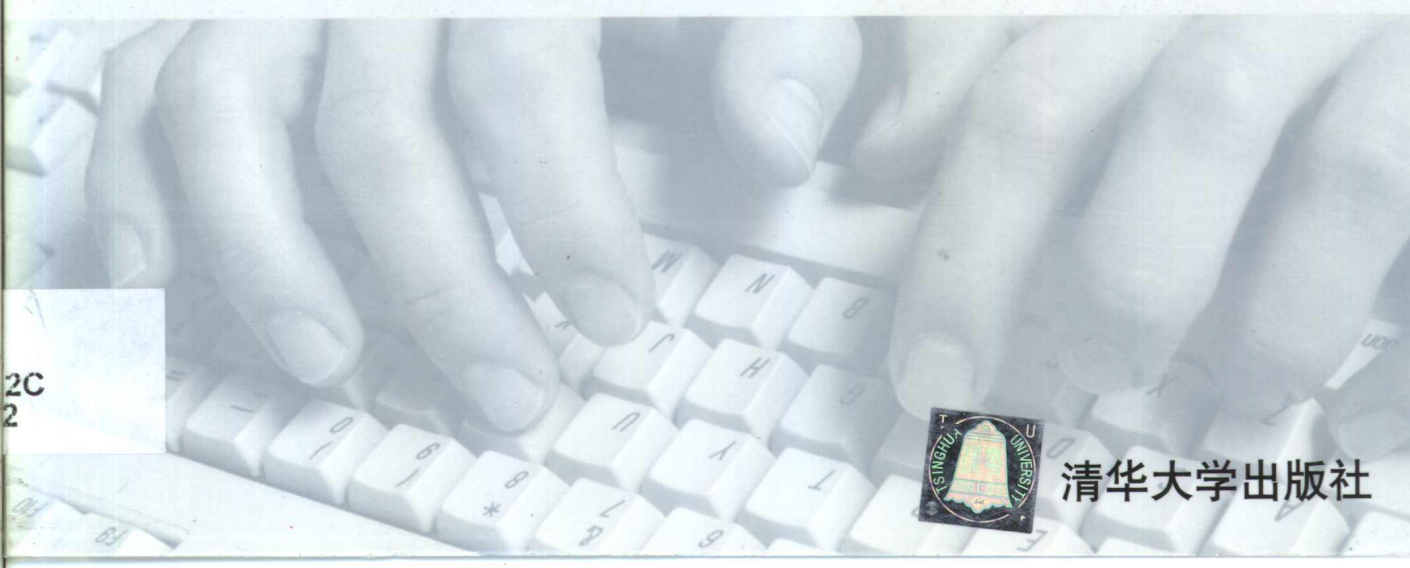
Visual Basic
高级程序设计

刘炳文 编著

Visual Basic 图形与多媒体程序设计



清华大学出版社

Visual Basic
高级程序设计

刘炳文　编著

Visual Basic 图形与多媒体程序设计

清华大学出版社

(京)新登字 158 号

内 容 简 介

本书介绍了 Visual Basic 本身提供的基本图形程序设计方法以及通过多媒体控件进行多媒体程序设计的技术，同时介绍了 Windows 应用程序接口(API)，并把它应用于 Visual Basic 的图形和多媒体程序设计中。书中提供了大量实例。

本书可供具有 Visual Basic 程序设计基础和一定实践经验，需要进一步应用 Visual Basic 进行较高层次软件开发的用户使用，也可供具有一定 Visual Basic 程序设计实践的读者自学或参考。

书　　名：Visual Basic 图形与多媒体程序设计
作　　者：刘炳文　编著
出 版 者：清华大学出版社(北京清华大学学研大厦，邮编 100084)
http://www.tup.tsinghua.edu.cn
责任编辑：焦　虹
印 刷 者：北京国马印刷厂
发 行 者：新华书店总店北京发行所
开　　本：787×1092 1/16　**印张**：19.5　**字数**：451 千字
版　　次：2002 年 8 月第 1 版　2002 年 8 月第 1 次印刷
书　　号：ISBN 7-302-05425-8/TP・3197
印　　数：0001～6000
定　　价：23.00 元

Visual Basic 图形与多媒体程序设计

第1章 Windows 应用程序接口(API)

