新编 Windows API 参考大全

本书编写组 编

张争平 审校

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
URL: http://www.phei.com.cn

新编 Windows API 参考大全

本书编写组　编

张争平　审校

電子工業出版社

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

作为 Microsoft 32 位平台的应用程序编程接口，Win32 API 是从事 Windows 应用程序开发所必备的。本书首先对 Win32 API 函数做完整的概述；然后收录五大类函数：窗口管理、图形设备接口、系统服务、国际特性以及网络服务；在附录部分，讲解如何在 Visual Basic 和 Delphi 中对其调用。

本书是从事 Windows 应用程序开发的软件工程师的必备参考手册。

图书在版编目(CIP)数据

新编 Windows API 参考大全/本书编写组编．-北京：电子工业出版社，2000.3
ISBN 7-5053-5777-8

Ⅰ.新… Ⅱ.〈… Ⅲ.窗口软件，Windows-程序设计 Ⅳ.TP316.7

中国版本图书馆 CIP 数据核字(2000)第 01568 号

书　　名：**新编 Windows API 参考大全**
编　　者：本书编写组
审 校 者：张争平
责任编辑：郭　立
特约编辑：史宝军
印 刷 者：北京天竺颖华印刷厂
装 订 者：三河市金马印装有限公司
出版发行：电子工业出版社　URL：http://www.phei.com.cn
　　北京市海淀区万寿路 173 信箱　邮编 100036
经　　销：各地新华书店
开　　本：787×1092　1/16　印张：62 字数：2392 千字
版　　次：2000 年 3 月第 1 版　2000 年 4 月第 2 次印刷
书　　号：ISBN 7-5053-5777-8/TP·2997
印　　数：4000 册　　定价：98.00 元

前　言

Win32 API 作为 Microsoft 32 位平台(包括：Windows 9x,Windows NT3.1/4.0/5.0,Windows CE)的应用程序编程接口，它是构筑所有 32 位 Windows 平台的基石，所有在 Windows 平台上运行的应用程序都可以调用这些函数。

从事 Windows 应用程序开发，离不开对 Win32 API 函数的调用。只有充分理解和利用 API 函数，才能深入到 Windows 的内部，充分挖掘系统提供的强大功能和灵活性。

近年来，随着 Microsoft 32 位平台的版本升级，Win32 API 函数的构成、功能与调用方式都有很大的发展变化，然而，国内很少有相关的新版资料出版。为了满足广大开发人员的迫切需求，我们经过认真收集、整理素材，组织编写了这本与各种 Microsoft 32 位平台最新版本同步的 Win32 API 参考手册。

全书收录了五大类函数：窗口管理、图形设备接口、系统服务、国际特性以及网络服务。所有函数均附有功能说明、参数说明、返回值说明、备注以及引用说明。另外，在本书的第一章，我们对 Win32 API 函数作了完整的概述；在附录部分，讲解了如何在 Visual Basic 和 Delphi 中对其调用。

由于篇幅较大，涉及技术内容广泛，加之时间仓促，书中难免存在不少错误或疏漏，希望广大读者给与批评指正。

编　者

本书编写组

主　编： 朱友芹

编　委：

李汇成	周家汉	夏建华	游少华	陈　情	陈　平	陈正德
李　明	张　泉	胡　勇	贾志勇	冯三红	冯　景	赵志高
胡光明	曹玉林	殷姬海	王东松	李晓枫	李铁柱	甘　霞
陈　勇	余玉兰	贾高军	赵　霞	欧阳福来	张保林	孙立志
张亚男	冯　涛	冯海涛	李家刚	夏　燕	王菊峰	何玉梅
周小喧	周　丽	金　勇	李玉凤	李明勇	曾　琼	潘晓兰
余志杰	叶来春	余伯平	许碧青	廖加银	沈立兵	赵继涛
余建国	周　浩	叶词双	何　杰	项志立	甘训明	田　华
孙仁发	张争平	宁　进	王三五	朱志华	黄龙泉	钱　兵
张志刚	马　军	付丽琼	孙　刚	叶新华	冯加词	甘训红

审　校： 张争平

目　录

第一章　Win32 API 概论

1.1　为什么使用 Win32 API

在 Windows 程序设计领域处于发展初期时，Windows 程序员可使用的编程工具唯有 API 函数。这些函数在程序员手中犹如“积木块”一样，可搭建出各种界面丰富、功能灵活的应用程序。不过，由于这些函数结构复杂，所以往往难以理解，而且容易误用。

随着软件技术的不断发展，在 Windows 平台上出现了很多优秀的可视化编程环境，程序员可以采用“所见即所得”的编程方式来开发具有精美用户界面和功能的应用程序。这些可视化编程环境操作简便、界面友好，比如：Visual C++，Delphi，Visual Basic 等等。在这些工具中提供了大量的类库和各种控件，它们替代了 API 的神秘功能。事实上，这些类库和控件都是构筑在 Windows API 的基础上的，但它们使用方便，加速了 Windows 应用程序的开发，所以受到程序员的普遍采用。有了这些类库和控件，程序员们便可以把主要精力放在整体功能的设计上，而不必过于关注具体细节。不过，这也导致了非常多的程序员在类库面前“固步自封”，对下层 API 函数的强大功能一无所知。

