ZIFF-DAVIS
ZD
PRESS

WINDOWS DESKTOP 实用程序集粹

WINDOWS DESKTOP UTILITIES

JEFF PROSISE

佟金荣　姜静波　译
姜静波　校

電子工業出版社

Windows Desktop
Utilities
Jeff Prosise

Windows Desktop实用程序集粹

佟金荣　姜静波　译
姜静波　校

電子工業出版社

（京）新登字055号

内 容 提 要

本书是美国著名PC专栏作家、操作系统专家为Windows用户和Windows程序员写的一本书，它通过作者精心组织的十个Windows Desktop实用程序详细介绍了大多数书籍尚未涉及的Windows编程技术和技巧。通过阅读本书，Windows普通用户可以学会这十个实用程序的用法，大大加强使用Windows的能力；Windows程序员可以学会大量鲜为人知的Windows编程技术，今后能编写出更高水平的Windows应用程序。

Windows Desktop实用程序集粹

〔美〕Jeff Prosise 著
佟金荣 姜静波 译
姜静波 校
责任编辑 贺 华
*
电子工业出版社出版（北京市万寿路）
电子工业出版社发行 各地新华书店经销
北京平谷玉福印刷厂印刷

开本：787×1092毫米 1/16 印张：26.75 字数：640 千字
1995年 1 月第 1 版 1995年 1 月第 1 次印刷
印数：5000 册 定价：43.00 元
ISBN 7-5053-2593-0/TP・781

出 版 说 明

计算机科学技术日新月异，为了引进国外最新计算机技术，提高我国计算机应用与开发的水平，中国电子工业出版社与美国万国图文有限公司合资兴办的北京美迪亚电子信息有限公司取得了美国Ziff-Davis Press的独家版权代理。Ziff-Davis Press授权本公司通过电子工业出版社等出版机构全权负责在中国大陆出版该公司的中文版和英文版图书。现在与广大读者见面的是最近推出的第一批图书。今后我们还将陆续推出Ziff-Davis Press的最新计算机图书和软件，为广大读者提供更好的服务，传递更多的信息。

美国Ziff-Davis Press是全美最大的计算机出版商之一，它出版的书籍、杂志和光盘，主办的展览和会议，提供的咨询和网络服务，形成了整个行业潮流的主导。我们优选翻译出版的图书是Ziff-Davis Press的最新计算机图书，并采用了该公司提供的电子排版文件，从而提高了质量并大大缩短了图书的出版时间，从根本上改变了以往翻译版图书要落后原版书较长的“时差”现象，这在电子技术日新月异的时代具有深远的意义。

北京美迪亚电子信息有限公司

1994年3月

献给Lori

致　谢

在计算机界，无论我走到哪里，都能碰到一些聪明的人在寻求使艰苦的工作变得容易的方法。我要把我的感谢范围扩大到Ziff-Davis Press公司中所有为本书的编辑和出版贡献了力量的人们，特别是要感谢Barbara Dahl、Deborah Craig和Neil Rubenking。我要感谢Cindy，作为本书的发行者她非常冷静。另外，我还要感谢在我那些在CompuServe服务网上的朋友，他们自愿地利用他们的宝贵时间测试了本书中的软件，并且向我报告了他们发现的程序错误。最后，感谢仁慈的主，他使我和我的全家沐浴在幸福之中，他给我们的远比我所祈求的要多。

引　言

如果你最近才开始转向使用Windows，那么很可能会为Windows给PC增加的强大新功能而激动得难以平静。例如，有了Windows以后，不但同时运行几个应用程序变得异常地容易，而且这些程序之间还可以利用剪贴板程序Clipboard来共享数据。至于动态数据交换(DDE)以及对象连接和嵌入(OLE)这样的高级互操作特性，使共享数据的这一概念达到了空前的高度。Windows的保护模式设计摆脱了MS-DOS实模式带来的内存限制，这样我们可以充分地利用PC的RAM中的每个或每兆字节，而再也不是只能使用640K的字节。当内存不够时，我们可通过技术代之以硬盘空间，“虚拟内存”是一个从大型机借用来的特性。Windows的可视化设计与其应用程序用户界面的一致性相结合，使新程序的学习再也不是一件乏味的事情了。

