Microsoft Microsoft Microsoft

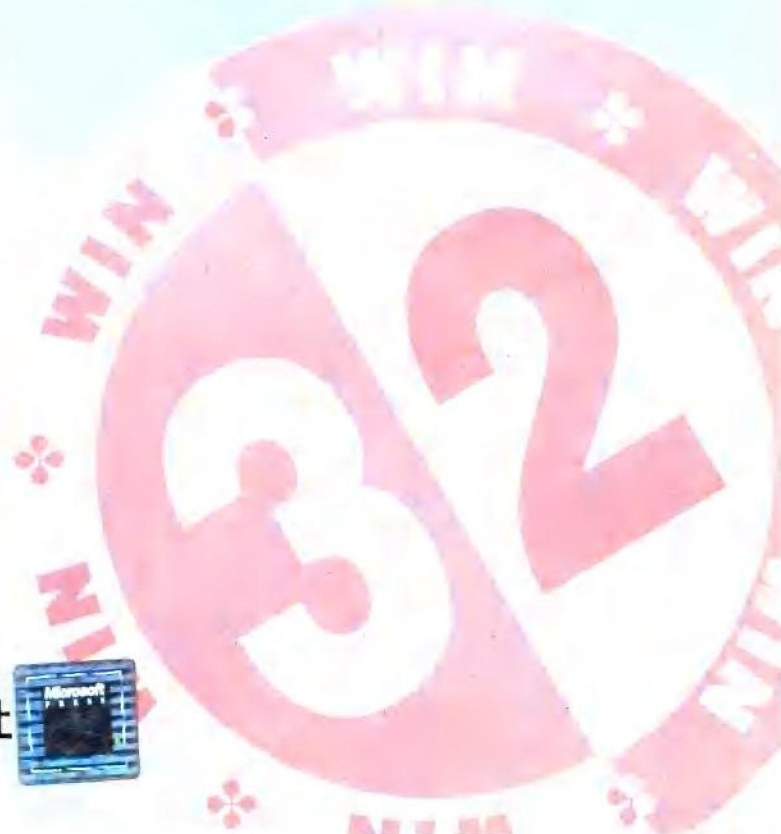
Microsoft® Win 32™ 程序员参考大全（一）

——窗口管理和图形设备接口

[美] Microsoft Corporation 著

欣力 李莉 陈维 李志峰 译

张燕 审校



清华大学出版社

Microsoft® Win32™程序员参考大全(一)

——窗口管理和图形设备接口

[美] Microsoft Corporation 著
欣力 李莉 陈维 李志峰 译
张 燕 审校

清华大学出版社

(京)新登字 158 号

Microsoft Win32 程序员参考大全(一)
——窗口管理和图形设备接口
Microsoft Win32 Programmer's Reference (Volume 1)
—— Windows Management and Graphics Device Interface
Microsoft Corporation

图书在版编目(CIP)数据

Microsoft Win32 程序员参考大全(一):窗口管理和图形设备接口/欣力等译 .—北京:清华大学出版社,1994

书名原文:Microsoft Win32 Programmer's Reference, Volume 1, Windows Management and Graphics Device Interface

ISBN 7-302-01670-4

Ⅰ.M… Ⅱ.欣… Ⅲ.①窗口软件-基本知识 ② 图形显示系统-接口 Ⅳ.① TP316 ②TP334

中国版本图书馆 CIP 数据核字(94)第 13313 号

出版者:清华大学出版社(北京清华大学校内,邮编 100084)
印刷者:清华大学印刷厂
发行者:新华书店总店北京科技发行所
开 本:787×1092 1/16 印张:39 字数:919 千字
版 次:1995 年 4 月第 1 版 1995 年 4 月第 1 次印刷
书 号:ISBN 7-302-01670-4/TP·717
印 数:0001—4000
定 价:67.50 元

引　言

Microsoft Win32 应用程序编程接口(API)使应用程序可以充分利用 Microsoft Windows 操作系统家族的 32 位能力,使用 Win32 API 写成的应用程序可以运行在单处理器的系统上,也可以运行在多处理器的系统上,而且可以移植到 RISC 结构的系统上。这套手册全面介绍了 Win32 API,包括窗口管理、图形、文件 I/O、线程、内存管理、安全性和网络。

本手册的组织

下面综述一下本手册的主要部分:

第 1 部分"窗口管理"　描述 Win32 API 中应用程序用来创建和管理窗口的部分。这一部分的章节详细介绍窗口、消息、消息队列、控制框、对话框和其它窗口管理主题的内容。