实际上，程序员要想开发出更灵活、更实用、更具效率的应用程序，必然要涉及到直接使用 API 函数。虽然类库和控件使应用程序的开发容易得多，但它们只提供 Microsoft Windows 的一般功能，对于一些比较复杂和特殊的功能来说，单使用类库和控件是难以实现的，必须直接使用 API 函数来编写。API 函数是构筑整个 Windows 框架的基石，只有充分理解和利用 API 函数，才能深入到 Windows 的内部，充分发挥各种 32 位平台的强大功能和灵活性，才能成功地扩展和突破类库、控件和可视开发环境的限制。

1.2　Win32 API 简介

Win32 API 即为 Microsoft 32 位平台的应用程序编程接口(Application Programming Interface)。所有在 Win32 平台上运行的应用程序都可以调用这些函数。

使用 Win32 API，应用程序可以充分挖掘 Windows 的 32 位操作系统的潜力。Mircrosoft 的所有 32 位平台都支持统一的 API，包括函数、结构、消息、宏及接口。使用 Win32 API 不但可以开发出在各种平台上都能成功运行的应用程序，而且也可以充分利用每个平台特有的功能和属性。

在具体编程时，程序实现方式的差异依赖于相应平台的底层功能的不同。最显著的差异是某些函数只能在更强大的平台上实现其功能。例如，安全函数只能在 Windows NT 操作系统下使用。另外一些主要差别就是系统限制，比如值的范围约束，或函数可管理的项目个数等等。

标准 Win32 API 函数可以分为以下几类：

窗口管理
窗口通用控制
Shell 特性
图形设备接口
系统服务
国际特性
网络服务

在下面各节中，我们分别介绍这 7 种类型的 API 函数。

1.2.1　窗口管理函数

窗口管理函数向应用程序提供了一些创建和管理用户界面的方法。你可以使用窗口管理函数创建和使用窗口来显示输出、提示用户进行输入以及完成其他一些与用户进行交互所需的工作。大多数应用程序都至少要创建一个窗口。

应用程序通过创建窗口类及相应的窗口过程来定义它们所用窗口的外观和行为。窗口类可标识窗口的缺省属性，比如窗口是否接受双击鼠标按钮的操作，或是否带有菜单。窗口过程中包含的代码用于定义窗口的行为，完成所需的任务，以及处理用户的输入。

应用程序可使用 GDI 函数来产生窗口的输出。由于所有的窗口都共享显示屏幕，所以应用程序不接受对整个屏幕的访问。系统管理所有的输出内容，并对它们进行排列和剪裁，使其能够适合相应的窗口。应用程序可以在处理输入消息时，或为了响应系统的需求而在窗口中绘图。当窗口的大小或位置发生变化时，系统通常会向应用程序发送一个消息，要求它对该窗口中原来未显露的区域进行重画。

应用程序以消息的形式接受鼠标和键盘输入。系统将鼠标移动、鼠标按钮操作转换为输入消息，并将这些消息放入该应用程序的消息队列中。系统为每个应用程序都自动提供一个消息队列。应用程序使用消息函数从消息队列中获取消息，并将它们分派给适当的窗口过程进行处理。

应用程序可以直接处理鼠标和键盘输入，也可以让系统使用菜单和键盘加速键将这些低级输入转换成命令消息。你可以使用菜单向用户展现一个命令列表。系统对所有菜单操作所需的动作进行管理，包括让用户选择一个命令，然后再向窗口过程发送一个标识该选择的消息。键盘加速键是应用程序定义的按键操作组合，系统可将其转换为消息。加速键通常对应于菜单中的某个命令，并与该命令产生相同的消息。

应用程序通过在对话框中向用户提示附加信息来响应命令消息。对话框实际是一个临时的窗口，用于显示信息或提示输入。一个对话框通常由一些表示按钮和方框的控制组成，可供用户进行选择或输入信息。对话框中可包括用于输入正文、滚动正文、从列表中选择列表项等操作的控制。对话框管理和处理来自这些控制的输入，使应用程序可使用这些信息，来完成所要求的命令操作。

通过使用“资源”可以共享很多有用的数据，比如位图、图标、字体和字符串等，只需将这些数据作为“资源”添加到应用程序或 DLL 文件中。应用程序通过使用资源函数，找到资源并将它们加载到内存来获取这些数据。

窗口管理函数还提供了其他一些与窗口有关的特性，比如插入标记(Caret)、剪贴板、光标、挂钩(Hook)、图标以及菜单等函数。

窗口管理函数包括以下几类：

易用特性函数（Accessibility Features）

Win32 API 提供的一系列易用特性使得有残疾的人也能很容易的使用计算机，Win32 API 提供了一些函数和结构来控制这些特性。

按钮函数（Button）

Microsoft 提供了对话框和控制来支持应用程序与用户之间的交互通讯。按钮就是一种控制，用户可通过点击按钮来向应用程序提供输入信息。

插入标记函数（Caret）

一个插入标记是位于窗口绘图区中的一个闪动的直线、方块或图标。插入标记通常用于指示文本或图形将插入的位置。Win32 应用程序可以使用插入标记函数来创建一个插入标记，改变它的闪动频率，显示、隐藏插入标记，或重新设置插入标记的位置。