但是，甚至Windows也有局限。大家只要使用Windows一段时间之后，就会发现它也有鞭长莫及之处。那么，Windows到底缺些什么呢？一言以蔽之：实用程序。多年以来，广大的MS-DOS用户已经编写了许许多多能够满足各种要求的实用程序，简直是应有尽有。（确实，这种说法有些言过于实！）相比之下，大多数的Windows用户或者没有时间来建立各种的实用程序，或者因为它们还根本没有被编写出来而无法找到需要使用的实用程序。例如，没有实用程序的帮助，Windows 就无法为我们提供用来检索硬盘以寻找包含某个单词或短语的文件或者将一个CD-ROM驱动器变成一台光盘播放机所需要的工具。如果没有实用程序的帮助，Windows可能如同MS-DOS一样的枯燥无味。

软件驱动着世界的进步

本书的编写目的，是通过一些设计用来增加用户生产率和能够使Windows成为一个更加令人喜爱的工作环境的桌面实用程序来加强Windows。本书介绍的十个实用程序各个功能并且具有极高的商业价值。这些Windows实用程序包括：

- FastExit，它可以在Windows的桌面上增加一个出口标志，用户只需双击这个标志就可以终止Windows。
- QuickStart，它通过把用户最常使用的应用程序列表保存在一个菜单中并在Windows的桌面上弹出这个菜单的方式来加速程序的启动过程。
- FindFile，它可以同时在几个磁盘上检索出用户指定了名字的文件，在几秒钟之内它就能够存储在任何磁盘上的任何文件。
- FindText，它可以在整个磁盘或磁盘的某部分检索出含有指定单词或短语的文件。
- RPNCalc，它为给Windows增加了一个基于逆波兰表示的HP风格的计算器。
- WinDial，它为Windows增加了一个电子电话目录和一个电话拨号程序。
- CDPlayer，它把PC的CD-ROM驱动器变成了一个立体声光盘唱机。
- AllPaper，它可以伸缩Windows的壁纸文件，使之能够适应任何的屏幕。

"chiseled-steel"式外观。

- 利用对话框作为主窗口，这一技术既可以节省实用程序的代码又可以节省开发时间。
- 利用Windows内部提供的Media Control Interface对用多媒体设备，例如CD-ROM驱动器，进行编程。
- 把存储在BMP文件中的图像转换为在窗口中可以显示的位图，并且在不牺牲设备的无关性的前提下对图像进行操作。

这些技术都是Windows程序员经常需要解决的问题。我并没有把这本书过分地称为"高级的Windows编程书"，但把它称为一本"Windows应用程序编程书"应该是当之无愧的。到现在为止，市场上最好的一本Windows编程书籍是由Jeffrey Richter编写的《Windows 3.1: A Developer's Guide》。读者可以利用本书给出的源代码和所讨论的编程技术来作为自己编程的出发点，也可以在本书给出的程序的基础上对他们进行改进使其更加完善。

关于作者

在编写本书中的实用程序时，我利用的是MicrosoftC/C++7和Microsoft的Windows Software Development Kit（SDK）。经过深思熟虑和同我的好友Neil Rubenking（本书的技术编辑）进行了长时间的讨论之后，我决定不要为了使这些程序在Borland C++和其它的编译系统中也能够被编译而对它们做一些必要的修改。当然，用户仍然可以利用Borland、Symantec或其它的编译程序来编译这些程序，但在编译之前可能需要对相应的make文件做一点必要的修改。为此，你可能需要解释为什么Microsoft要在函数名字的前面加上一个下划线，例如在并不属于ANSI C标准的_itoa函数前面增加一个"_"。这种约定并没有被广泛采用。我认为，真正的程序员编写的代码并不需要在10个不同的编译下进行编译，那种情况只能出现在书或杂志上。我不喜欢只是为了适应不同的编译程序这样一个简单目的就在代码中增加很多的#define、#ifdef。另外，其它的编译程序有时流行有时又会不流行，但是当使用Microsoft C进行Windows编程时，所编出的程序就会永远地符合标准。

