dựng các tính chất mới trên lớp con này. Hàm lớp mới có thể trực tiếp xử lý một số chỉ lệnh và chuyển giao các chỉ lệnh message cho lớp hiện có.
Nghĩa là tất cả các cửa sổ được tạo trên lớp mới này sẽ có hàm Window mặc nhiên cộng với các chức năng mới được tạo ra bởi lớp con.
Đối với Kỹ thuật Subclassing một Window để chặn tất cả các chỉ lệnh gửi đến nó có thể chọn các phương án:
• Kiểm tra message để biết
• Can thiệp vào message trước khi để nó đến đích.
• Bẫy một message sau khi Window gốc đã xử lý, thay đổi kết quả nó trả về cho ứng dụng gọi hay hệ điều hành.
• Can thiệp một message, tự xử lý nó. Không đưa nó cho Window gốc. Thay Window gốc làm mọi việc.
Để một thủ tục kết hợp với mỗi Window và xử lý tất cả các chỉ lệnh message đến ta thường dùng hàm SendMessage hay PostMessage.

Ta xét ví dụ sau để hiểu rõ về API, ta không xét về lập trình thông thường Visual Basic mà chỉ xét đến bản chất API qua ví dụ nhỏ này thôi.

Giả sử bạn tạo một form con như sau: Có Text Box với Name =TxtPass, Nút lệnh 1 có Name =CmdOK, nút lệnh 2 có Name=CmdCancel.
Ta viết lệnh để tìm nhanh chóng chuỗi bên trong hộp TxtPass.

**MÃ: CHỌN HẾT**

```
Option Explicit
'Khai báo
Public Const LB_FINDSTRINGEXACT = &H1A2
Declare Function GetFocus Lib "user32" Alias "GetFocus" () As Long
Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hwnd As Long, ByVal wMsg As Long, ByVal wParam As Long, lParam As Any) As Long
Sub cmdCancel_Click()
End
End Sub
Sub CmdOK_Click()
Dim hw%, chuoitrave&
TxtPass.SetForcus
hw=txtPass.hwnd
```

Từ VB5 đến VB6 Kỹ thuật subclassing được đề nghị thông qua từ khóa địa chỉ AddressOf. Thông thường để sử dụng ta phải làm các bước:
1. Chuẩn bị thủ tục thay thế thủ tục Window.
2. Ghi nhớ địa chỉ cũ bằng hàm SetWindowLong, đồng thời đặt địa chỉ mới vào địa chỉ của thủ tục thay thế.

**MÃ: CHỌN HẾT**

```
DiaChiCu= SetWindowLong(hWnd, GWL_WNDPROC, AddressOf Tên_Thủ_tục_Thay_Thế)
```

Sẽ có bạn hỏi AddressOf Tên_Thủ_tục_Thay_Thế là gì? Điều này ta không quan tâm vì khi nạp vào RAM, hệ thống sẽ tự điền và thay thế giúp cho chúng ta, là giá trị tuỳ theo máy cũng như tuỳ theo chương trình của mình, thế mới hay chứ.
Hàm này có khai báo chuẩn như sau:

**MÃ: CHỌN HẾT**

```
Declare Function SetWindowLong Lib "user32" Alias "SetWindowLongA" (ByVal hwnd As Long, ByVal nIndex As Long, ByVal dwNewLong As Long) As Long
```

Trong đó:
Hwnd - Cán của Window cần đổi thuộc tính
nIndex chọn một trong các hằng sau tuỳ theo mục đích:
Const GWL_HINSTANCE = (-6) - Handle của minh hoạ làm chủ Window
Const GWL_EXSTYLE = (-20) - Kiểu mở rộng của Window
Const GWL_STYLE = (-16) - Kiểu Window
Const GWL_ID = (-12) - ID của một Window con trong một hộp thoại
Const GWL_USERDATA = (-21) - Được định nghĩa bởi ứng dụng
Const GWL_HWNDPARENT = (-8) - Cán của Window cha.
Const GWL_WNDPROC = (-4) - Địa chỉ của thủ tục Window
Const DWL_DLGPROC = 4 - Địa chỉ hàm Dialog của Window
Const DWL_MSGRESULT = 0 - Giá trị trả về của thông điệp được xử lý trong hàm Dialog
Const DWL_USER = 8 - Được định nghĩa bởi ứng dụng.

Đối với tính chất GWL_WNDPROC, hàm SetWindowLong không chỉ đặt một giá trị mới mà còn trả về địa chỉ trước của đề mục đó. Ta lưu địa chỉ cũ để khi hồi phục lại khi kết thúc bằng chính hàm này.
SetWindowLong hWnd, GWL_WNDPROC, DiaChiCu
Những chỉ lệnh message gửi đến cho Window nó sẽ đưa vào cho thủ tục thay thế của bạn. Nếu thủ tục

thay thế của bạn muốn thủ tục cũ thực hiện thì dùng hàm gọi thủ tục cũ CallWindowProc như sau:
ViecChoCuLam = CallWindowProc(DiaChiCu, hwnd, uMsg, wParam, lParam)
Trong đó biến ViecChoCuLam là bạn khịa ra để việc gọi hàm thực hiện cho thuận lợi.
Xin lưu ý việc khôi phục địa chỉ phải được thực hiện tránh quên và phải đặt trước lệnh End. Nếu không sẽ treo và dẫn tới nguy hiểm.
Các ví dụ chuyên sâu về SubClassing xin download từ Web Lê Hoàn.
Chúc các bạn đạt được ý nguyện mong muốn.