组合框函数（Combo Box）

组合框是由 COMBOBOX 类定义的一种控制，综合了列表框和编辑控制的很多功能。使用组合框函数可以在组合框中显示或获取不同类型的数据。

通用对话框函数（Common Dialog Box）

通用对话框是在通用对话框库中定义的，其功能是用来完成一些通用的任务，比如打开文件、打印文档等。通用对话框为用户提供了一个统一的用户界面，使用户在不同的应用程序中完成通用任务时的操作都相同，不必每次都学习不同的操作过程。

光标函数（Cursor）

光标是显示屏幕上的一个小图形，其所在的位置由指点设备比如鼠标、光笔或轨迹球等控制。当用户移动鼠标时，系统就会随之移动光标的位置。应用程序使用 Win32 光标函数可以创建、加载、显示、移动、限制和删除光标。

对话框函数（Dialog Box）

对话框是应用程序创建的一个临时窗口，用于获取用户的输入。应用程序通常使用对话框向用户显示一些命令提示信息。一个对话框一般由一个或多个控制(子窗口)组成，这些控制可用来输入文本、选择选项或执行命令动作。

编辑控制函数（Edit Control）

编辑控制是一个矩形窗口，通常用在对话框中，用户可通过键盘向编辑控制中输入和编辑文本。系统对 Unicode 文本(字符采用双字节编码)和 ANSI(字符采用单字节编码)文本都支持。

图标函数（Icon）

图标是一个图片，由一个位图图像组成，并和一个掩码组合构成该图片的透明区域。当提到图标时，可以是下列两种情况：

1) 单个图标图像。资源类型为 RT_ICON。

2) 一组图标图像，系统或应用程序可从中选择。资源类型为 RT_GROUP_ICON。

应用程序使用图标函数可以创建、显示、删除和复制图标。

键盘加速键函数（Keyboard Accelerator）

键盘加速键(或简称为加速键)是一个按键操作或多个按键操作的组合，可向应用程序发送 WM_COMMAND 或 WM_SYSCOMMAND 消息。

使用键盘加速键函数可以拷贝、创建、加载或删除加速键表，还可以将加速键消息转换为命令消息。

键盘输入函数（Keyboard Input）

键盘输入函数提供了接受和处理键盘输入的方法。

列表框函数（List Box）

Microsoft 的 Win32 API 提供了对话框和控制来支持应用程序与用户之间的交互通讯。列表框是一个控制窗口，其中包含一系列选项，可供用户进行选择。使用列表框函数可以在列表框中显示或获取不同类型的数据。

菜单函数（Menu）

菜单函数向 Win32 应用程序提供了一系列创建、管理和使用菜单的方法，包括对菜单条、菜单项、子菜单等的处理。

消息和消息队列函数（Message and Message Queue）

消息和消息队列函数向 Win32 应用程序提供了一系列使用消息和消息队列的方法，包括对消息进行传播、发送、获取、转换等操作。

鼠标输入函数（Mouse Input）

鼠标输入函数提供了接受和处理鼠标输入的方法。

多文档接口函数（Multiple Document Interface）

多文档接口(MDI)是应用程序定义用户界面的一种规范，在这种界面下，用户可以同时使用多个文档。

资源函数（Resource）

一个资源是一些二进制数据，可以添加到 Win32 应用程序的可执行文件中。资源既可以是标准的，也可以是自己定义的。标准资源中的数据包括图标、光标、菜单、对话框、位图、增强元文件、字体、加速键表、消息表入口、字符串表入口或版本。应用程序定义的资源（也称为定制的资源）可以包含特殊应用程序所需的任何数据。

使用资源函数可以添加、删除、拷贝、替换或加载各种资源数据。

滚动条函数（Scroll Bar）

在 Win32 应用程序的窗口中，可以显示比该窗口的显示区更大的数据对象，比如文档或位图。当窗口提供了滚动条时，用户就可以通过拖动滚动条来浏览该数据对象中位于显示区外面的部分。

滚动条包括水平滚动条和垂直滚动条。使用滚动条函数可以创建和管理这两种滚动条。

窗口函数（Window）

在图形化的 Win32 应用程序中，窗口是屏幕上的一个矩形区域，应用程序可在该区域中显示输出结果，并接受用户输入。因此，一个图形化的 Win32 应用程序的首要任务之一就是创建一个窗口。

一个窗口与其他窗口共享显示屏幕，也包括其他应用程序所创建的窗口。一次只能有一个窗口接受用户的输入。用户可以使用鼠标、键盘或其他输入设备与该窗口及拥有该窗口的应用程序进行交互。使用窗口函数可以创建和管理窗口。