如果你不同意我的观点，那么我倒很想听听你的意见。实际上，我衷心希望能够听到关于本书的任何意见，包括对将来再版的建议。你最好通过MCI Mail或CompuServe服务网与我联系。在MCI Mail上，我的用户名是JPROSISE，ID号为312-3074。在CompuServe服务网上，我的ID号是72241,44。我负责ZiffNet网络上的"Programming and Tips/Utilities"论坛，这是Ziff-Davis在CompuServe服务网上拥有的一块小天地。每天，我至少要在那里查看两次消息。另外，我还要定期查看CompuServe服务网的e-mail系统。如果你无法通过这些电子媒介与我联系，请你来信。来信请寄：

Jeff Prosise
c/o Ziff-Davis Press
5903 Christie Avenue
Emeryville, CA 94608

目　录

第一章 Windows及其编程入门

对于程序员来说，PC有着无穷的潜力等待着他们去开发。特别是进行Windows编程时，情况尤为如此。Windows是一个丰富的，面向图形的环境，它的开发将把我们载入二十一世纪的下一代应用程序的理想环境。尽管Windows 的功能已经十分强大，但是它仍然缺少一些每个Windows用户在使用Windows系统时都需要的基本工具，如电话拨号程序、正文搜索程序以及弹出式程序启动工具。

本书有两个目的，第一个目的是为Windows用户提供一些Microsoft没有提供的基本工具；第二个目的是用实际的、真正的源代码来说明Windows程序的工作方式。如果读者已经是一名Windows程序员，那么可以利用本书所介绍的思想和技术来改进自己的程序。如果读者虽然以前从未进行过Windows编程，但对它的编程思想非常感兴趣，那么可以从本书中直接了解到利用Microsoft C和Windows软件开发工具集（SDK）进行Windows编程的方法。

对于涉及到Windows复杂内容的书籍，特别是那些包含Windows程序源代码的书籍，最好都先介绍一下Windows和Windows程序是如何工作的。Windows是一个复杂的操作环境，它与DOS这类基于字符的传统操作系统不同。实际上，Windows要对程序的操作进行全面的管理，而不是启动一个应用程序之后就放任自流。对于一个Windows程序来说，它只有在收到Windows发来的通知时，才会绘制其窗口（这点与DOS程序相反，DOS程序只要高兴就可以在任何时候绘制屏幕），只有再次收到Windows通知时，才会终止这一操作，如此等等。在Windows环境下，许多普通的事件，如用户使用鼠标器拖着窗口在屏幕上移动或者单击选择Minimize按钮将一窗口变成一个图标，都是由Windows处理的，而无需用户程序进行任何干预。Windows程序甚至不要求用户使用传统的方式进行输入；当移动一下鼠标器、单击选择一个按钮或者按下一个按键时，Windows会把这一事件通知给相应的程序并给它一个响应这一事件的机会。

在本章中，读者将能学到Windows的工作方式以及Windows应用程序构造方式这些方面的基本知识。即使读者对Windows编程不感兴趣，通过本章的学习你也可以深入地了解Windows，并且学到Windows程序是怎样被编译成可执行形式的。此外，你还可以看到一个简单Windows实用程序的源代码，这个实用程序可以把一个出口标志放在Windows的桌面窗口上，并且当使用鼠标器双击选择该标志时可以终止Windows的执行。

Windows工作方式的总体概述

实质上，Windows是个消息驱动的操作系统，Windows应用程序所执行的每个操作几乎都是对Windows本身发给该应用程序的消息的响应。例如，应用程序接到一条WM_PAINT消息便重绘窗口，接到一条WM_COMMAND消息便对选择屏幕上按钮的鼠标器单击事件作出响应。在没有接到消息时，Windows应用程序就什么也不做。这与DOS程序形成了鲜明的对照，