Hết Copy. Các phần lược bỏ, xin xem địa chỉ gốc
Chiều thứ 6 20/3

---

**o0o--truongphu--o0o**

.....
Ghé thăm:
**Chuyện Linh Tinh**

**truongphu**
VIP

## Hơn 60 hàm API liên qua cửa sổ

CN 22/03/2009 9:08 am

Bổ sung
**56. Hàm AnimateWindow**

Hàm AnimateWindow cho phép Bạn thực hiện các hiệu ứng đặc biệt khi mở hoặc đóng một cửa sổ ứng dụng. Có 3 kiểu hiệu ứng là : roll, slide, và hiệu ứng pha trộn (alpha-blended fade).
Yêu cầu: Windows 2000, Windows 98 trở lên.

Dec:

> Declare Function AnimateWindow Lib "user32" (ByVal hwnd As Long, ByVal dwTime As Long, ByVal dwFlags As Long) As Boolean

Tham số:

• hwnd
[in] Handle cửa sổ cần tạo hiệu ứng và việc gọi hiệu ứng phải ở trong chính cửa sổ này.

• dwTime
[in] Xác định thời gian thực hiện hiệu ứng (bằng mili giây). Thông thường là 200.

• dwFlags
[in] Xác định kiểu hiệu ứng. Tham số này có thể gồm một hay nhiều giá trị sau:

AW_SLIDE
Sử dụng hiệu ứng slide. Mặc định hiệu ứng roll sẽ được sử dụng. Giá trị này sẽ được bỏ qua nếu sử dụng AW_CENTER.

AW_ACTIVATE
Kích hoạt window. Không dùng giá trị này cùng với AW_HIDE.

AW_BLEND

Sử dụng hiệu ứng fade. Giá trị này được dùng chỉ khi hwnd là top-level window.

AW_HIDE
Ẩn window. Mặc định là cửa sổ được show.

AW_CENTER
Tạo window xuất hiện với hiệu ứng thu vụn từng mảnh nhỏ vào trong nếu AW_HIDE được dùng hoặc mở rộng ra ngoài nếu không dùng AW_HIDE.

AW_HOR_POSITIVE
Tạo hiệu ứng hoạt hình cho window từ trái sang phải. Giá trị này có thể được dùng với hiệu ứng roll hoặc slide và nó bị bỏ qua khi sử dụng với AW_CENTER hoặc AW_BLEND.

AW_HOR_NEGATIVE
Tương tự như AW_HOR_POSITIVE nhưng từ phải sang trái.

AW_VER_POSITIVE
Tương tự như AW_HOR_POSITIVE nhưng từ trên xuống dưới.

AW_VER_NEGATIVE
Tương tự như AW_VER_NEGATIVE nhưng từ dưới lên trên.

Giá trị trả về:
Nếu thành công giá trị khác không (nonzero) sẽ được trả về và ngược lại.
Hàm sẽ thực hiện không thành công trong các trường hợp sau:
{/i} Cửa sổ sử dụng window region; hoặc đã visible và Bạn đang cố gắng show nó lên; hoặc window đã hidden và Bạn đang hide nó một lần nữa.
Để có thêm thông tin về các lỗi hãy gọi hàm GetLastError.{/i}

**Ví dụ với VB6:**

MÃ: CHỌN HẾT

```
Const AW_HOR_POSITIVE = &H1
Const AW_HOR_NEGATIVE = &H2
Const AW_VER_POSITIVE = &H4
Const AW_VER_NEGATIVE = &H8
Const AW_CENTER = &H10
Const AW_HIDE = &H10000
Const AW_ACTIVATE = &H20000
Const AW_SLIDE = &H40000
Const AW_BLEND = &H80000

Private Declare Function AnimateWindow Lib "user32" (ByVal hwnd As Long, ByVal dwTime As Long, ByVal dwFlags As Long) As Boolean
Private Sub Form_Load()
```

5. (Đã có)
**Hàm BeginDeferWindowPos**

Hàm BeginDeferWindowPos sẽ cấp phát bộ nhớ cho một multiple-window- position structure và lấy handle trả về structure đó.
Yêu cầu: Windows NT 3.1, Windows 95

Dec:

> Declare Function BeginDeferWindowPos Lib "user32" Alias "BeginDeferWindowPos" (ByVal nNumWindows As Long) As Long

Tham số:

• nNumWindows
Xác định initial number của windows dùng để lưu trữ tông tin về position. Hàm DeferWindowPos làm tăng thêm kích thước của structure, nếu cần.

Giá trị trả về:
Nếu thành công, giá trị trả về sẽ xác định the multiple-window - position structure. Nếu không đủ tài nguyên hệ thống để cấp phát, giá trị NULL sẽ được trả về.

Ví dụ bằng VB6:

**MÃ: CHỌN HẾT**

```
Const WS_BORDER = &H800000
Const WS_DLGFRAME = &H400000
Const WS_THICKFRAME = &H40000
Const WS_CAPTION = &HC00000                  '  WS_BORDER Or WS_DLGFRAME
Const HWND_BOTTOM = 1
Const HWND_TOP = 0
Const HWND_TOPMOST = -1
Const HWND_NOTOPMOST = -2
Const SWP_SHOWWINDOW = &H40
Private Type RECT
        Left As Long
        Top As Long
        Right As Long
```

6. (Đã có trên)
**Hàm DeferWindowPos**

Hàm DeferWindowPos cập nhật multiple-window-position structure đã được xác định trước đó cho một cửa sổ được chỉ định. Hàm này sau đó sẽ trả về handle của structure đã được update. Hàm EndDeferWindowPos sử dụng các thông tin trong structure này để thay đổi đồng thời vị trí (position) và kích thước (size) của cửa sổ. Hàm BeginDeferWindowPos sẽ tạo structure này.
Yêu cầu: Windows NT 3.1, Windows 95 trở lên.