窗口类函数（Window Class）

一个窗口类是一个属性的集合，系统将该属性集合用作创建窗口的模板。每个窗口都是某个窗口类的一个成员。使用窗口类函数可以获取有关窗口类的各种信息。

窗口过程函数（Window Procedure）

每个窗口类都有一个相关的窗口过程，该过程实际是一个函数，用于处理发送给该类的所有窗口的所有消息。一个窗口的所有外观显示和行为都依赖于窗口过程对这些消息的响应。使用窗口过程函数可以调用相应的窗口过程。

窗口属性函数（Window Property）

窗口属性是分配给一个窗口的任何数据。一个窗口属性通常是指向某特定窗口数据的句柄，但它也可以是任何 32 位的值。每个窗口属性都由一个字符串名字标识。使用窗口属性函数可以添加、获取、设

置或删除窗口属性。

1.2.2 窗口通用控制

系统 Shell 提供了一些控制，使用这些控制可以使窗口具有与众不同的外观。由于这些控制是由 DLL 支持的，是操作系统的一部分，所以它们对所有的应用程序都可用。使用通用控制有助于使应用程序的用户界面与系统 Shell 及其他应用程序保持一致。由于开发一个控制需要花费一定的时间，所以直接使用通用控制也可以节省大量的开发时间。

通用控制是由通用控制库 COMCTL32.DLL 支持的一个控制窗口集。与其他控制一样，一个通用控制也是应用程序的一个子窗口，它与其他窗口联合使用，完成 I/O 操作。通用控制 DLL 包括一个编程接口，应用程序可使用其中的函数创建和管理控制，以及从控制中接受用户输入。

1.2.3 Shell 特性

Win32 API 中包含一些接口和函数，应用程序可使用它们来增强系统 Shell 的各方面功能。

一个名字空间是一个符号集合，比如文件和目录名字，或数据库关键字。Shell 使用一个单层结构的名字空间来组织用户关心的所有对象，包括文件、存储设备、打印机及网络资源。名字空间类似于文件系统的目录结构，只不过名字空间中包含的是对象，而不是文件和目录。

快捷键（也称为一个 Shell 连接）是一个数据对象，它包含的信息可用于访问位于 Shell 名字空间的任何位置的其他对象。使用快捷键时，应用程序不必知道对象的当前名字和位置就可以访问该对象。可以通过快捷键访问的对象包括文件、文件夹、磁盘驱动器、打印机及网络资源。

有几种方法可以扩展 Shell。系统使用图标来表示 Shell 名字空间中的文件。缺省情况下，系统对具有相同文件扩展名的所有文件都显示相同的图标。可以用一个图标句柄来改变某特殊文件的缺省图标。使用上下文相关菜单句柄可以修改一个上下文相关菜单的内容，这也是一种 Shell 扩展。当用户用鼠标右键点击或拖动一个对象时，系统会显示一个上下文相关菜单。该上下文相关菜单中所包含的命令只应用在被点击或拖动的对象上。大多数上下文相关菜单都包含一个 Properties 命令，用于显示所选中项目的属性表。一个属性表由一系列重叠的窗口组成(每个窗口称为一页)，用于显示有关某个对象的信息。属性表句柄是一种 Shell 扩展，使用它可以向系统定义的属性表中添加页，或替换控制面板的属性表的某些页。一个拷贝挂钩(Hook)句柄是一种 Shell 扩展，可以允许或拒绝对一个文件对象的移动、拷贝、删除或重命名。

系统 Shell 包含一个快速查看(Quick View)命令，使用户可以直接查看一个文件的内容,而不必运行创建该文件的应用程序。文件浏览器提供了一个用于查看文件的用户界面。Shell 使用文件扩展名来确定应运行哪个浏览器。你可以为新的文件格式提供文件浏览器，或用具有更强功能的浏览器来替换原来的浏览器。文件浏览器与文件分析器联合使用，后者的功能是对文件名进行分析，以便确定应生成哪种类型文件的 Quick View。你还可以提供其他的文件分析器来支持新的文件类型。

1.2.4 图形设备接口

图形设备接口(GDI)提供了一系列的函数和相关的结构，应用程序可以使用它们在显示器、打印机或其他设备上生成图形化的输出结果。使用 GDI 函数可以绘制直线、曲线、闭合图形、路径、文本以及位图图像。所绘制的图形的颜色和风格依赖于所创建的绘图对象，即画笔、笔刷和字体。你可以使用画笔来绘制直线和曲线，使用笔刷来填充闭合图形的内部，使用字体来书写文本。

应用程序通过创建设备环境(DC)，可以直接向指定的设备进行输出。设备环境是一个 GDI 管理的结构，其中包含一些有关设备的信息，比如它的操作方式及当前的选择。应用程序可使用设备环境函数来创建 DC。GDI 将返回一个设备环境句柄，在随后的调用中，该句柄用于表示该设备。例如，应用程序可以使用该句柄来获取有关该设备性能的一些信息，诸如它的类型(显示器、打印机或其他设备)，它的显示界面的尺寸和分辨率等。

应用程序可以直接向一个物理设备进行输出，比如显示器或打印机；也可以向一个“逻辑”设备进行输出，比如内存设备或元文件。逻辑设备向应用程序所提供的保存输出结果的格式，可以很容易地将其发送到物理设备上。一旦应用程序将输出结果记录到了一个元文件中，那么该元文件就可以被使用任意多次，并且该输出结果可以被发送到任意多个物理设备上。