DOS程序为了得到输入要不断地对计算机进行轮询。在DOS应用程序运行期间，它全时占有CPU。

图1.1说明了当按下一个键或鼠标器按钮时，Windows内部所出现的情况。Windows首先在“系统消息队列”中生成一条描述该事件的消息，然后将这条消息拷贝到相应“应用程序的消息队列”中。所谓“消息队列”也就是一块内存，这块内存用来存储消息一直到消息被检索读出。系统消息队列只有一个，但每个应用程序都有一个自己的消息队列。应用程序从应用程序消息队列中检索读出消息并将其分派给相应的窗口过程，窗口过程包含了所有相关消息的处理程序。例如，若按了一个键，窗口过程就接到一条WM_KEYDOWN消息；若按了鼠标器的左按钮，窗口过程就接受到一条WM_LBUTTONDOWN消息。当窗口过程处理完相应的消息之后，再将控制归还给Windows。

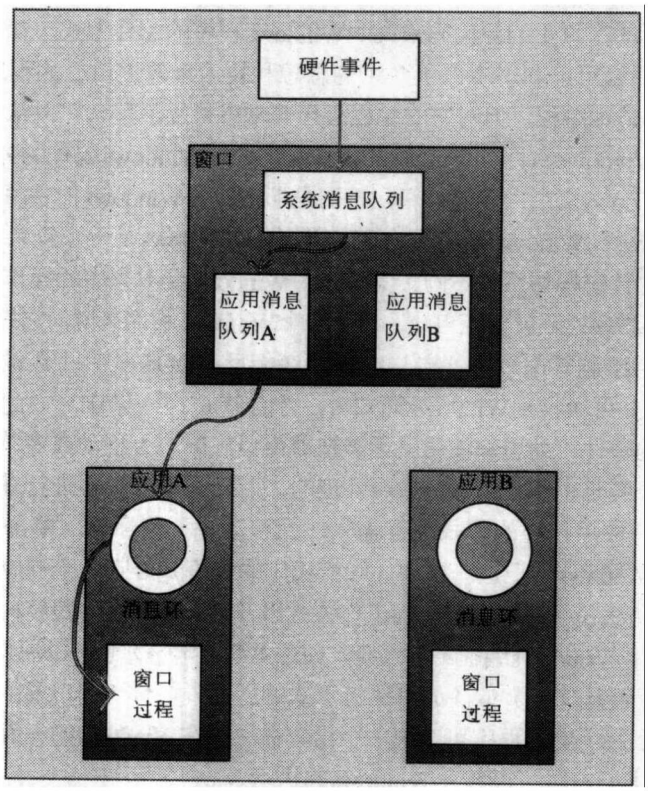

图1.1 鼠标器和键盘消息的处理流程

并不是所有消息都是在系统消息队列中产生的，也不是所有消息在到达窗口过程的途中都要经过应用程序消息队列。Windows将绝大多数非鼠标器或非键盘消息直接发送给应用程序的窗口过程，绕过了其消息队列，从而保证了重要的消息能得到迅速的处理。

Windows制定了一个粗糙的多任务调度方式，这种方式通常被称为“协同式多任务调度”或“不可抢占式多任务调度”，即每当一应用程序试图从它的消息队列中检索一条消息而该

消息队列又为空时，就将控制让给其它的应用程序。这种调度方式的危险在于倘若一个Windows程序运行很长一段时间都不检查它的消息队列，那么其它的程序在此期间将处于挂起状态。将来新版的Windows将使用基于硬件的多任务调度方式代替上述调度方式，它将利用硬件定时器为所有正在运行的程序分配CPU时间，从而形成真正的抢占式调度方式。这种更先进的多任务调度方式已被Windows NT采用。

应用程序编程接口

从最简单的意义上讲，一个Windows程序基本上就是集成在一个程序中的事件处理程序的集合。Windows向应用程序发送一条消息，通知它一个事件已经发生或即将发生，应用程序对相应的消息做出响应。各个Windows应用程序消息响应方式的变化性和独特性使它们彼此各不相同。