Dec:

> Declare Function DeferWindowPos Lib "user32" Alias "DeferWindowPos" (ByVal hWinPosInfo As Long, ByVal hwnd As Long, ByVal hWndInsertAfter As Long, ByVal x As Long, ByVal y As Long, ByVal cx As Long, ByVal cy As Long, ByVal wFlags As Long) As Long

Tham số:

• hWinPosInfo
Xác định một multiple-window - position structure chứa thông tin về kích thước và vị trí của một hoặc nhiều windows. Structure này được trả về bởi hàm BeginDeferWindowPos hoặc vừa mới gọi bằng hàm DeferWindowPos.

• hWnd

Xác định window đã được lưu trữ trong structure được update thông tin.

• hWndInsertAfter
Identifies the window that precedes the positioned window in the Z order. Tham số này phải là một handle cửa sổ hoặc một trong các giá trị sau:

HWND_BOTTOM
Đặt window xuống đáy (bottom) của Z order. Nếu tham số hWnd xác định một window ở vị trí cao nhất thì window sẽ mất vị trí đó và được thay thế vào vị trí bottom của tất cả các windows khác.

HWND_NOTOPMOST
Đặt window ở trên tất cả các non-topmost windows (điều này có nghĩa là ở đằng sau tất cả các topmost windows). Giá trị này sẽ không có tác dụng nếu window đã là một non-topmost window.

HWND_TOP
Đặt window vào vị trí cao nhất của Z order.

HWND_TOPMOST
Đặt window ở trên tất cả non-topmost windows. Window sẽ giữ vị trí topmost này của nó ngay cả khi nó bị deactive. Tham số này sẽ bị bỏ qua nếu SWP_NOZORDER được thiết lập trong tham số uFlags.

• x
Xác định x-coordinate của góc trái bên trên của cửa sổ.

• y
Xác định y-coordinate của góc trái bên trên của cửa sổ.

• cx
Xác định bề rộng mới của window, bằng pixels.

• cy
Xác định chiều cao mới của window, bằng pixels.

• uFlags
Xác định một sự kết hợp của các giá trị sau và sẽ tác động đến kích thước và vị trí của window:

SWP_DRAWFRAME
Vẽ một frame (được định nghĩa trong phần mô tả class của window) xung quanh window.
SWP_FRAMECHANGED
Gửi thông điệp WM_NCCALCSIZE đến window, ngay cả trong trường hợp khi kích thước của window không bị thay đổi. Nếu tham số này không được xác định, WM_NCCALCSIZE sẽ chỉ được gửi khi kích thước của window bị thay đổi.
SWP_HIDEWINDOW
Hides the window.
SWP_NOACTIVATE
Không kích hoạt (activate) window. Nếu tham số này không được thiết lập, window sẽ được kích hoạt và được chuyển lên thành top của các topmost hoặc non-topmost group khác (tùy thuộc và thiết lập của hWndInsertAfter parameter).
SWP_NOCOPYBITS
Discards the entire contents of the client area. Nếu tham số này not specified, contents hợp lệ của các client area sẽ được lưu trữ và dược copy vào trong client area sau khi window bị thay đổi kích thước hoặc vị trí.
SWP_NOMOVE
Giữ lại vị trí hiện tại (bỏ qua tham số X và Y).
SWP_NOOWNERZORDER
Không thay đổi vị trí của chính window trong Z order.
SWP_NOREDRAW
Không redraw lại các thay đổi. Nếu tham số này không được set, no repainting of any kind occurs. Điều

này tác động đến client area, các nonclient area (bao gồm cả title bar và scroll bars), và bất kỳ phần nào của parent window uncovered như kết quả tác động khi window được moved. Khi tham số này được thiết lập, ứng dụng phải được explicitly invalidate hoặc redraw any parts of the window and parent window cần được redrawing.
SWP_NOREPOSITION
Tương tự như SWP_NOOWNERZORDER.
SWP_NOSENDCHANGING
Ngăn ngừa window nhận thông điệp WM_WINDOWPOSCHANGING.
SWP_NOSIZE
Lưu giữ lại kích thước hiện tại (bỏ qua tham số cx và cy).
SWP_NOZORDER
Lưu giữ lại Z order hiện tại (bỏ qua tham số hWndInsertAfter).
SWP_SHOWWINDOW
Displays the window.

Giá trị trả về:
Giá trị trả về xác định multiple-window - position structure đã được câph nhật. Thẻ handle được trả về bằng hàm này có thể khác với thẻ handle gửi đến nó. Thẻ handle mới mà hàm này trả về sẽ được dùng trong lện gọi tiếp theo đến hàm DeferWindowPos hoặc hàm EndDeferWindowPos.
Nếu tài nguyên hệ thống không còn đủ để thực hiện, giá trị trả về sẽ là NULL.

Ví dụ VB6

**MÃ: CHỌN HẾT**

```
Const WS_BORDER = &H800000
Const WS_DLGFRAME = &H400000
Const WS_THICKFRAME = &H40000
Const WS_CAPTION = &HC00000                    '  WS_BORDER Or WS_DLGFRAME
Const HWND_BOTTOM = 1
Const HWND_TOP = 0
Const HWND_TOPMOST = -1
Const HWND_NOTOPMOST = -2
Const SWP_SHOWWINDOW = &H40
Private Type RECT
        Left As Long
        Top As Long
        Right As Long
```

**o0o--truongphu--o0o**

.....
Ghé thăm:
**Chuyện Linh Tinh**

truongphu
VIP

## Hơn 60 hàm API liên qua cửa sổ

CN 22/03/2009 9:15 am

13. Đã có trên
**Hàm CloseWindow**

Hàm CloseWindow sẽ thu nhỏ nhưng không destroy cửa sổ được xác định.
Yêu cầu: Windows NT 3.1, Windows 95 trở lên.