Windows 应用程序接口(application programming interface,API)是 Windows 的重要功能之一。它实际上是一组用 C 语言编写的函数库,拥有数以千计的函数,这些函数随同 Windows 一起安装。在 Visual Basic 应用程序中,可以像调用普通过程一样调用 API 中的函数,实现所需要的操作。

API 函数所在的函数库称为动态链接库(dynamic link library,DLL)。笔者编著的《Visual Basic ActiveX 程序设计》一书介绍了用 Visual Basic 建立 DLL 的操作,可以通过建立引用来使用 DLL 中的方法。本章将要介绍的 DLL 是由其他语言(C/C++)建立的,由这些 DLL 提供的函数称为 API 函数。下面将介绍在 Visual Basic 中调用这些 API 函数的一般方法和注意事项,为学习后面几章的内容作好准备。

1.1 静态链接与动态链接库

API 函数放在动态链接库中,因此,在具体介绍 API 函数调用之前,先介绍静态链接与动态链接的区别,并介绍动态链接库的基本类型。

1.1.1 静态链接

在 DOS 环境中用编译型语言开发应用程序时,链接或连接是经常用到的概念。例如,在用 C 语言或 C++语言编写程序时,如果需要调用运行时间库中已有的函数,程序员无需在自己的源代码中重写库中的函数,而只需给出函数名和所需要的参数,即可执行由该函数提供的操作。在生成可执行文件时,首先对源程序进行编译,生成目标文件(.OBJ)。此时标准 C 或 C++库还没有被链接到应用程序中,在目标文件内只有函数的调用代码,没有函数本身的代码。当用链接程序(link)把目标文件(.OBJ)与库文件(.LIB)相链接,生成可执行文件(.EXE)时,链接程序才从库文件中取出源程序中所使用的库函数,加入到经链接生成的可执行文件中。因此,可执行文件实际上包含了源程序及其所调用的库函数的代码。

在 C、C++等编译型语言中,这种用链接程序链接 C 或 C++库函数的方式叫做静态

链接。有了静态链接，就可以把一些公用函数放入库文件中，程序员不必自己编写这些函数，即无需在源代码中包含这些函数，而只是给出相应的调用指令，在链接期间由链接程序将所调用的函数的代码拷贝到应用程序的可执行文件中，如图 1.1 所示。静态链接是单任务(DOS)环境下常用的链接方式。之所以称为“静态链接”，是因为在生成可执行文件时，程序访问库函数所需要的全部寻址信息都是固定的，而且在程序运行时保持不变。

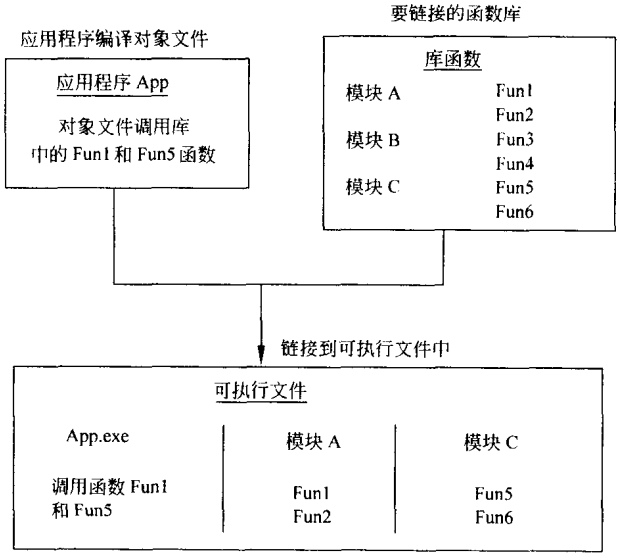


图 1.1　静态链接

在一般情况下，当把库函数链接到可执行文件时，链接程序将链入函数所在的整个模块，因此，在图 1.1 中，函数 Fun2 和 Fun6 虽然没有被程序 App 调用，但仍被包含到可执行文件中。对一些新版的编译和链接程序(如 C++)来说，则允许链入单个函数。

1.1.2　动态链接库

Windows 是多任务操作系统。在多任务环境中，应用程序共享内存资源。如果多个应用程序都调用库文件中相同的函数，则在链接时把该函数拷贝给每个应用程序，运行时在内存中生成同一函数的多个拷贝，造成内存资源的浪费。此外，如果修改了库中函数的代码，则必须对调用该函数的应用程序重新进行链接。因此，在 Windows 环境下，通常不使用静态链接方式，而使用动态链接库(dynamic link library，DLL)。