第 2 部分"图形设备接口"　描述 Win32 API 中应用程序用于设备无关图形的部分。这一部分的章节详细介绍设备描述表、转换、元文件、图元、位图和其它有关图形主题的内容。

本手册的目的在于阐述 Win32 API 的目的,解释 API 背后的操作系统概念,还说明了 Win32 函数如何相互配合来完成某项任务,并不说明如何编写、编译和连接包含这些函数的程序。

有关 Microsoft Win32 程序员参考手册

Microsoft Win32 程序员参考手册一套共五卷,全面描述了 Win32 API,包括函数及相关的数据类型、宏、结构和消息;程序员参考手册是有关 Windows 编程信息的主要参考源。

第 1 卷和第 2 卷描述 Win32 API 函数的用途,并解释这些函数背后的概念与原理。这两册是为不熟悉 Windows 或第一次接触某些部分的程序员设计的,它们提供了理解 Windows 编程所需的基本信息。

第 3 卷和第 4 卷是 Win32 按字母顺序的函数列表,它们定义了每一函数的语法、参数和返回值;第 5 卷按字母顺序列出了 Win32 的数据类型、宏、消息和结构。第 3 卷到第 5 卷是为那些熟悉 Windows 编程和那些只需了解特定函数的程序员编写的。

Microsoft Windows 与 C 语言

C 语言是基于 Windows 应用程序的首选开发语言,Windows 的许多编程特性都是针对 C 语言而设计的。尽管基于 Windows 的应用程序也可以用其它语言开发,但是 C 语言访问 Windows 函数最直接也最简单。正是由于这个原因,所有语法描述和程序范例都是

用 C 编程语言写的。

Wind 32 API 使用了标准 C 语言中没有的类型、宏和结构，这些类型、宏和结构使得基于 Windows 应用程序的建立工作更为简单，也使应用程序源代码更加清晰、易懂。本手册中讨论的类型、宏和结构都在 Win32 的 C 语言头文件中定义。

第 1 卷和第 2 卷的许多章节中含有代码范例，这些例子说明了如何使用 Win32 函数来完成任务；几乎所有的范例都是些代码片段，而不是完整的程序，它们的目的是说明函数可以使用的上下文关系，一般情况下，都假设例子中使用的变量、结构和常量都已定义、初始化，或者定义并初始化。有时例子中使用普通语言描述某一任务，而不是给出相应的语言。

尽管这些范例并不完整，但是使用下列步骤，你仍然可以在应用程序中使用它们：

- 在程序中包含文件 WINDOWS.H。
- 定义范例中使用的常量和函数以及结构中的常量。
- 定义并初始化所有变量。
- 使用相应的语句替换表示任务的描述。
- 检查返回值的出错情况并执行相应操作。

该手册中的有些范例既使用了 Win32，也使用 C 运行时的函数。

本书约定

下列约定用于全手册的语法定义。

约 定	含　义
斜体字符	指出某一位置或变量：实际值由你来提供。例如，语句 SetCursorPos(*X*, *Y*)，要求用值代替参数 *X* 和 *Y*。
[]	其中是可选参数。
|	用于分隔“或”选项。
…	表示前一项可以重复。
⋮	表示范例应用程序中的省略部分。

第一部分

窗口管理

第1章 窗 口

1.1 关于窗口

Microsoft Windows 应用程序中的窗口是屏幕上的一个正方形区域，是应用程序用来显示输出或接收用户的输入的。一个窗口与别的窗口(包括其它应用程序的窗口)共享屏幕，同一时间内只有一个窗口可接收用户的输入，用户可以通过鼠标、键盘或其它输入设备与该窗口以及拥有它的应用程序进行交互。

使用窗口的目的也就是因为 Windows 应用程序需要与用户进行交互来完成某些任务，所以 Windows 应用程序首要的任务就是创建一个窗口。这一章讲述 Windows 的应用程序编程接口(API) 的基本组成，应用程序用它来创建和使用窗口；管理窗口之间的关系；改变窗口的大小以及移动和显示窗口。

1.1.1 桌面窗口

启动 Windows 系统，它就会自动创建桌面窗口。桌面窗口是系统定义的窗口，这个窗口绘制了屏幕的背景，作为 Windows 应用程序显示窗口的基础。

桌面窗口使用了存放在位图文件(扩展名是.BMP)中的一个位图来绘制屏幕的背景，由这个位图所创建的图案被称为桌面壁纸。一般来说，桌面窗口所用的位图是在下面这个注册桌面壁纸的关键字所指定的文件中的：