Windows有900多个相同的编程函数，应用程序可以调用这些函数来做一些非常有意义的工作，例如在窗口的客户区内绘制正文或打开一个文件（所谓窗口的“客户区”也就是窗口边界以内的窗口可见区域，应用程序在客户区内可以自由绘画。Windows负责对窗口中客户区以外的部分进行维护，其中包括窗口边界、标题棒和菜单棒。这些区域以及窗口中客户区以外的其它一些区域就组成了窗口的“非客户区”）。这些编程函数就组成了“Windows应用程序编程接口”，简称为Windows API。Windows程序员要想使自己成为一名专家，关键的难点就在于要熟悉Windows API函数，Windows API函数不但数量繁多而且各不相同。幸运的是，在最初开始编写Windows程序的时候，并不需要必须对每个API函数都很熟悉。绝大多数的Windows程序，包括本书所描述的实用程序，使用的都只是API全集中的一个子集。通常即使是一些资深的Windows程序员，对某些生僻的Windows函数也并不熟悉。

学习API的最好途径就是考察一下Windows程序的源代码。一些API函数，例如RegisterClass和CreateWindow，在每个Windows程序中几乎都要用到。而其它一些API函数，象CreateEllipticRgnIndirect，却只在少数一些Windows程序中才会用到。Windows SDK包括了以书面和电子文档两种形式提供了Windows API的完整参考信息，它列出了所有的Windows API函数。许多API函数都有多个参数，当用户首次接触时会感到很棘手。例如在SDK中，经常要用到的CreateWindow函数的文档说明如下：

```
CreateWindow (2, x)

HWND CreateWindow (lpszClassName, lpszWindowName, dwStyle, x, y, nWidth,
nHeight, hwndparent, hmenu, hinst, lpvParam)

LPCSTR lpszClassName;        /* address of registered class name      */
LPCSTR lpszWindowName;       /* address of window text                */
DWORD  dwStyle;              /* window style                          */
int x;                       /* horizontal position of window         */
int y;                       /* vertical position of window           */
int nWidth;                  /* window width                          */
int nHeight;                 /* window height                         */
```

```
HWND    hwndParent;     /* handle of parent window                    */
HMENU   hmenu;          /* handle of menu or child-window identifier */
HINSTANCE hinst;        /* handle of application instance             */
void FAR* lpvParam;     /* address of window-creation data            */
```

总的来说， CreateWindow有11个不同的输入参数， 这些参数对从窗口的初始大小和位置（x,y,nWidth和nHeight）到将在标题棒中显示的标题（lpszWindowName）每个方面都进行了说明。这一说明含有如此之多的信息，这使用户不得不从这些C语言的描述中理出它的头绪。但当用户编写几个Windows程序之后，就会习惯成自然地学会CreateWindow的使用。 因此，如果你以前没有编过Windows程序，那么不要失望。请放心，一旦尝试几次后，你一定会发现编写Windows程序变得比较容易一些。

消息、消息还是消息

现在，在读者的脑海中应该牢牢建立起这样的概念：Windows程序中发生的每件事几乎都是消息产生的结果。总而言之，Windows定义了200多条不同种类的消息，并且若程序需要，还可以另外定义一些应用程序专用的消息。