Dec:

> Declare Function CloseWindow Lib "user32" Alias "CloseWindow" (ByVal hwnd As Long) As Long

Tham số:
• hWnd
handle cửa sổ cần thu nhỏ.

Giá trị trả về:
Nonzero: nếu thành công, và zero nếu ngược lại.
Gọi hàm GetLastError để có thêm thông tin lỗi.

Ví dụ VB6

**MÃ: CHỌN HẾT**

```
1. Private Declare Function CloseWindow Lib "user32" (ByVal hwnd As Long) As Long
2. Private Sub Form_Load()
3.     CloseWindow Me.hwnd
4. End Sub
```

16. Đã có tren

**Hàm EnableWindow**

Hàm EnableWindow sẽ enables hoặc disables các tác động của mouse và keyboard lên window hoặc control đã xác định. Khi disabled, window sẽ không nhận bất kỳ tác động nào như click chuột hoặc nhấn phím, và khi enabled thì ngược lại.
Yêu cầu: Windows NT 3.1; Windows 95 trở lên.

Dec:

> Declare Function EnableWindow Lib "user32" Alias "EnableWindow" (ByVal hwnd As Long, ByVal fEnable As Long) As Long

Tham số:
• hWnd Xác định window sẽ được enabled hoặc disabled.

• bEnable Nếu tham số này là TRUE thì window sẽ enabled và ngược lại.

Giá trị trả về:
nonzero nếu thực hiện thành công, zero: không. Gọi hàm GetLastError để biết chi tiết lỗi.

Ví dụ VB6

**MÃ: CHỌN HẾT**

```
1. {i}'This project needs two command buttons{/i}
2. Private Declare Function IsWindowEnabled Lib "user32" (ByVal hwnd As Long) As Long
3. Private Declare Function EnableWindow Lib "user32" (ByVal hwnd As Long, ByVal fEnable As Long) As
   Long
4. Private Sub Command2_Click()
5.     ' Reverse the enabled status of Command1.  If the window is
6.     ' disabled, enable it; if it is enabled, disable it.
7.     Dim wasenabled As Long  ' receives enabled/disabled status of Command1
8.     Dim retval As Long  ' return value
9.     ' Determine if the window Command1 is currently enabled or not.
10.    wasenabled = IsWindowEnabled(Command1.hwnd)
11.    If wasenabled = 0 Then  ' if not enabled, enable it
```

```
12.         retval = EnableWindow(Command1.hwnd, 1)
```

8. Đã có trên
**Hàm EndDeferWindowPos**

Hàm EndDeferWindowPos function sẽ update đồng thời vị trí và kích thước của 1 hoặc nhiều window của chu kỳ đơn screen-refreshing.
Các hàm liên quan: BeginDeferWindowPos, DeferWindowPos
Yêu cầu: Windows NT 3.1; Windows 95 trở lên.

Dec:

> “
> Declare Function EndDeferWindowPos Lib "user32" Alias "EndDeferWindowPos" (ByVal hWinPosInfo As Long) As Long

Tham số:
• hWinPosInfo Xác định multiple-window - position structure có chứa thông tin về kích thước và vị trí của 1 hoặc nhiều window. Cấu trúc nội tại này sẽ được trả về bởi hàm BeginDeferWindowPos hoặc bởi hầu hết các lệnh vừa gọi đến hàm DeferWindowPos.

Giá trị trả về:
nonzero nếu thực hiện thành công, zero: không. Gọi hàm GetLastError để biết chi tiết lỗi.

Ví dụ VB6

**MÃ: CHỌN HẾT**

```
 1. Const WS_BORDER = &H800000
 2. Const WS_DLGFRAME = &H400000
 3. Const WS_THICKFRAME = &H40000
 4. Const WS_CAPTION = &HC00000    '  WS_BORDER Or WS_DLGFRAME
 5. Const HWND_BOTTOM = 1
 6. Const HWND_TOP = 0
 7. Const HWND_TOPMOST = -1
 8. Const HWND_NOTOPMOST = -2
 9. Const SWP_SHOWWINDOW = &H40
10. Private Type RECT
11.         Left As Long
12.         Top As Long
13.         Right As Long
```

17. Đã có trên

**Hàm EnumChildWindows**

Hàm EnumChildWindows sẽ liệt kê các windows con trực thuộc window cha đã được xác định bằng cách lần lượt gửi handle của mỗi cửa sổ con tới 1 hàm callback do ứng dụng định nghĩa. EnumChildWindows tiếp tục cho tới khi cửa sổ con cuối cùng được liệt kê hoặc hàm callback trả về FALSE.
Các hàm liên quan: EnumWindows
Yêu cầu: Windows NT 3.1; Windows 95 trở lên.
Dec:

> “
> Declare Function EnumChildWindows Lib "user32" Alias "EnumChildWindows" (ByVal hWndParent As Long, ByVal lpEnumFunc As Long, ByVal lParam As Long) As Long

Tham số:
• hWndParent Xác định window cha sẽ được liệt kê các window con.

• lpEnumFunc Địa chỉ hàm callback do ứng dụng định nghĩa. Xem thêm hàm EnumChildProc.

• lParam Xác định giá trị 32-bit do ứng dụng tùy chọn, làm tham số cho hàm callback

Giá trị trả về:
nonzero nếu thực hiện thành công, zero: không. Gọi hàm GetLastError để biết chi tiết lỗi.