HKEY_CURRENT_USER \ Control Panel \ Desktop \ Wallpaper

由系统配置的应用程序，如 Windows Control Panel，通过函数 SetDeskWallpaper 指定另外一个位图文件名来改变桌面壁纸，SetDeskWallpaper 从指定文件中安装位图，用这个位图绘制屏幕的背景，并在注册关键字 Wallpaper 中填写这个新的文件名。关于位图，参见第 29 章“位图”。有关注册，参见第 52 章“注册和初始化文件”。

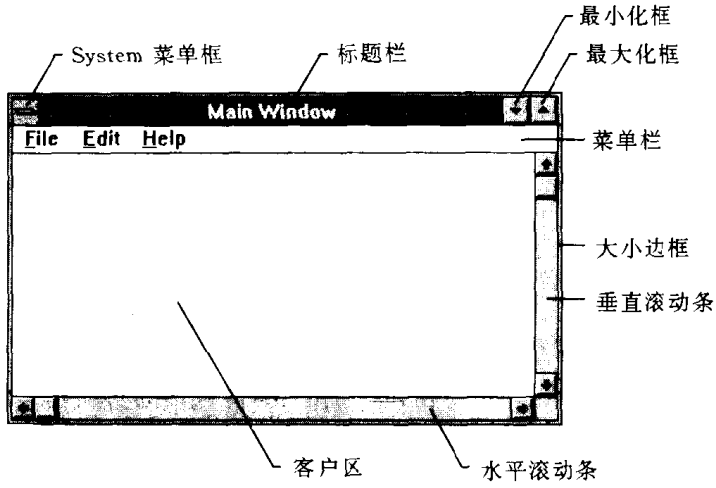
1.1.2 应用程序窗口

每一个 Windows 应用程序至少要创建一个窗口，称之为主窗口，作为应用程序的主窗口，这个窗口是用户与应用程序之间的主要接口。绝大部分应用程序还会直接或间接地创建许多其它的窗口，来完成与主窗口有关的任务，每一个窗口都是用来显示输出或是从用户得到输入。

应用程序窗口的组成

应用程序窗口一般是由标题栏、菜单栏、System 菜单(也叫 Control 菜单)、最小化框、最大化框、改变大小的边框、客户区、水平滚动条和垂直滚动条等组成。应用程序主窗口

通常含有上面列出的所有成员，下图是一个典型主窗口的组成情况：



标题栏用于显示应用程序定义的一行正文，通常是应用程序的名字或说明该窗口的用途，由应用程序在创建窗口时指定。标题栏使得用户可通过鼠标或其它的定点设备来移动窗口。

绝大部分的应用程序都有一个菜单栏，列出了应用程序所支持的命令，菜单栏中的项是命令的主要分类。从菜单栏中选中某一项通常都会显示一个弹出菜单，其中的项是对应于指定分类中的某个任务，用户可选择一个命令让应用程序完成该项任务。

System 菜单框是一个位图，单击它会显示 System 菜单，System 菜单是一个由 Windows 系统创建和管理的菜单，其中含有标准的菜单项设置，用户可通过它改变窗口的大小或对窗口重新定位，关闭应用程序或打开 Windows Task List。有关菜单及 System 菜单，参见第 16 章“菜单”。

最大化和最小化框也是位图，单击它会改变窗口的大小和位置。如果用户单击最大化框，Windows 系统就把窗口放大到屏幕的尺寸并定位窗口，这样窗口就会覆盖整个屏幕，与此同时 Windows 系统就会把最大化框换成恢复框。恢复框同样也是一个位图，单击它则把窗口恢复到原先的尺寸和位置。

如果用户单击最小化框，Windows 系统就把窗口减小到图符的尺寸，窗口定位在屏幕的底部，在此位置上显示窗口的图符。位图是一个 32×32 象素的位图，它表示一个窗口。有关图符，参见第 23 章“图符”。

改变大小边框是围绕窗口四周的一个区域，通过它用户可用鼠标或其它的定点设备改变窗口的大小。

客户区是窗口的一部分，应用程序用于显示输出，如正文或图形。例如，桌面印刷应用程序在客户区中显示文档的当前页面，应用程序必须提供一个称之为窗口过程的函数，来处理窗口的输入并在客户区中显示输出。有关窗口过程，参见第 4 章“窗口过程”。