动态链接库是一个函数库，是由可被其他程序或 DLL 调用的函数集合组成的可执行文件模块。之所以被称为动态链接库，是因为 DLL 的代码并不是某个应用程序的组成部分，而是在运行时链接到应用程序中。与动态链接不同，静态链接方式是在链接期间把库(静态链接库)中的代码链接到可执行文件中，也就是说，在可执行文件中含有库函数的代码。

动态链接分为两个阶段，即链接过程和装入过程。当应用程序调用动态链接库中的

某个函数时，链接程序并不拷贝被调用函数的代码，而只是从引入库中拷贝一些指示信息，指出被调用函数属于哪个动态链接库(.DLL 文件)。因此，在应用程序的可执行文件中，存放的不是被调用的函数的代码，而是 DLL 中该函数的内存地址。程序运行后，当需要调用该函数时，进入装入过程，把应用程序与 DLL 库一起装入内存，由 Windows 读入 DLL 中的函数并运行程序。

可以看出，动态链接是在应用程序被装入到内存时进行的。这样，当多个应用程序调用库中的同一个函数时，不会在内存中有该函数的多个拷贝，而是只有一份拷贝；每个应用程序的可执行文件中装入的只是该函数的内存地址，程序运行时再把应用程序代码与被调用函数代码动态链接起来，从而可以节省内存资源。同时，由于 DLL 与应用程序分开，即使更新 DLL，也不用修改已编译好的可执行文件。

Visual Basic 与库的链接使用的是动态链接方式。在 Visual Basic 应用程序中使用 DLL 有以下几个优点：

(1) Visual Basic 是一种解释执行的程序设计语言，没有编译执行的语言的运行速度快。如果需要在 Visual Basic 中实现一些较为费时的复杂操作，则可以使用由更快的程序设计语言(如 C、C++等)编写的动态链接库。

(2) 动态链接库由 Windows 管理，多个应用程序可以共享同一个 DLL，从而可以节省内存和磁盘空间。当多个应用程序使用某种相同的功能时，可以由动态链接库提供这部分代码，而不必在每个应用程序中重复编写这些代码，动态链接库可以被每个应用程序调用。

(3) 使用动态链接库易于维护用户程序，即对动态链接库的修改不会影响用户程序。例如，厂家可以修改打印机驱动程序以加入新的功能，而这种修改不需要用户重新编译原来的 EXE 文件。

使用动态链接库的主要缺点是：在运行应用程序时，Windows 必须将应用程序所需要的函数从动态链接库中调出来；而如果使用静态链接，则在生成 EXE 文件时就将所需要的函数放入应用程序中，在装入应用程序时，这些函数即同时被装入。此外，整个动态链接库必须随着相应的 EXE 文件一起走，即使只用到其中的一小部分也要这样做。

1.1.3 动态链接库与 API 函数

Windows 本身提供了数以千计的应用程序接口(API)函数，这些函数存放在不同的动态链接库中，按其功能可划分为三大类动态链接库(即 KERNAL、GDI 和 USER)以及许多较小的动态链接库，见表 1.1。

上述大多数动态链接库的名字中都有“32”。API 函数分为两类，即 16 位 API 和 32 位 API。3.0 版的 Visual Basic 只能调用 16 位 API 函数；4.0 版的 Visual Basic 既可调用 16 位的 API 函数，也可以调用 32 位的 API 函数；5.0 版以后，Visual Basic 只能调用 32 位的 API 函数。习惯上把 16 位的 API 称为 Win16 API，而把 32 位的 API 称为 Win32 API。包含 32 位 API 函数的动态链接库的名字中大多含有“32”字样，以示区别。表 1.2 列出了随同 Microsoft 各种操作系统使用的 Windows API 版本。

表 1.1 Windows 动态链接库

名 称	说 明
KERNEL32. DLL	低级内核函数。用于内存管理、任务管理、文件管理、资源控制及相关操作
USER32. DLL	与 Windows 管理有关的函数。包括消息、菜单、光标、插入标志、计时器、通信以及其他大多数非显示函数
GDI32. DLL	图形设备接口库。在该动态链接库中含有与设备输出有关的函数，包括大多数绘图、显示环境、图元文件、坐标及字体函数
COMDLG32. DLL，LZ32. DLL，VERSION. DLL	这几个动态链接库提供一些附加功能。包括对通用对话框、文件压缩以及版本控制的支持。在某些情况下，可以直接访问这些功能