应用程序可以使用属性函数来设置设备的操作方式和当前的选择。操作方式包括文本和背景颜色，混色方式(也称为二元光栅操作，用于确定画笔或笔刷的颜色与绘图区域现有的颜色如何进行混色)，映射

方式(用于指定 GDI 如何将应用程序所用的坐标映射到设备坐标系统上)。当前的选择是指绘图时使用哪个绘图对象。

图形设备接口函数包括以下几类：

位图函数（Bitmap）

位图是一个图形对象，可将图像作为文件进行创建、处理（比例缩放、滚动、旋转和绘制）和存储。位图函数提供了一系列处理位图的方法。

笔刷函数（Brush）

笔刷是一种绘图工具，Win32 应用程序可使用它绘制多边形、椭圆形和路径的内部。绘图应用程序使用笔刷绘制图形；字处理应用程序使用笔刷绘制水线；计算机辅助设计（CAD）应用程序使用笔刷绘制截面视图的内部；电子表格应用程序使用笔刷绘制饼图的扇形和直方图的方条。笔刷函数提供了一系列创建和使用笔刷的方法。

剪裁函数（Clipping）

剪裁是一种处理过程，它将输出到某个区域或路径中的内容限制在应用程序窗口的显示区内。剪裁函数提供了一系列处理剪裁区域的方法。

颜色函数（Color）

颜色是组成 Win32 应用程序所生成的图片和图像的一个重要元素。Win32 API 提供了一系列管理和使用画笔、笔刷、文本和位图的颜色的函数。

坐标空间及映射函数（Coordinate Space and Transformation）

Win32 应用程序使用坐标空间和映射函数对输出的图形进行比例缩放、旋转、转换、剪裁和反射。坐标空间是基于笛卡尔坐标系统的一个平面空间。该坐标系统要求有两个垂直相交的、长度相等的坐标轴。共有 4 种坐标空间：现实坐标、页面坐标、设备坐标、物理设备坐标（显示区，或桌面，或打印纸的页面）。映射方式就是改变（“映射”）对象的大小、方向和形状的一种算法。

设备环境函数（Device Context）

设备环境是一个结构，它定义了一系列图形对象及其相关的属性，以及会影响输出结果的绘图方式。这些图形对象包括：画笔（用于画直线），笔刷（用于绘图和填充），位图（用于屏幕的拷贝或滚动），调色板（用于定义可用的颜色集），剪裁区（用于剪裁和其他操作），路径（用于绘图和画图操作）。设备环境函数用于对设备环境进行创建、删除或获取信息。

填充图形函数（Filled Shape）

填充图形是一些几何图形，其轮廓由当前的画笔绘制，内部由当前的笔刷填充。共有 5 种填充图形：椭圆，弦图，饼图，多边形，矩形。填充图形函数用于对填充图形进行操作。

字体和文本函数（Font and Text）

字体用于在视频显示器或其他输出设备上绘制文本。Win32 API 提供了一系列用于安装、选择和查询各种字体的字体和文本函数。

ICM 2.0 函数

Microsoft Windows 98 和 Windows NT 5.0 所使用的颜色管理方案称为 Image Color Management 版本 2.0，或 ICM2.0，由一系列函数组成。

直线和曲线函数（Line and Curve）

直线和曲线用于在光栅设备上绘制输出图形。一条直线是光栅显示器上的一系列高亮像素点（或打印纸上的一系列点），由两个点进行标识：起点和终点。一条规则曲线也是光栅显示器上的一系列高亮像素点（或打印纸上的一系列点），符合某个二次曲线段的周界（或一部分）。不规则曲线则是由不符合二次曲线段的一系列像素点组成。

元文件函数（Metafile）

元文件是一个结构的集合，这些结构是以与设备无关的格式存储图像。设备无关是元文件与位图的差异之一。与位图不同，元文件保证是与设备无关的。不过，元文件有一个缺点：它通常比位图的绘图速度慢。因此，如果一个应用程序要求有较快的绘图速度，而不需要具有设备无关性，则应该用位图代替元文件。

元文件函数提供了一些对元文件进行操作的方法。

多显示器支持函数（Multiple Display Monitors）

每个 Windows 工作站所支持的显示器个数是不受限制的。可以用创建邻接区域的方式安排多个显示器。每个显示器的大小和颜色深浅都可以独立设置。

所有的显示器屏幕一起构成了一个虚拟屏幕。桌面窗口覆盖整个虚拟屏幕，而不仅仅是某个显示屏

幕。由于现有的应用程序都要求显示器具有一个原点坐标(0，0)，所以虚拟屏幕必须在某个显示器上包含原点坐标(0，0)，这个显示器就被看作是主显示器。

每个物理显示设备都由一个 HMONITOR 类型的显示器句柄表示。一个显示器在它的整个生存期间具有相同的 HMONITOR 值。

任何显示设备环境(DC)的 Win32 函数所返回的值都是主显示器的 DC。要想获取其他显示器的 DC，可使用 EnumDisplayMonitors 函数。系统对每个显示器调用回调函数，为该显示器传入一个 DC 值。用户可以使用该 DC 在该显示器上绘图。