水平和垂直滚动条把鼠标或键盘的输入转换成一个数值，应用程序用来按水平或垂直方向移动客户区的内容。例如，显示一个较长文件的字处理应用程序，通常就得提供一

个垂直滚动条，以便用户向上或向下翻页。

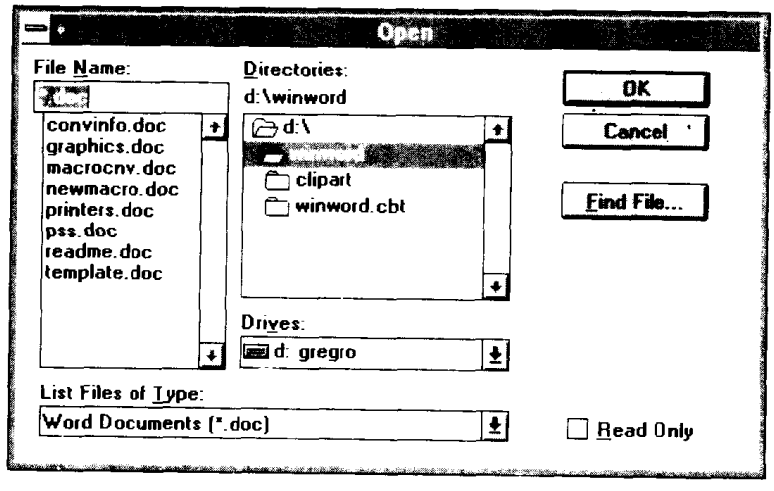
标题栏、菜单栏、System 菜单、最小化和最大化框、改变大小边框以及滚动条统称为窗口的非客户区，它们差不多都是由 Windows 系统来管理的；应用程序则管理窗口的其它事情，特别是管理客户区的外观及操作。

控制框、对话框以及消息框

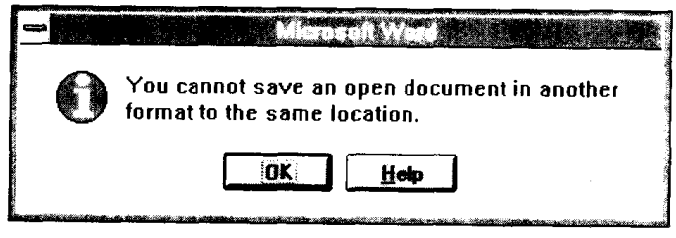
应用程序还使用另外几种类型的窗口，包括控制框、对话框和消息框。

控制框是应用程序用来获得用户特定信息的窗口，比如：要打开文件的名字或是正文选择中有关点的尺寸的设置。应用程序也会通过控制框获取所需要的信息，以便控制应用程序的某种特性。例如，字处理应用程序通常提供的一个控制，让用户设置自动换行特性的开或关。有关控制框，参见第 9 章“控制框”及第 15 章“静态控制框”。

控制框总是要与其它窗口连用的，典型的是对话框。对话框是一个含有一个或多个控制框的窗口。应用程序可通过对话框提示用户提供完成某一个命令所需的输入。例如，含有打开文件命令的应用程序，就得显示一个对话框，其中的控制框让用户指定路径或文件名。下图就是一个 Open 对话框及相关的控制文件。



消息框是用于给用户一些提示或警告的窗口。例如，消息框能够在应用程序完成某项任务过程中出现问题时通知用户。下图所示的对话框解释了应用程序 Microsoft Word for Windows 为什么不能完成一个保存文件的操作。



对话框和消息框一般没有主窗口中那些窗口成员，通常都有一个标题栏、System 菜

单、一个边框(不是改变大小边框)及客户区。一般没有菜单栏、最小化和最大化按钮或滚动条。有关对话框和消息框,参见第 18 章“对话框”。

1.1.3 创建窗口

应用程序可以通过函数 CreateWindow 或 CreateWindowEx 来创建它的主窗口,并提供 Windows 系统定义窗口属性所需的信息。函数 CreateWindowEx 有一个参数 dwExStyle,函数 CreateWindow 则没有;换句话说这两个函数是等同的,CreateWindow 只是简单地调用了 CreateWindowEx,把参数 dwExStyle 置成 0,基于这个原因,本章所涉及到的只是函数 CreateWindowEx。

Windows 系统还提供了另外一些函数——包括 DialogBox、CreateDialog 及 MessageBox ——用于创建特殊用途的窗口,比如对话框和消息框。有关这些函数,参见第 18 章“对话框”。

1.1.3.1 窗口属性

应用程序在创建窗口时必须提供下列信息:

- 窗口类
- 窗口名
- 窗口风格
- 父窗口或属主窗口
- 尺寸
- 位置
- 定位
- 子窗口标识或菜单句柄
- 实例句柄
- 创建数据

下面对这些属性进行一一说明。

窗口类

每一个窗口都从属于某一窗口类,应用程序必须在创建某类窗口之前注册窗口类,窗口类定义了窗口的外观和特性。窗口类的主要部分是一个窗口过程,也就是接收和处理给窗口的输入和请求的函数,Windows 系统以消息的形式给窗口提供输入或请求。有关窗口类,窗口过程或消息,参见第 3 章“窗口类”、第 4 章“窗口过程”以及第 2 章“消息与消息队列”。

窗口名

窗口可以有一个名字,窗口名(也叫窗口正文)是便于用户识别一个窗口的正文字符串。主窗口、对话框或消息框一般是在其标题栏上显示窗口名。而对于控制,窗口名的外观取决于控制的类。按钮、编辑控制或静态控制是在控制所占据的矩形框内显示其窗口名;列表框、组合框或静态控制则不显示其窗口名。

应用程序在创建窗口以后可用函数 SetWindowText 来改变窗口名,通过函数

GetWindowTextLength 和 GetWindowText 来检取当前窗口名的正文。

窗口风格

每个窗口都有一个或几个窗口风格，窗口风格是一个命名的常量，由它定义窗口类没有指定的窗口外观及特性。例如，由 SCROLLBAR 类创建一个滚动条控制，但 SBS_HORZ 及 SBS_VERT 风格决定所创建的滚动条是水平的还是垂直的。只有少数窗口风格适用于所有窗口，绝大部分只是提供给特定窗口类的窗口。

Windows，从某种程度上说是由类的窗口过程来解释风格的。

父窗口或属主窗口

窗口可以有一个父窗口，有父窗口的窗口称之为子窗口，由父窗口提供的坐标系统对子窗口进行定位。父窗口会影响窗口的外观，例如，如果一个子窗口被裁剪，那么子窗口就不会超出其父窗口的边框。如果一个窗口没有父窗口或是父窗口就是桌面窗口，那么就叫做顶层窗口。应用程序通过函数 EnumWindows 来获取它的每一个顶层窗口的句柄，再由 EnumWindows 把每一个顶层窗口的句柄传给应用程序定义的回调函数（由 Windows 系统调用的函数）。

一个窗口可以拥有别的窗口，或者被别的窗口所拥有，被拥有的窗口总是在其属主窗口的前面，它的属主窗口被最小化时隐藏起来，并随属主窗口的销毁而销毁。

位置、尺寸和在 Z 轴中的次序

每一个窗口都有它的位置、尺寸及在 Z 轴中的次序。一般窗口的位置由相对屏幕左上角的坐标决定；如果是子窗口则是由相对于它的父窗口客户区左上角的坐标决定。窗口的尺寸是其高度和宽度的象素值；窗口在 Z 轴上的次序则是窗口在覆盖窗口堆中的位置。

子窗口标识或菜单句柄

子窗口可以有一个子窗口标识，它是由应用程序定义的对应于该子窗口的一个唯一的值。子窗口标识对创建多个子窗口的应用程序来说是特别有用的。创建一个子窗口时，应用程序就为它设置一个子窗口标识，创建了窗口之后，可通过函数 SetWindowLong 来改变窗口标识，或是用函数 GetWindowLong 检取标识。

除了子窗口，每个窗口都能有一个菜单，应用程序是在注册窗口类或创建窗口时，通过提供一个菜单句柄来产生菜单。

实例句柄

每个 Windows 应用程序都有一个与之相应的实例句柄，Windows 系统在应用程序开始的时候就为它提供了实例句柄。因为同一个应用程序可以有多个拷贝，Windows 系统就是利用实例句柄来区分应用程序的不同实例。应用程序必须为不同的窗口指定实例句柄，包括那些创建窗口的实例。

创建数据

每个窗口都可有与之相应的由应用程序定义的创建数据。在窗口第一次被创建时，Windows 系统把数据的指针传给所创建窗口的窗口过程，窗口过程用这些数据初始化应用程序定义的变量。

1.1.3.2 创建主窗口

每个 Windows 应用程序都得用 WinMain 作为入口，函数 WinMain 完成一系列工作，包括注册主窗口的窗口类并创建主窗口。WinMain 调用函数 RegisterClass，注册主窗口类，函数 CreateWindowEx 创建主窗口。