表 1.2 Windows API

API	说 明
Win16	原始的 Windows 16 位 API。这是 Windows 和 Windows for Workgroups 3. x 使用的主 API。16 位模式也得到了 Windows 95 和 Windows NT 的支持。可以用 Visual Basic 16 位版本(包括 3.0 和 4.0)访问这种 API。
Win32	纯 32 位 API。这是 Windows NT 的主 API，可通过 32 位版的 Visual Basic 或 Visual Basic for Applications(VBA)访问这种 API。
Win32c，Windows 95	Win32 的一个子集，由 Windows 95 提供支持(其中的“c”代表 Windows 95 最初的名字，即“Chicago”(芝加哥))。可通过这种 API 调用任何 Win32 函数，但许多函数尚未在 Windows 95 中实现，特别是 NT 特有的一些功能，没被固化到 Windows 95 中。另外，Win32c 也提供了一些特有的 API 函数，这些函数尚未集成到 NT 的 Win32 中。
Win32s	Win32 更小的一个子集。Windows 和 Windows For Workgroups 3. x 支持这种 API，但 Visual Basic 4.0 以后的版本不支持它。

如前所述，Windows 是由几个核心的动态链接库构成的。由于采用了这种特殊的操作系统结构，从而可以相当方便地逐步修改 Windows，不断增加新的功能。例如，当 Microsoft 需要为操作系统增加电子邮件功能时，不必对整个 Windows 进行更新，只需添加一个新的动态链接库即可实现。事实上，近年来在 Windows 中出现的大多数新功能都采用新的动态链接库的形式，而且随着这些新 DLL 的发行，对 API 也有了新的扩展。通常把新增加的 DLL 称为扩展动态链接库。表 1.3 列出了部分主要的扩展 DLL。

Windows API 正处于一个快速进化的阶段，函数的数量和复杂程度不断增加，人们根本无法跟上它的发展。况且为了有效地使用 API 函数，也没有必要成为精通全部 Windows API 的专家。对于一般用户来说，最重要的首先是掌握 Windows 的一般结构和一些基本概念，以解答各种 API 函数的问题；其次是要学习如何阅读 API 文档和函数声明，以便在 Visual Basic 中使用所需要的函数。

表 1.3　部分主要的扩展 DLL

DLL 名称	说　　明
COMCTL32.DLL	实现了一个新的 Windows 控件集，称为 Windows 公共控件（Visual Basic 中的 Toolbar、TreeView、ListView、ImageList 等都属于这个控件集）。该 DLL 最初是为 Windows 95 建立的，目前也适用于 Windows NT。
MAPI32.DLL	提供了一系列 API 函数，为应用程序添加电子邮件功能。
NETAPI32.DLL	提供了一系列用于访问和控制网络的 API 函数。
ODBC32.DLL	用于实现开放数据库链接（ODBC）功能的 DLL。该扩展库中的函数提供了一个标准 API，可以实现与不同数据库的链接。
WINMM.DLL	提供对系统多媒体功能的访问。

1.2　Win32 API 简介

前面讲过，Win32 API 是 Microsoft 32 位平台的应用程序编程接口，所有在 Win32 平台上运行的应用程序都可以调用这些函数。

Microsoft 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，应用程序能充分挖掘 Windows 的 32 位操作系统的潜力，不但可以开发出在各种平台上都能运行的应用程序，而且可以充分利用每个平台特有的功能和属性。

标准 Win32 API 函数可以分为以下几类：窗口管理、图形设备接口、系统服务、窗口通用控制、Shell 特性、国际特性、网络服务。

1.2.1　窗口管理、图形设备接口及系统服务函数

1. 窗口管理函数

窗口管理函数提供了建立和管理用户界面的方法。用窗口管理函数可以建立窗口，并可以通过窗口来显示输出、提示用户输入以及完成其他一些与用户进行交互所需要的操作。

应用程序通过建立窗口类及相应的窗口过程来定义窗口的外观和操作。窗口类可标识窗口的默认属性，例如窗口是否接收双击鼠标操作，是否带有菜单等。窗口过程中所包含的代码用来定义窗口的操作，完成指定的任务以及处理用户的输入。