绘图和画图函数（Painting and Drawing）

绘图和画图函数为应用程序提供了一系列在窗口中绘图的方法，以及如何创建和使用显示设备环境(DC)的方法。

路径函数（Path）

一个路径是指一个或多个被填充、被绘制轮廓或既被填充又被绘制轮廓的图形(或形状)。Win32 应用程序将路径用作很多用途，在绘图和画图应用程序中使用路径。计算机辅助设计(CAD)应用程序用路径来创建唯一剪裁区，绘制不规则形状的轮廓，以及填充不规则形状的内部。路径函数用于创建、改变和绘制路径。

画笔函数（Pen）

画笔是 Win32 应用程序用于绘制直线和曲线的图形工具。画图应用程序使用画笔来画手画线、直线以及曲线。计算机辅助设计(CAD)应用程序用画笔来画可见线、隐藏线、截面线、中心线等等。字处理和桌面出版应用程序用画笔来画边界和水线。电子表格应用程序用画笔来指明图表的趋向，以及勾勒直方图和饼图的轮廓。画笔函数提供了一系列使用画笔的方法。

打印和打印假脱机函数（Printing and Print Spooler）

Microsoft Windows 和 Windows NT 提供了一套完整的函数，使应用程序可以在不同的设备上进行打印，如激光打印机，向量绘图仪，光栅打印机，以及传真机等。

矩形函数（Rectangle）

Win32 应用程序使用矩形来指定显示屏幕上或窗口中的一个矩形区域。矩形函数用于对矩形进行操作。

区域函数（Region）

区域是指一个可被填充、着色、转换和加外框的形状，包括矩形、多边形或椭圆(或这几种形状的组合)，用于完成击键测试(测试光标位置)。

区域函数用于对区域进行操作。

1.2.5 系统服务

系统服务函数为应用程序提供了访问计算机资源以及底层操作系统特性的手段，比如访问内存、文件系统、设备、进程和线程。应用程序使用系统服务函数来管理和监视它所需要的资源。例如，应用程序可使用内存管理函数来分配和释放内存，使用进程管理和同步函数来启动和调整多个应用程序或在一个应用程序中运行的多个线程的操作。

系统服务函数提供了访问文件、目录以及输入输出(I/O)设备的手段。应用程序使用文件 I/O 函数可以访问保存在指定计算机以及网络计算机上的磁盘和其他存储设备上的文件和目录。这些函数支持各种文件系统，从 FAT 文件系统，CD-ROM 文件系统(CDFS)，到 NTFS。

系统访问函数为应用程序提供了一些可以与其他应用程序共享代码或信息的方法。例如，可以将一些有用的过程放到 DLL 中，使它们对所有的应用程序都可用。应用程序只需使用 DLL 函数将动态链接库加载进来并获取各过程的地址，就可以使用这些过程了。通讯函数用于向通讯端口写入数据及从通讯端口读出数据，并控制这些端口的操作方式。有几种内部通讯(IPC)的方法，比如 DDE、管道(Pipe)、邮槽(Mailslot)和文件映射。对于提供安全属性的操作系统来说，应用程序可使用安全函数来访问安全数据，并保护这些数据不会被有意或无意地访问或破坏。

使用系统服务函数可以访问有关系统和其他应用程序的信息。应用程序可用系统信息函数来确定计算机的特别属性，比如是否出现鼠标、显示屏幕上的元素具有多大尺寸。注册和初始化函数用于将应用程序的特殊信息保存到系统文件中，以便于该应用程序的新实例对象，甚至其他应用程序都可以获取和使用这些信息。

应用程序使用系统服务函数可以处理执行过程中的的一些特殊情况，比如错误处理、事件日志、异

常处理。还有一些属性可用于调试和提高性能。例如，使用调试函数可对其他进程的执行过程进行单步控制，而性能监视函数则可对某个进程的执行路径进行跟踪。

系统服务函数还提供了一些特性，可用于创建其他类型的应用程序，比如控制台应用程序和服务。

系统服务函数包括以下几类：

访问控制函数（Access Control）

Microsoft Windows NT 所提供的安全功能对 Win32 应用程序是自动使用的。在系统中运行的每个应用程序都受由 Windows NT 的特殊配置所提供的安全功能所影响。Windows NT 是支持 Win32 安全功能的唯一平台。

Windows NT 的安全功能对大多数 Win32 函数的影响都是最小的，不需要安全功能的 Win32 应用程序不必合并任何特殊代码。不过，你可使用 Windows NT 的安全属性向 Win32 应用程序提供一些服务。

访问控制函数提供了一系列控制访问 Win32 对象(比如文件)、管理函数(比如设置系统时间或审核运行动作的函数)的 Windows NT 安全模型。

原子函数（Atom）

原子表格是一个系统定义的表格，用于保存字符串和相应的标识符。应用程序将一个字符串放到原子表格中，并接受一个 16 位的整数(称为一个原子)，用于访问该字符串。放到原子表格中的字符串被称为原子名字。