可移植性：在 Windows NT 系统中，入口点可以是另外的一个名字，而不是 WinMain，而在 Windows 3.x 系统中则必须是 WinMain。

Windows 系统创建了主窗口之后不会自动显示它，应用程序必须调用函数 ShowWindow 来显示主窗口。通常应用程序的 WinMain 函数在创建了主窗口之后就调用 ShowWindow，WinMain 把两个参数传给 ShowWindow：主窗口的句柄以及一个决定主窗口在第一次被显示时是最小化还是最大化的标志，一般来说，这个标志可以是 Windows 系统头文件中定义的以 SW_ 作为前缀的任何常量，但是如果是调用 ShowWindow 来显示应用程序的主窗口，那么这个标志必须设置成 SW_SHOWDEFAULT，这个标志通知 Windows 系统，要根据启动应用程序的程序来显示窗口。

1.1.3.3 创建窗口消息

创建任何窗口，Windows 系统都要向窗口的窗口过程发送消息。Windows 系统创建了窗口的非客户区后要发送 WM_NCCREATE 消息，创建了客户区后则需发送 WM_CREATE 消息。窗口过程在 Windows 系统显示窗口之前接收这两种消息，这两种消息都含有结构 CREATESTRUCT 的指针，该结构中含有函数 CreateWindowEx 中指定的所有信息。通常窗口过程根据接收到的消息完成初始化工作。

如果创建一个子窗口，Windows 系统在发送了 WM_NCCREATE 和 WM_CREATE 消息之后，还要给父窗口发送 WM_PARENTNOTIFY 消息。在创建窗口的时候还可能会发送其它的消息，消息的数目和次序要由窗口类及风格以及用于创建窗口的函数来确定。在后面的有关章节中讲解了这些消息。

1.1.3.4 多线程应用程序

Windows NT 应用程序可以运行多个线程，每一个线程都可以创建窗口。应用程序可通过函数 EnumThreadWindows 统计由某一个线程所创建的窗口，这个函数再把每个线程窗口的句柄传给应用程序定义的回调函数。函数 GetWindowThreadProcessId 返回创建某个窗口的线程的标识。

1.1.4 窗口句柄

创建了窗口之后，创建函数返回唯一标识窗口的窗口句柄，应用程序在其它函数中用这个句柄以确保是对该窗口的操作。窗口句柄属于 HWND 数据类型；应用程序必须在说明一个窗口句柄的变量时使用这种类型。

Windows 系统还有另外几个专用的常量，在某些函数中可用来代替窗口句柄，这些常量以 HWND_ 作为前缀。例如，可在函数 SetWindowPos 中用 HWND_TOP 和 HWND_BOTTOM 常量把窗口移到 Z 次序的顶部或底部。

尽管常量 NULL 不是一个窗口句柄，但应用程序可在一些函数中用它来表明没有窗口受到影响。例如，把函数 CreateWindowEx 的 hwndParent 参数置成 NULL，那么所创建的窗口就没有父窗口或属主窗口。有些函数可能返回 NULL 而不是一个句柄，表示给定的操作没有施加给任何窗口。

应用程序可以用函数 FindWindow 来确定系统中是否有指定类名或窗口名的窗口，如果有这样的窗口，FindWindow 就返回这个窗口的句柄。函数 IsWindow 确定一个窗口句柄是否标识一个有效的、存在的窗口。

1.1.5 窗口风格

Windows 系统提供了通用的窗口风格和特定类窗口风格，通用窗口风格是以 WS_ 为前缀的常量；它们可以组合使用以形成不同的窗口类型，如主窗口、对话框及子窗口。特定类窗口风格决定了属于预定义的控制类窗口的外观和特性，如编辑控制和列表框。这一节讲述通用窗口风格，有关特定类风格，参见第 10 章“按钮”及第 15 章“静态控制”。

1.1.5.1 覆盖窗口

覆盖窗口是一个顶层窗口，它有一个标题栏，边框和客户区，用作应用程序的主窗口。它也可以有一个 System 菜单，最小化和最大化框及滚动条，作为主窗口使用的覆盖窗口一般具备所有这些成员。

应用程序通过在函数 CreateWindowEx 中指定 WS_OVERLAPPED 或 WS_OVERLAPPEDWINDOW 风格来创建一个覆盖窗口，用 WS_OVERLAPPED 风格创建的覆盖窗口带有标题栏和边框；用 WS_OVERLAPPEDWINDOW 风格创建的覆盖窗口则具有标题栏、改变大小边框、System 菜单、最小化和最大化框。