Ví dụ VB6

**MÃ: CHỌN HẾT**

```
{i}'in a form{/i}
Private Sub Form_Load()
    Me.AutoRedraw = True
    EnumChildWindows GetDesktopWindow, AddressOf EnumChildProc, ByVal 0&
End Sub
'in a module
Declare Function GetDesktopWindow Lib "user32" () As Long
Declare Function EnumChildWindows Lib "user32" (ByVal hWndParent As Long, ByVal lpEnumFunc As Long,
ByVal lParam As Long) As Long
Declare Function GetWindowText Lib "user32" Alias "GetWindowTextA" (ByVal hwnd As Long, ByVal
lpString As String, ByVal cch As Long) As Long
Declare Function GetWindowTextLength Lib "user32" Alias "GetWindowTextLengthA" (ByVal hwnd As Long)
As Long
```

22. Đã có trên
**Hàm FlashWindow**

Hàm FlashWindow dùng để flash window được chỉ ra.
Các hàm liên quan: FlashWindowEx
Yêu cầu: Windows NT 3.1; Windows 95 trở lên.

Dec:

> Declare Function FlashWindow Lib "user32" Alias "FlashWindow" (ByVal hwnd As Long, ByVal bInvert As Long) As Long

Tham số:
• hWnd Xác định window sẽ được flash.

• bInvert Trạng thái của window: sẽ được flash hoặc quay về trạng thái nguyên thủy của nó. Window được flash sẽ được chuyển từ trạng thái này sang trạng thái khác nếu tham số này được đặt TRUE. Nếu là FALSE, window sẽ được quay về tình trạng ban đầu của nó (active hoặc inactive).

Giá trị trả về:
Giá trị trả về sẽ xác định tình trạng của window khi trước gọi hàm FlashWindow. Nếu window đã được active trước khi gọi hàm này, giá trị nonzero sẽ được trả về.

Ví dụ VB6

**MÃ: CHỌN HẾT**

```
'This project needs a timer,Interval 1000

'In general section
```

```
4. Private Declare Function FlashWindow Lib "user32" (ByVal hwnd As Long, ByVal bInvert As Long) As Long
5. Const Invert = 1
6.
7. Private Sub Timer1_Timer()
8.     'Flash the window
9.     FlashWindow Me.hwnd, Invert
10. End Sub
```

## 57. Hàm FlashWindowEx

Hàm FlashWindowEx sẽ flash window được chỉ ra cụ thể. Khi dùng hàm này, Bạn có thể thời gian mà hệ thống sẽ hiển thị windows, trong khi hàm FlashWindow sẽ chỉ hiển thị window mỗi một lần.
Các hàm liên quan: FlashWindow
Yêu cầu: Windows 2000; Windows 98 trở lên.

Dec:

> “
> Declare Function FlashWindowEx Lib "user32" (pfwi As FLASHWINFO) As Boolean

Tham số:
• pfwi [in] Con trỏ trỏ đến cấu trúc FLASHWINFO.

Giá trị trả về:
Giá trị trả về sẽ xác định tình trạng của window khi trước gọi hàm FlashWindowEx. Nếu window caption đã xác định được kích hoạt trước khi gọi thì hàm trả về giá trị nonzero, ngược lại là zero.

Ví dụ VB6

MÃ: CHỌN HẾT

```
1. Const FLASHW_STOP = 0 'Stop flashing. The system restores the window to its original state.
2. Const FLASHW_CAPTION = &H1 'Flash the window caption.
3. Const FLASHW_TRAY = &H2 'Flash the taskbar button.
4. Const FLASHW_ALL = (FLASHW_CAPTION Or FLASHW_TRAY) 'Flash both the window caption and taskbar
   button. This is equivalent to setting the FLASHW_CAPTION Or FLASHW_TRAY flags.
5. Const FLASHW_TIMER = &H4 'Flash continuously, until the FLASHW_STOP flag is set.
6. Const FLASHW_TIMERNOFG = &HC 'Flash continuously until the window comes to the foreground.
7. Private Type FLASHWINFO
8.     cbSize As Long
9.     hwnd As Long
10.    dwFlags As Long
11.    uCount As Long
12.    dwTimeout As Long
```

24. Đã có trên
**Hàm GetClassInfo**

Hàm GetClassInfo dùng lấy thông tin về một window class.
Các hàm liên quan: GetClassInfoEx, GetClassName
Yêu cầu: Windows NT 3.1; Windows 95 trở lên.

Dec:

> Declare Function GetClassInfo Lib "user32" Alias "GetClassInfoA" (ByVal hInstance As Long, ByVal lpClassName As String, lpWndClass As WNDCLASS) As Long

Tham số:
• hInstance Xác định application đã tạo nên class. Để lấy thông tin về các class được định nghĩa bởi Windows (ví như buttons list boxes), hãy thiết lập giá trị của tham số này về NULL.

• lpClassName Chỉ đến chuỗi (kết thúc bằng null) có chứa class name và phải được đăng ký trước đó hoặc là một class đã được đăng ký trước bởi hàm RegisterClass. Tham số có thể là một integer. Nếu vậy, nó phải là một global được tạo bởi hàm GlobalAddAtom trước đó. Giá trị 16-bít (nhỏ hơn 0xC000) phải đặt ở từ thấp (low-order word) của lpClassName; còn từ nhớ cao phải chứa zero.

• lpWndClass Trỏ đến cấu trúc WNDCLASS sẽ nhận thông tin về class.

Giá trị trả về:
Nếu hàm tìm thấy và copy được dữ liệu trong class, giá trị nonzero được trả về, ngược lại trả về zero. Xem thêm lỗi bằng hàm GetLastError.

Ví dụ VB6

**MÃ: CHỌN HẾT**

```
Private Type WNDCLASS
    style As Long
    lpfnwndproc As Long
    cbClsextra As Long
    cbWndExtra2 As Long
    hInstance As Long
    hIcon As Long
    hCursor As Long
    hbrBackground As Long
    lpszMenuName As String
    lpszClassName As String
End Type
Private Declare Function GetSysColor Lib "user32" (ByVal nIndex As Long) As Long
```

## 58. Hàm GetClassInfoEx

Hàm GetClassInfoEx nhận thông tin về window class, bao gồm cả handle của small icon tích hợp với window class.
Các hàm liên quan: GetClassInfo
Yêu cầu: Windows NT 3.1; Windows 95 trở lên.