应用程序可以用图形设备接口（GDI）函数来产生窗口输出。由于所有的窗口都共享显示屏幕，因而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，以便能适合相应的窗口。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收鼠标和键盘输入。系统把鼠标移动、鼠标按钮操作转换为输入消息，并将这些信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，应用程序用消息函数从消息队列中获取信息，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，也可以让系统通过菜单和快捷键将这些低级输入转换成命令消息。系统对所有菜单操作进行管理，包括选择命令，然后向窗口过程发送一个标识该选择的消息。快捷键是应用程序定义的按键操作组合，系统可以将其转换为消息。快捷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。

在应用程序中，可以通过对话框向用户提示附加信息来响应命令消息。对话框实际上是一个临时窗口，用来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件组成，可供用户选择或输入信息。对话框中可包含用于输入文本、滚动文本、从列表中选择项目等操作的控件。对话框管理和处理来自这些控件的输入，使应用程序可以使用这些信息，以完成所要求的操作。

通过“资源”可以共享很多有用的数据，例如位图、图标、字体、字符串等，只要将这些数据作为“资源”添加到应用程序或 DLL 文件中即可实现。应用程序通过资源函数找到所需的资源并将它们加载到内存来获取相应的数据。

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

2. 图形设备接口函数

图形设备接口(GDI)提供了一系列函数的相关的结构，可用来在显示器、打印机或其他设备上生成图形化的输出结果。用 GDI 函数可以绘制直线、曲线、闭合图形、文本以及位图图像。所绘制的图形的颜色和风格依赖于所建立的绘图对象，即画笔、刷子和字体。可以用画笔绘制直线和曲线，用刷子来填充闭合图形的内部，用字体来书写文本。

通过建立设备环境(DC)，可以直接向指定的设备进行输出。设备环境是 GDI 管理的一个结构，其中含有一些有关设备的信息，例如操作方式及当前的选择。应用程序可以用设备环境函数来建立 DC。GDI 将返回一个设备环境句柄，在随后的调用中，该句柄用来表示这个设备。例如，应用程序可以用该句柄来获取有关这个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(如显示器或打印机)进行输出，也可以向一个逻辑设备(如内存设备或元文件)进行输出。应用程序将输出结果记录到元文件后，该元文件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。操作方式指的是文本和背景颜色、混色方式(也称二元光栅操作，用来确定刷子的颜色与绘图区域现有的颜色如何混色)、映射方式(用来指定 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所使用的绘图对象。

3. 系统服务

系统服务函数提供了访问计算机资源以及底层操作系统特性的手段，例如访问内存、文件系统、设备、进程和线程。使用系统服务函数，应用程序可以管理和监视所需要的资源。例如，可以用内存管理函数来分配和释放内存，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。用 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，包括 FAT 文件系统、CD-ROM 文件系统(CDFS)和 NT 文件系统(NTFS)。

系统访问函数提供了可以与其他应用程序共享代码或信息的方法。例如，可以把一些过程放到 DLL 中，使所有应用程序都可以使用；应用程序只要用 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。用通信函数可以向通信端口写入数据或从通信端口读出数据，并可控制这些端口的操作方式。通信函数提供了几种内部通信(IPC)的方法，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，可以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。

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。用系统信息函数可以确定计算机的某些属性，比如是否出现鼠标，显示屏幕上的元素具有多大尺寸等。用注册和初始化函数可以把应用程序的特定信息保存到系统文件中，以便该应用程序的新实例对象或其他应用程序获取和使用这些信息。

用系统服务函数可以处理执行过程中的一些特殊情况，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，用调试函数可以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功能，可以用来建立其他类型的应用程序，如控制台应用程序和服务等。

1.2.2 其他函数

1. Shell 特性

Win32 API 中有一些接口和函数，可用来增强系统 Shell 的功能。

Shell 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。一个名字空间是一个符号的集合，例如文件和目录名字或数据库关键字。名字空间类似于文件系统的目录结构。只是名字空间中包含的是对象，而不是文件和目录。

快捷键(也称 Shell 链接)是一个数据对象，它所包含的信息可用来访问位于 Shell 名字空间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。通过快捷键可以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。

可以用以下几种方法扩展 Shell：

(1) 图标　系统用图标来表示 Shell 名字空间中的文件。在默认情况下，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类文件的默认图标。