原子函数提供了一系列对原子进行添加、删除、初始化等的操作。

客户/服务器访问控制函数（Client/Server Access Control）

客户/服务器访问控制函数包括三类：

用于模拟客户机。

用于检查和设置私有对象上的安全描述符。

用于生成安全时间日志中的审核消息。

剪贴板函数（Clipboard）

剪贴板是由一系列函数和消息组成，Win32 应用程序可使用它来传输数据。由于所有的应用程序都可以访问剪贴板，所以数据可以很容易地在应用程序之间或一个应用程序内部进行传输。

通讯函数（Communication）

通讯资源是一个物理或逻辑设备，用于提供双向的异步数据流。例如，串行端口、并行端口、传真机以及调制解调器都是通讯资源。对于每个通讯资源都有一个服务供应程序(包含一个库或驱动程序)，使应用程序可以访问该资源。通讯函数是通讯设备所使用的函数。

控制台函数（Console）

Microsoft Windows 和 Windows NT 提供了控制台函数，用于管理字符模式的应用程序(这种应用程序未提供自己的图形用户界面)的输入和输出(I/O)。

数据解压库函数（Data Decompression Library）

数据解压库函数在 LZEXPAND.DLL 中声明，用于对压缩的文件进行解压。

调试函数（Debugging）

调试器是一个应用程序，开发人员可使用它来检查和改正编程错误。Win32 API 的调试函数为用户提供了一系列的调试手段。

设备输入和输出函数（Device Input and Output）

Win32 应用程序使用设备输入和输出控制与设备驱动程序进行通讯。被访问的设备由设备句柄标识，而设备驱动程序要完成的动作则由控制代码来指定。

动态数据交换函数（Dynamic Data Exchange）

Win32 API 为不能使用“动态数据交换管理库(DDEML)”的应用程序提供了一系列实现动态数据交换的函数。

动态数据交换管理函数（Dynamic Data Exchange Management）

动态数据交换(DDE)是一种内部通讯方式，即使用共享内存在应用程序之间交换数据。应用程序可以使用 DDE 进行一次性的数据传输，以及数据的即时交换和更新。

动态数据交换管理函数为用户提供了一系列管理动态数据交换的手段。

动态链接库函数（Dynamic-Link Library）

动态连接库(DLL)是由函数和数据组成的一些模块。一个 DLL 是由它的调用模块(.EXE 或.DLL)在运行时加载的。当一个 DLL 被加载后，它就被映射到其调用进程的地址空间中。

DLL 可以定义两种函数：外部的和内部的。外部函数可以被其他模块调用，内部函数只能在声明它

的 DLL 内部被调用。尽管 DLL 可以输出数据，但它的数据通常只能由它的函数使用。

DLL 提供了一种使应用程序模块化的方法，这样就可以更容易地更新和重用程序的功能。DLL 也有助于在几个应用程序同时使用相同的功能时减少内存开销，因为虽然每个应用程序都拥有一份数据的备份，但它们可以共享代码。

错误函数（Error）

写得好的应用程序应包括一些能够处理意外错误并可从错误中顺利恢复的代码。当发生错误时，应用程序可能需要用户进行干预，或自己恢复。在一些极端情况下，应用程序可能会将用户从系统中退出或关机。错误函数为用户提供了一些进行错误处理的方法。

事件日志函数（Event Logging）

很多应用程序都在不同的属性错误日志中记录错误和事件。这些属性错误日志具有不同的格式，并显示不同的用户界面，而且无法将数据合并起来得到一个完整的报告。因此，用户必须要检查各种数据来诊断问题。Windows NT 的事件日志为应用程序(和操作系统)提供了一种标准、集中的方法，来记录重要的软件和硬件事件。事件日志服务将事件从不同的地方保存到一个称为“事件日志”的集合中。Windows NT 还提供了一个事件浏览器和编程接口，用于查看日志和检查日志。事件日志函数提供了一系列编写和检查事件日志的方法。

文件函数（File）

文件是计算机存储信息的基本单位，不同的信息可分别存放在不同的文件中。应用程序可使用文件函数对文件进行输入和输出(I/O)操作。

文件安装库函数（File Installation Library）

Win32 API 包含一个文件安装库，应用程序使用它可以更容易地安装文件，使安装程序能分析当前已安装的文件。

文件映射函数（File Mapping）

文件映射函数用于对文件映射对象进行操作。

文件系统函数（File System）

Win32 应用程序依赖文件系统来保存和获取存储设备上的信息。文件系统提供了应用程序在与存储设备相关的个别卷上创建和访问文件及目录时所需的底层支持。

每个文件系统都由一个或多个驱动程序和所支持的动态链接库(定义文件系统的数据格式和特性)组成。它们确定了文件名的约定、安全性及可恢复性的级别，以及输入输出(I/O)操作的一般性能。文件系统函数用于对文件系统进行操作。

句柄和对象函数（Handle and Object）

对象是一个表示系统资源的数据结构，比如表示一个文件、线程或图像。应用程序不能直接访问对象所表示的对象数据或系统资源，而是必须使用对象句柄。对象句柄可用于检查和修改系统资源。每个句柄在一个内部维护的表中都有一项。在这些项中包含资源的地址以及标识资源类型的方法。句柄和对象函数用于对句柄和对象进行操作。