Dec:

> Declare Function GetClassInfoEx Lib "user32" Alias "GetClassInfoExA" (ByVal hInstance As Long, ByVal lpClassName As String, lpWndClass As WNDCLASSEX) As Long

Tham số:
• hinst xác định application đã tạo class đó. Để nhận thông tin về các class của Windows (ví như buttons hay list boxes vậy), hãy đặt tham số này giá trị NULL.

• lpszClass Trỏ đến string (kết thúc bằng null) chứa class name. Tham số này tương tự như trong hàm GetClassInfo vậy.

• lpwcx Trỏ đến WNDCLASSEX structure sẽ nhận infor về class.

Giá trị trả về:
Như hàm GetClassInfo vậy.

Ví dụ VB6

**MÃ: CHỌN HẾT**

```
Private Type WNDCLASSEX
    cbSize As Long
    style As Long
    lpfnWndProc As Long
    cbClsExtra As Long
    cbWndExtra As Long
    hInstance As Long
    hIcon As Long
    hCursor As Long
    hbrBackground As Long
    lpszMenuName As String
    lpszClassName As String
    hIconSm As Long
```

**o0o--truongphu--o0o**

.....
Ghé thăm:
**Chuyện Linh Tinh**

truongphu
VIP

## Hơn 60 hàm API liên qua cửa sổ

CN 22/03/2009 9:20 am

### 59. Hàm GetClassName

Hàm GetClassName nhận về name of the class mà window đã chỉ ra trực thuộc.
Các hàm liên quan: None
Yêu cầu: Windows NT 3.1; Windows 95 trở lên.

Dec:

> Declare Function GetClassName Lib "user32" Alias "GetClassNameA" (ByVal hwnd As Long, ByVal lpClassName As String, ByVal nMaxCount As Long) As Long

Tham số:
• hWnd Xác định window và gián tiếp qua đó xác định class mà window đó trực thuộc.

• lpClassName Trỏ đến bộ đêm để nhận class name string.

• nMaxCount Xác định chiều dài, bằng ký tự, của bộ đệm mà tham số lpClassName trỏ đến. Class name string được cắt gọn nếu nó dài hơn bộ đệm.

Giá trị trả về:
Nếu thành công, giá trị trả về là số ký tự được copy đến bộ đệm, nếu không trả về zero. Xem thêm lỗi bằng hàm GetLastError.

Ví dụ VB6

**MÃ: CHỌN HẾT**

```
1. Private Declare Function FindWindow Lib "user32" Alias "FindWindowA" (ByVal lpClassName As String,
   ByVal lpWindowName As String) As Long
2. Private Declare Function PostMessage Lib "user32" Alias "PostMessageA" (ByVal hwnd As Long, ByVal
   wMsg As Long, ByVal wParam As Long, lParam As Any) As Long
3. Private Declare Function GetClassName Lib "user32" Alias "GetClassNameA" (ByVal hwnd As Long, ByVal
   lpClassName As String, ByVal nMaxCount As Long) As Long
4. Private Declare Function ShowWindow Lib "user32" (ByVal hwnd As Long, ByVal nCmdShow As Long) As
   Long
5. Const SW_SHOWNORMAL = 1
6. Const WM_CLOSE = &H10
7. Const gcClassnameMSWord = "OpusApp"
8. Const gcClassnameMSExcel = "XLMAIN"
9. Const gcClassnameMSIExplorer = "IEFrame"
```

## 60. Hàm GetTitleBarInfo

Hàm GetTitleBarInfo nhận các thông tin về title bar đã được chỉ định.
Các hàm liên quan: GetWindowRect
Yêu cầu: Windows NT 4 SP6; Windows 98 trở lên.

Dec:

> Declare Function GetTitleBarInfo Lib "user32.dll" (ByVal hwnd As Long, ByRef pti As TITLEBARINFO) As Long

Tham số:
• hwnd
[in] Handle của title bar sẽ được lấy thông tin.

• pti
[out] Địa chỉ cấu trúc TITLEBARINFO để nhận thông tin. Chú ý rằng Bạn phải set TITLEBARINFO.cbSize về sizeof(TITLEBARINFO) trước khi gọi hàm này.

Giá trị trả về:
Nếu thành công, giá trị trả về là nonzero, nếu không trả về zero. Xem thêm lỗi bằng hàm GetLastError.

Ví dụ VB6

**MÃ: CHỌN HẾT**

```
1. Option Explicit
2. Private Const STATE_SYSTEM_FOCUSABLE = &H100000
3. Private Const STATE_SYSTEM_INVISIBLE = &H8000
4. Private Const STATE_SYSTEM_OFFSCREEN = &H10000
5. Private Const STATE_SYSTEM_UNAVAILABLE = &H1
6. Private Const STATE_SYSTEM_PRESSED = &H8
7. Private Const CCHILDREN_TITLEBAR = 5
8. Private Type RECT
9.     Left As Long
10.    Top As Long
11.    Right As Long
12.    Bottom As Long
13. End Type
```

26. Đã có trên
**Hàm GetWindowLong**

Hàm GetWindowLong nhận các thông tin về cửa sổ đã được chỉ định.

Các hàm liên quan: SetWindowLong
Yêu cầu: Windows NT 3.1; Windows 95 trở lên.