(2) 上下文相关菜单句柄　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。当用鼠标右键单击或拖动一个对象时，系统会显示一个上下文相关菜单。该菜单中的命令只应用于被单击或拖动的对象。大多数上下文相关菜单都含有一个属性(Properties)命令，用来显示所选择对象的属性表。一个属性表由一系列重叠的窗口组成

(每个窗口称为一页),用来显示某个对象的有关信息。

(3) 属性表句柄　用属性表句柄可以向系统定义的属性表中添加页,或者替换控制面板的属性表的某些页。

(4) 拷贝挂钩(Hook)句柄　用挂钩句柄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 有一个快速查看(Quick View)命令,使用户可以直接查看一个文件的内容,而不必运行建立该文件的应用程序。文件浏览器提供了一个用来查看文件的用户界面,Shell 通过文件扩展名来确定应运行哪个应用程序。可以为新的文件格式提供文件浏览器,或者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,以确定应生成哪种类型文件的 Quick View。也可以提供其他的文件分析器来支持新的文件类型。

2. Windows 公共控件

系统 Shell 提供了一些控件,可用来设计不同外观的窗口。这些控件是由 DLL 支持的,是操作系统的一部分,因而适用于所有的应用程序。在应用程序中使用通用控件,可以使用户界面与系统 Shell 及其他应用程序保持一致。由于开发控件需要花费一定的时间,因而直接使用通用控件可以节省大量的开发时间。

公共控件是由公共控件库 COMCTL32. DLL 支持的一个控件窗口集。和其他控件一样,一个公共控件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。公共控件 DLL 含有一个编程接口,可以用其中的函数建立和管理控件。

在 Visual Basic 6. 0 中,可以作为 ActiveX 控件直接使用公共控件(见笔者编著《Visual Basic ActiveX 程序设计》)。

3. 国际特性函数

国际特性函数有助于编写国际化的应用程序。Unicode 字符集使用 16 位值表示计算过程中的所有字符,国家语言支持(NLS)函数可以将应用程序本地化,输入方法编辑器(IME)函数可用来输入 Unicode 和 DCBS 格式字符的文本。

4. 网络服务

网络服务函数主要用于网络的操作,包括网络上不同计算机应用程序之间的通信,在网络中各计算机上建立和管理共享资源(如共享目录和网络打印机)的链接等。

网络接口包括 Windows 网络函数、Windows 套接字(Socket)、NetBIOS、RAS、SNMP、Net 函数以及网络 DDE。Windows 9x 只支持这些函数中的一部分。

1.3　在 Visual Basic 中使用动态链接库

前面介绍了动态链接库和 API 的概念。在 Visual Basic 中使用动态链接库,是扩展 Visual Basic 的功能、充分发挥 Windows 系统性能的重要手段。用户可以在 Visual Basic 应用程序中调用这些动态链接库中的函数,通过它们使用和管理 Windows 系统环境及硬件设备。同时,Visual Basic 几乎可以使用由任何语言生成的动态链接库。对于某些 Visual Basic 本身不能实现的功能,可以用其他语言开发,然后将这些函数制作成动态链

接库，在 Visual Basic 中调用。

常用的动态链接库是安装在 Windows 目录下、随 Windows 软件包提供的 DLL。实际上，Windows 操作系统本身就经常使用这些动态链接库，而这些动态链接库中所包含的函数就称为 Windows API 函数。与 Visual Basic 中的函数不同的是，为了调用 API 函数，必须先对要调用的函数加以“声明”。调用时的参数传送则与 Visual Basic 中的函数调用基本相同。

1.3.1 声明

对于 Visual Basic 应用程序来说，动态链接库中的函数是外部过程。为了调用这些函数，必须提供有关的信息。这种提供信息的操作称为声明，Visual Basic 就是通过这种声明来访问动态链接库的。在调用时，Visual Basic 将根据声明确定参数的个数，并进行类型检查。

在不同的语言中，子程序的称谓也不一样。例如在 C 语言或 C++中，子程序一律称为函数，在其他语言中可能称为函数、过程、子例程等。Win32 API 是用 C 语言编写的，因此一律称为函数。而在 Visual Basic 中，子程序一般称为过程。其中，有返回值的过程称为函数过程或 Function 过程，没有返回值的过程称为子程序过程或 Sub 过程。在本书中，将由 DLL 提供的函数称为 Win32 API 函数，或者简称为 API 函数，有时候也称为 API。有的 API 函数需要返回值，有的则没有返回值，其中有返回值的 API 函数与 Visual Basic 中的 Function 过程对应，而没有返回值的 API 函数与 Sub 过程对应。