1.1.5.2 弹出窗口

弹出窗口是一个特定类型的覆盖窗口，通常用于对话框、消息框及其它显示在应用程序主窗口外面的临时窗口中。对弹出窗口来说，标题栏是任选的，除此之外，一个弹出窗口与以 WS_OVERLAPPED 风格创建的覆盖窗口是一样的。

应用程序通过在函数 CreateWindowEx 中设定 WS_POPUP 风格来创建一个弹出窗口，如果需要标题栏，那么还得设定 WS_CAPTION 风格。应用程序可使用 WS_POPUPWINDOW 风格创建一个具有边框和 System 菜单的弹出窗口。WS_CAPTION 风格必须与 WS_POPUPWINDOW 风格组合使用以确保 System 菜单是可见的。

1.1.5.3 子窗口

子窗口具有 WS_CHILD 风格，被限制在其父窗口的客户区中，应用程序通常就是用子窗口来把客户区分成几个功能区域。应用程序通过在函数 CreateWindowEx 中设定

WS_CHILD 风格来创建一个子窗口。

子窗口必须有它的父窗口，父窗口可以是覆盖窗口、弹出窗口或者甚至是另外一个子窗口，应用程序在调用函数 CreateWindowEx 时指定父窗口。如果应用程序在函数 CreateWindowEx 中指定 WS_CHILD 风格但没有指定父窗口，Windows 系统就不会创建这个窗口。

子窗口有一个客户区，但是除非有明确的要求否则不会有任何其它特性。应用程序可以为子窗口请求一个标题栏、System 菜单、最小化和最大化框、边框和滚动条。不管是在注册子窗口类还是在创建子窗口时，应用程序设置的菜单句柄都会被忽略掉。

定位

Windows 系统总是相对父窗口客户区左上角定位子窗口，子窗口的任何部位都不会出现在父窗口边框之外。如果应用程序创建的子窗口比父窗口大，或是在定位时，子窗口的某些部分或全部超出了父窗口的边框，Windows 系统就会裁剪子窗口，也就是说超出父窗口客户区以外的部分不会被显示出来。下表列出的操作，影响父窗口的同时也会影响子窗口：

父窗口	子 窗 口
销毁	在父窗口被销毁之前被销毁。
隐藏	在父窗口被隐藏之前被隐藏，子窗口只有在父窗口可见时可见。
移动	跟随父窗口客户区一起移动，移动过后，子窗口将会对绘制它的客户区做出响应。
显示	在显示了父窗口之后显示。

裁剪

Windows 系统不会自动地裁剪父窗口客户区中的子窗口，这就意味着父窗口可以在子窗口的位置上进行绘画。但如果父窗口具有 WS_CLIPCHILDREN 风格，Windows 系统就会裁剪父窗口客户区中的子窗口，这样父窗口就不能在上面进行绘画。

子窗口覆盖同一客户区中的其它子窗口，一个或多个其它的子窗口共享同一个父窗口的子窗口叫兄弟窗口，兄弟窗口也可在相互间的客户区中绘画，除非其中某个子窗口具有 WS_CLIPSIBLINGS 风格，如果应用程序为子窗口指定了这个风格，那么画在这个窗口中的子兄弟窗口的任何部分就被裁剪。

如果窗口具有 WS_CLIPCHILDREN 或 WS_CLIPSIBLINGS 风格，系统的性能会有所降低。每个窗口都要消耗系统资源，所以应用程序不能不加区分地使用子窗口。应用程序应该在主窗口的窗口过程中对其主窗口进行逻辑划分，而不是使用子窗口。

与父窗口的关系

应用程序通过调用函数 SetParent 改变子窗口的父窗口，这种情况下，Windows 系统从旧父窗口的客户区中删除这个子窗口，并把它移到新父窗口的客户区。如果 SetParent 指定一个 NULL 句柄，桌面窗口就成为其新的父窗口，这样子窗口就画在桌面窗口上，在任何其它窗口边框的外面。函数 GetParent 用来检取子窗口的父窗口句柄。

父窗口把它的客户区中的一个部分让与子窗口，子窗口就从这个区域接收所有的消

息。父窗口的每一个子窗口的窗口类是没有必要相同的，这就是说，应用程序可用具有不同外观以及能够完成不同任务的子窗口来填充一个父窗口，例如，对话框可以含有许多不同类型的控制框，每一个控制框就是一个子窗口，从用户得到不同类型的数据。

子窗口只能有一个父窗口，但一个父窗口可以有许多子窗口，子窗口还可以再有子窗口。在这个窗口链中，每一个子窗口都被叫做原始父窗口的子孙窗口，应用程序可用函数 IsChild 来确定一个子窗口是否是某一父窗口的子窗口或是子孙窗口。