Dec:

> Declare Function GetWindowLong Lib "user32" Alias "GetWindowLongA" (ByVal hwnd As Long, ByVal nIndex As Long) As Long

Tham số:
• hWnd định nghĩa the window và, qua đó, xác định class mà window trực thuộc.

• nIndex Xác định zero-based trỏ đến giá trị sẽ được nhận về. Các giá trị hợp lệ là trong giới hạn từ zero đến số byte của bộ nhớ window mở rộng, trừ 4; ví dụ, nếu Bạn xác định 12 hoặc nhiều hơn nữa các byte của bộ nhớ mở rộng, giá trị 8 sẽ chỉ đến (index) số nguyên (32-bit) thứ 3. Để nhận các giá trị bất kỳ nào khác hãy dùng các tham số sau:

GWL_EXSTYLE Nhận về các kiểu window mở rộng (extended window).
GWL_STYLE Nhận về các kiểu window thông thường.
GWL_WNDPROC Nhận về địa chỉ của một thủ tục cửa sổ (window procedure), hoặc một handle mô tả địa chỉ của window procedure. Bạn phải dùng hàm CallWindowProc để gọi window procedure.
GWL_HINSTANCE Nhận về handle của instance của ứng dụng.
GWL_HWNDPARENT Nhận về handle của cửa sổ cha, nếu có.
GWL_ID Nhận về định danh của window.
GWL_USERDATA Nhận về giá trị 32-bit có liên quan đến window. Mỗi window có một giá trị 32-bit tương ứng được dự định dùng cho ứng dụng đã tạo ra window.

Các giá trị sau đây được coi là hợp lệ khi tham số hWnd xác định một dialog box:
DWL_DLGPROC Nhận về địa chỉ của thủ tục dialog box, hoặc một handle mổ tả địa chỉ của dialog box procedure. Bạn phải dùng hàm CallWindowProc để gọi một dialog box procedure.
DWL_MSGRESULT Nhận giá trị trả về của một message processed trong dialog box procedure.
DWL_USER Nhận về các thông tin mở rộng riêng biệt của ứng dụng, như các handle hoặc các địa chỉ.

Giá trị trả về:
Nếu thành công, giá trị 32-bit đã yêu cầu được trả về, nếu không trả về zero. Xem thêm lỗi bằng hàm GetLastError.

Ví dụ VB6

**MÃ: CHỌN HẾT**

```
1. 'This project needs a TextBox
2. Private Declare Function GetWindowLong Lib "user32" Alias "GetWindowLongA" (ByVal hwnd As Long,
   ByVal nIndex As Long) As Long
3. Private Declare Function SetWindowLong Lib "user32" Alias "SetWindowLongA" (ByVal hwnd As Long,
   ByVal nIndex As Long, ByVal dwNewLong As Long) As Long
4. Const GWL_STYLE = (-16)
5. Const ES_NUMBER = &H2000&
6. Public Sub SetNumber(NumberText As TextBox, Flag As Boolean)
7.     Dim curstyle As Long, newstyle As Long
8.
9.     'retrieve the window style
```

```
10.     curstyle = GetWindowLong(NumberText.hwnd, GWL_STYLE)
11.
```

**'This project needs one form**
**' Also set StartupObject to 'Sub Main'**
**' (-> Project Properties -> General Tab -> Startup Object)**

**MÃ: CHỌN HẾT**

```
1. '---- Declarations
2.
3. Declare Function RegisterClass Lib "user32" Alias "RegisterClassA" (Class As WNDCLASS) As Long
4. Declare Function UnregisterClass Lib "user32" Alias "UnregisterClassA" (ByVal lpClassName As String,
   ByVal hInstance As Long) As Long
5. Declare Function CreateWindowEx Lib "user32" Alias "CreateWindowExA" (ByVal dwExStyle As Long, ByVal
   lpClassName As String, ByVal lpWindowName As String, ByVal dwStyle As Long, ByVal x As Long, ByVal y
   As Long, ByVal nWidth As Long, ByVal nHeight As Long, ByVal hWndParent As Long, ByVal hMenu As Long,
   ByVal hInstance As Long, lpParam As Any) As Long
6. Declare Function DefWindowProc Lib "user32" Alias "DefWindowProcA" (ByVal hWnd As Long, ByVal wMsg
   As Long, ByVal wParam As Long, ByVal lParam As Long) As Long
7. Declare Sub PostQuitMessage Lib "user32" (ByVal nExitCode As Long)
8. Declare Function GetMessage Lib "user32" Alias "GetMessageA" (lpMsg As Msg, ByVal hWnd As Long,
```

## 61. Hàm GetWindowPlacement

Hàm GetWindowPlacement nhận các thông tin hiển thị trạng thái, các vị trí được restore, minimize, maximize của cửa sổ xác định.

Các hàm liên quan: SetWindowPlacement, SetWindowPos
Yêu cầu: Windows NT 3.1; Windows 95 trở lên.

Dec:

> “
> Declare Function GetWindowPlacement Lib "user32" (ByVal hwnd As Long, lpwndpl As WINDOWPLACEMENT) As Long

Tham số:
• hWnd
[in] Handle to the window.

• lpwndpl
[out] Địa chỉ của cấu trúc WINDOWPLACEMENT sẽ nhận trạng thái hiển thị và vị trí của window. Trước khi gọi hàm GetWindowPlacement, hayc đặt thành phần length của WINDOWPLACEMENT về sizeof(WINDOWPLACEMENT). Hàm GetWindowPlacement sẽ fail nếu lpwndpl->length không được đặt đúng.

Giá trị trả về:
Nếu thành công, giá trị nonzero được trả về, nếu không trả về zero. Xem thêm lỗi bằng hàm GetLastError.