在调用一个 API 函数之前，必须先在 Visual Basic 的标准模块、类模块或窗体模块中用 Declare 语句进行声明。只有在声明之后，才可以把它作为 Visual Basic 自己的函数来使用。这是因为，API 函数存在于 Visual Basic 应用程序之外的 DLL 文件中，在使用时必须指明 DLL 文件的位置以及相应的调用参数。

在一个 Visual Basic 应用程序(工程)中，要调用的 API 函数只需声明一次，以后即可在程序的任何地方调用。一般来说，API 函数的声明可以出现在三个位置：一是窗体的声明部分，二是标准模块中，三是类模块中。如果出现在窗体的声明部分，则必须作为私有过程声明，即在声明中加上关键字“Private”；如果出现在标准模块中，则应作为公用过程声明，即在声明中加上关键字“Public”(或省略)。当然，如果声明的 API 函数只在标准模块中使用，也可以声明为私有过程。在窗体声明部分声明的 API 函数只能在本窗体模块中调用，其他窗体和标准模块不能调用。而在标准模块中声明为公用的 API 函数，可以在应用程序的任何窗体和其他标准模块中调用。

API 函数的声明通过 Declare 语句来实现。当要调用的 API 函数有返回值时，在 Visual Basic 中作为 Function 过程声明；如果没有返回值则作为 Sub 过程声明。因此，API 函数的声明有两种格式。

格式 1：

```
Declare Sub API 函数名 Lib "库名" [Alias "别名"]([参数表列])
```

格式 2：

```
Declare Function API 函数名 Lib "库名" [Alias "别名"] ([参数表列]) As 类型
```

不难看出，格式 1 用来声明没有返回值的 API 函数，而格式 2 用于有返回值的 API 函数。例如：

```
Declare Sub InvertRect Lib "User" (ByVal hDC As Integer, aRect As RECT)
```

将用格式 1 声明一个 API 函数，而

```
Declare Function GetSystemMetrics Lib "User" (ByVal n As Integer) As Integer
```

则用格式 2 声明一个 API 函数。

在 C 语言中，有些函数的类型为 void，即没有返回值。对于这类函数，在 Visual Basic 中应声明为 Sub 过程（即格式 1）；其他类型的函数均有返回值，必须声明为 Function 过程（格式 2）。

Declare 语句的两种格式基本相同，只是 Function 过程的声明需要有返回值类型。现将 Declare 语句的各种成分介绍如下：

1. API 函数名

“API 函数名”指的是 DLL 中的函数名，同时也是 Visual Basic 应用程序中使用的过程名。也就是说，在 Visual Basic 声明的过程名与 DLL 中的函数名是相同的。

2. 库名

“库名”指的是函数所在的动态链接库，即 DLL 文件的名称，它告诉 Visual Basic 要使用的动态链接库在什么地方。“库名”在 Lib 子句中指定，可以含有完整的路径，也可以只给出文件名（即库名）。如果带有完整的路径，则在指定的目录中查找 DLL，例如：

```
Declare Function lzcopy Lib "c:\win\lzexpand.dll" (ByVal S As Integer, _
        ByVal D As Integer) As Long
```

一般来说，如果要调用的 API 函数属于 Windows 核心库（如 User32、Kernel32 或 GDI32），则在指定“库名”时可以不给出完整路径，并可省略扩展名 DLL，例如：

```
Public Declare Function GetTickCount Lib "kernel32" () As Long
```

而如果要调用的 API 函数不属于 Windows 核心库，则在 Lib 子句中应指定 DLL 的路径，扩展名 DLL 也不能省略，例如：

```
Declare Function ExamFunc Lib "c:\windows\Example.dll" (Byval x As Integer, _
                                    y As Integer) As Long
```

如果不带有路径，只给出文件名，则 Visual Basic 按以下顺序查找所需要的 DLL：

(1) EXE 文件所在的目录；

(2) 当前工作目录；

(3) 32 位 Windows 系统目录；

(4) 16 位 Windows 系统目录；