函数 EnumChildWindows 用来统计父窗口的子窗口，EnumChildWindows 把每一个子窗口的句柄传给应用程序定义的回调函数，给定父窗口的所有子孙窗口都被统计。

消息

Windows 系统把子窗口的输入消息直接传给子窗口，这些消息并不通过父窗口来传送，唯一的例外就是假如子窗口已通过函数 EnableWindow 被禁止，在这种情况下，Windows 系统就把给子窗口的任何输入消息传给其父窗口，这就使得父窗口检查输入消息或是根据需要允许子窗口。

子窗口可以有一个唯一的整数标识，子窗口标识在控制窗口时是很重要的，应用程序通过向它发送消息来指导控制的活动，应用程序使用控制的子窗口标识表示向该控制发送消息。另外，控制还得向其父窗口发送通知消息，通知消息中就含有这个控制的子窗口标识，父窗口用它来识别消息的来处。应用程序为其它类型的子窗口指定子窗口标识是通过把函数 CreateWindowEx 的 hmenu 参数设置为某一个值而不是菜单句柄。

1.1.5.4 窗口边框

Windows 提供下列类型的边框。

风　格	描　述
WS_BORDER	创建一个有单线边框的窗口。
WS_DLGFRAME	创建一个有双边框的窗口，这种风格的窗口大多用于对话框，而且这种风格的窗口是不能有标题栏的。
WS_EX_DLGMODALFRAME	创建一个有双边框的窗口，但不同于 WS_DLGFRAME 风格的是，应用程序还可通过设置 WS_CAPTION 为窗口创建一个标题栏。
WS_THICKFRAME	创建一个有重置大小边框的窗口。

具有 WS_OVERLAPPED 和 WS_POPUPWINDOW 风格的窗口一般具有 WS_BORDER 风格。其余的边框风格必须与 WS_OVERLAPPED 和 WS_POPUPWINDOW 风格组合使用，以形成具有不同边框风格的覆盖窗口。

如果带有 WS_POPUP 或 WS_CHILD 风格的窗口没有指定边框风格，系统就创建一个不带边框的窗口，应用程序可用无边框窗口来划分父窗口的客户区，这样的划分用户是看不见的。

1.1.5.5 非客户区组成

窗口的非客户区可以有一个标题栏、System 菜单、最小化和最大化框、重置大小边框、水平和垂直滚动条。应用程序创建一个窗口时，其非客户区的组成可通过在函数 CreateWindowEx 中指定下列风格来设定。

风　格	描　述
WS_CAPTION	创建一个有标题栏的窗口(含有 WS_BORDER 风格)。
WS_HSCROLL	创建一个有水平滚动条的窗口。
WS_MAXIMIZEBOX	创建一个有最大化框的窗口。
WS_MINIMIZEBOX	创建一个有最小化框的窗口。
WS_SYSMENU	创建一个在标题栏中有 System 菜单框的窗口，同时还得设置 WS_CAPTION 风格。
WS_VSCROLL	创建一个有垂直滚动条的窗口。

1.1.5.6 初始状态

下面所列出的风格决定一个窗口的初始状态：被允许还是被禁止，可见还是不可见，最小化还是最大化。

风　格	描　述
WS_DISABLED	所创建的窗口一开始是被禁止的，被禁止的窗口是不能从用户接收输入的。
WS_MAXIMIZE	所创建的窗口一开始是处于最大化的。
WS_MINIMIZE	所创建的窗口一开始是处于最小化的。
WS_VISIBLE	所创建的窗口一开始是可见的。

1.1.5.7 父窗口和子窗口的风格

下面所列出的风格影响父窗口与它的子窗口，以及子窗口与兄弟窗口之间的裁剪关系：

风　格	描　述
WS_CLIPCHILDREN	在父窗口中进行绘画时，子窗口所占空间是排除在外的，在创建父窗口时使用这种风格。
WS_CLIPSIBLINGS	裁剪相关的子窗口；也就是说，在某一个子窗口接收到 WM_PAINT 消息时，WS_CLIPSIBLINGS 风格裁剪要更新的子窗口中被其它子窗口覆盖的区域。如果没有指定 WS_CLIPSIBLINGS，子窗口又相互覆盖着，那么需要在一个子窗口的客户区进行绘画时，有可能画到另一个相邻子窗口的客户区中。这种风格只能用于 WS_CHILD 风格。