Ví dụ VB6

**MÃ: CHỌN HẾT**

```
Private Const SW_MINIMIZE = 6
Private Type POINTAPI
        x As Long
        y As Long
End Type
Private Type RECT
        Left As Long
        Top As Long
        Right As Long
        Bottom As Long
End Type
Private Type WINDOWPLACEMENT
        Length As Long
```

28. Đã có trên

**Hàm GetWindowText**

Hàm GetWindowText sẽ copy đoạn text của window's title bar (nếu có) vào bộ đệm. Nếu window được chỉ định là một control, đoạn text của control sẽ được copy.

Các hàm liên quan: SetWindowText
Yêu cầu: Windows NT 3.1; Windows 95 trở lên.

Dec:

> Declare Function GetWindowText Lib "user32" Alias "GetWindowTextA" (ByVal hwnd As Long, ByVal lpString As String, ByVal cch As Long) As Long

Tham số:
• hWnd định nghĩa window/control có chứa đoạn text.

• lpString ĐỊa chỉ bộ đệm sẽ nhận đoạn text đó.

• nMaxCount Xác định số ký tự tối đa được copy vào bộ đệm, gồm cả ký tự NULL. Nếu đoạn text quá lớn, nó sẽ bị cắt gọn đi.

Giá trị trả về:
Nếu thành công, giá trị được trả về là chiều dài, tính bằng ký tự, của đoạn text được copy và không bao gồm ký tự kết thúc. Nếu window không có title bar hoặc text, hoặc nếu title bar empty, hoặc nếu window/control handle không hợp lệ, giá trị trả về là zero. Hàm này không thể lấy được đoạn text của một control chỉnh sửa được trong các ứng dụng khác. Xem thêm lỗi bằng hàm GetLastError.

Ví dụ VB6
Window Text

**MÃ: CHỌN HẾT**

```
Private Declare Function GetWindowText Lib "user32" Alias "GetWindowTextA" (ByVal hwnd As Long,
ByVal lpString As String, ByVal cch As Long) As Long
Private Declare Function SetWindowText Lib "user32" Alias "SetWindowTextA" (ByVal hwnd As Long,
ByVal lpString As String) As Long
Private Sub Form_Activate()
    Dim MyStr As String
    'Create a buffer
    MyStr = String(100, Chr$(0))
    'Get the windowtext
```

```
    GetWindowText Me.hwnd, MyStr, 100
    'strip the rest of buffer
    MyStr = Left$(MyStr, InStr(MyStr, Chr$(0)) - 1)
    'Triple the window's text
```

29. Đã có trên
**Hàm GetWindowTextLength**

Hàm GetWindowTextLength sẽ nhận về chiều dài, bằng ký tự, của window's title bar text đã được chỉ định (nếu window đó có title bar). Nếu window là một control, hàm này sẽ nhận chiều dài của đoạn text bên trong control.

Các hàm liên quan: GetWindowText
Yêu cầu: Windows NT 3.1; Windows 95 trở lên.

Dec:

> Declare Function GetWindowTextLength Lib "user32" Alias "GetWindowTextLengthA" (ByVal hwnd As Long) As Long

Tham số:
• hWnd Xác định window/control.

Giá trị trả về:
Nếu thành công, giá trị được trả về là chiều dài, tính bằng ký tự, của đoạn text. Trong những điều kiện chắc chắn, giá trị này thậm chí còn lớn hơn cả chiều dài của đoạn text. Nếu window không có text, giá trị trả về là zero. Xem thêm lỗi bằng hàm GetLastError.

Ví dụ VB6

**MÃ: CHỌN HẾT**

```
Private Declare Function GetWindowText Lib "user32" Alias "GetWindowTextA" (ByVal hwnd As Long, ByVal lpString As String, ByVal cch As Long) As Long
Private Declare Function GetWindowTextLength Lib "user32" Alias "GetWindowTextLengthA" (ByVal hwnd As Long) As Long
Private Sub Form_Activate()
Dim MyStr As String
'Create a buffer
MyStr = String(GetWindowTextLength(Me.hwnd) + 1, Chr$(0))
'Get the window's text
GetWindowText Me.hwnd, MyStr, Len(MyStr)
MsgBox MyStr
End Sub
```

**o0o--truongphu--o0o**

.....
Ghé thăm:
**Chuyện Linh Tinh**

truongphu
VIP

## Re: Sử dụng API trong Visual Basic

T.Năm 10/06/2010 6:08 pm

Nhiều bạn lúng túng khi tìm hWnd và class name
exe sau đây, code của NoBi, tôi design lại cho gọn

hoạt động: Run exe
rê chuột đến cửa sổ cần tìm
_ Ngừng: nhấn phím Ctrl
_ resume: click vào form của exe

TẬP TIN ĐÍNH KÈM

HWnd và ClassName.rar
(4.64 KiB) Đã tải **1079** lần

**o0o--truongphu--o0o**

.....
Ghé thăm:
**Chuyện Linh Tinh**

**ninjatcb**
Thành viên chính thức

## Re: Sử dụng API trong Visual Basic

CN 03/02/2019 9:46 am

Em chạy thử công cụ này trên Zalo thì không thấy bắt được ô để gõ Password. Trên zalo có nhiều cửa sổ (như là nút bấm, textbox) nhưng nó chỉ hiện được
Classname = Chrome_widgetwin_1
Xin các anh chỉ giúp thêm

Xem bài viết cách đây: Tất cả bài viết
Sắp xếp theo Ngày gửi
Tăng dần
Chuyển

Trả lời

19 bài viết • Trang **1** trên **1**

Quay về "[VB] Bài viết hướng dẫn"

Chuyển đến

**ĐANG TRỰC TUYẾN**

Đang xem chuyên mục này: **Không có thành viên nào trực tuyến.** và **1** khách



Sử dụng phần mềm phpBB® Forum Software © phpBB Limited