Hook 函数

Hook 是系统消息处理机制中的一部分。在系统消息处理机制中，应用程序可安装一个子程序来监视系统中的消息传送情况，并可处理某些类型的消息(在这些消息到达目的窗口过程之前)。Hook 函数用于对 Hook 进行操作。

ImageHlp 函数

ImageHlp 函数由 IMAGEHLP DLL 提供。ImageHlp 函数可用于 PE 格式的图像。PE 图像由一个兼容的 Win32 连接程序提供，比如由 Microsoft Developer Studio 提供。

超大整数操作函数（Large Integer Operations）

Win32 API 提供了一系列超大整数操作函数对超大整数(64 位)进行操作。

底层访问控制函数（Low-Level Access Control）

底层访问控制函数用于对安全描述符和访问控制列表(ACL)进行操作。大多数底层访问控制函数在各种版本的 Windows NT 上都支持。底层访问控制函数是 Win32 API 所提供的三套访问控制函数之一。

LSAPI 函数

如果在 Windows 95 和 Windows NT 上运行的应用程序调用了 LSAPI 1.10 (License Service Application Programming Interface)，那么就可以使用软件计量和许可跟踪技术。LSAPI 包含 7 个函数，用于提供许可服务。

邮槽函数（Mailslot）

邮槽是一种单向的内部处理通讯(IPC)机制。Win32 应用程序可以在邮槽中保存消息，邮槽的所有者可以获取保存在其中的消息。这些消息通常是通过网络发送到一台指定的计算机上，或发送到某个指定域中的所有计算机上。域是一组工作站和服务器，共享一个组名。

可以选择使用命名管道来代替邮槽进行内部处理通讯。命名管道是两个进程交换消息的一种简单方法。而邮槽则是一个进程向多个进程广播消息的一种简单方法。需要考虑的重要一点是邮槽使用邮包，而命名管道则不用。邮槽函数可用于创建邮槽、设置或获取邮槽信息。

内存管理函数（Memory Management）

内存管理函数用于分配和使用内存。

管道函数（Pipe）

管道是一段共享内存，用于进程通讯。创建管道的进程称为管道服务程序。连接管道的进程称为管道客户程序。某个进程向管道中写入信息，然后其他进程从管道中读出信息。管道函数用于创建、管理和使用管道。

电源管理函数（Power Management）

电源管理函数用于对计算机的电源进行管理。

进程和线程函数（Process and Thread）

一个 Win32 应用程序由一个或多个进程组成。在最简单的条件下，一个进程就是一个可执行程序，在该进程的环境中运行一个或多个线程。线程是操作系统分配处理器时间的基本单位。一个线程可以执行进程代码的任何部分，包括正被其他线程执行的部分。一个“纤度”（Fiber）是一个执行单位，必须由应用程序手工调度。“纤度”在调度它的线程环境中运行。

作业对象允许进程组被作为一个单位进行管理。作业对象是可命名、可得到及可共享的对象，用于控制与其相关的进程的属性。在作业对象上完成的操作会影响所有与该作业对象相关的进程。

进程和线程函数包括三类函数：进程和线程函数、作业对象函数、“纤度”函数。

注册函数（Registry）

注册表是一个系统定义的数据库，应用程序和系统构件可使用它来保存和获取配置数据。注册函数用于对注册表进行操作。

字符串处理函数（String Manipulation）

字符串处理函数用于对字符串进行处理。

结构化的异常处理函数（Structured Exception Handling）

异常是在程序执行过程中发生的一种事件，发生异常时需要执行正常的控制流程以外的代码。共有两种异常：硬件异常和软件异常。硬件异常是由 CPU 引发的，可能由于执行了某些指令序列而产生，比如除零操作，或访问一个无效的内存地址。软件异常是由应用程序或操作系统显式地引发。例如，当系统检测出一个无效的参数值时就会引发一个异常。

结构化的异常处理是一种同时处理软件异常和硬件异常的机制。因此，在程序中可用作对硬件和软件异常一起进行处理。使用结构化的异常处理使用户可以完全控制对异常的处理，为调试器提供支持，并且对所有编程语言和机器都是可用的。

同步函数（Synchronization）

Win32 API 提供了各种方法来调整执行过程中的多个进程。同步函数为线程提供了一系列对资源访问进行同步的机制。

系统信息函数（System Information）

系统信息函数用于修改系统的配置、设置和属性。

系统消息函数（System Message）

系统消息函数用于向一些系统部件发送系统消息，比如应用程序、网络驱动器、系统级设备驱动器等。

系统关机函数（System Shutdown）

应用程序可使用系统关机函数将当前的用户退出系统、关机，或锁定工作站。

磁带备份函数（Tape Backup）

备份应用程序可使用磁带备份函数从磁带中读取数据，向磁带中写入数据，初始化磁带，以及获取磁带或磁带驱动信息。

时间函数（Time）

Microsoft Windows 和 Windows NT 提供了各种日期和时间函数，用于获取和设置系统及个别文件的日期和时间。