消息可以分为八大类：

- 输入消息：当任何程序输入事件发生时，就要把这类消息发给该程序的窗口过程。这类消息包括各种鼠标器消息和键盘消息，它还包括具有多种用途的WM_COMMAND消息和WM_TIMER消息。其中WM_COMMAND消息负责通知窗口过程有某个菜单项已被选择，或者某个子控制（如按钮、编辑框或单选按钮这些专用窗口）有一条来自其父窗口的消息；而WM_TIMER是一条当程序分配了一个叫“定时器”的 Windows资源之后以特定的时间间隔发送的消息。
- 控制消息：这类消息有一个程序向其子控制发出，它们用来改变子控制的行为或从子控制获取信息。例如，EM_LIMITTEXT消息的用途是限制一个编辑控制所能接受的输入字符数量，而LB_ADDSTRING则用来向列表框增加一个列表项。
- 窗口管理消息：当在窗口的生存期内发生了重要的事件时，Windows就向应用程序发送这类消息。例如，WM_CREATE、WM_PAINT和WM_DESTROY都是窗口管理消息。当建立窗口时，就要向相应的窗口过程发送一条WM_CREATE消息；当需要重新绘制窗口的客户区时，就要向相应的窗口过程发送一条WM_PAINT消息；当窗口被破坏了时，就要向相应的窗口过程发送一条WM_DESTROY消息。
- 非客户消息：这类消息用来通知一个窗口其非客户区中发生了一个事件。例如，WM_NCPAINT消息用来通知相应的窗口过程其窗口的非客户区需要重新绘制。在通常情况下，应用程序将忽略这条消息，即允许Windows重新绘制其非客户区。但是如果愿意的话，应用程序也可以自行处理WM_NCPAINT消息并改变窗口的外观。
- 剪贴板消息：这类消息用来控制裁剪、拷贝和贴粘这些标准的剪贴板操作，并且通知应用程序发生了剪贴板事件。
- DDE消息：当两个应用程序通过DDE（动态数据交换）交换信息时就相互发送这类消息。

- MDI消息：这类消息用来管理MDI（多文档界面）窗口。
- 系统消息：这类消息用来通知Windows应用程序Windows系统发生了某种事件或产生了某些变化。例如，WM_PALLETTEISCHANGING消息用来通知一个应用程序另外一个应用程序正在改变系统的硬件调色板。

当程序接到一条消息时，该消息还带有另外三个信息项：消息将要发到的窗口的句柄、一个称作wParam的16位整数和一个称作lParam的32位整数（wParam中的w代表字（word），lParam中的l代表长字（long））。一般说来，wParam和lParam还包括关于该条消息的另外一些信息。例如，当接到一条WM_LBUTTONDOWN消息时，lParam的低16位包含了鼠标器的x坐标，高16位包含了鼠标器的y坐标。wParam含有的是一些标志位，它们分别用来标识Shift键、Ctrl键以及鼠标器的中间及右按钮的状态（Windows可以支持最多有三个按钮的鼠标器）。根据这些额外的参数，接到消息的程序可以准确地辨别出在按鼠标器按钮时鼠标器指针的精确位置以及Shift键和Ctrl键是否处于按下状态，还可以辨别与此同时是否还按下了别的鼠标器按钮。

表1.1列出了Windows应用程序最常接收的25条消息。这个表并不完全，它只是用来说明Windows可以支持许多不同的消息，此处它只是用来帮助读者深入理解Windows的消息系统。我们将在遇到源代码程序清单中的具体消息时，再讨论包含在wParam和lParam中的特定信息。

表1.1 常用的Windows消息

消　息	发 送 条 件
WM_CHAR	按下一个字符键
WM_COMMAND	选择一个菜单项或发生了一个控制窗口事件（例如按下一个按钮或选择一个列表项）
WM_CREATE	建立窗口
WM_DESTROY	删除窗口
WM_HSCROLL	单击水平滚动棒
WM_INITMENU	显示菜单
WM_KEYDOWN	按下一键
WM_KEYUP	释放一键
WM_KILLFOCUS	窗口不再是当前输入窗口
WM_LBUTTONDBLCLK	双击鼠标器左按钮
WM_LBUTTONDOWN	按下鼠标器左按钮
WM_LBUTTONUP	释放鼠标器左按钮
WM_MOUSEMOVE	移动鼠标器
WM_PAINT	窗口需要重新绘制
WM_QUERYENDSESSION	窗口准备终止
WM_QUERYOPEN	恢复最小化的窗口
WM_QUIT	应用程序终止
WM_RBUTTONDBLCLK	双击鼠标器右按钮
WM_RBUTTONDOWN	按下鼠标器右按钮
WM_RBUTTONUP	释放鼠标器右按钮
WM_SETFOCUS	窗口变成当前输入窗口
WM_SHOWWINDOW	隐藏窗口或显示窗口
WM_SIZE	窗口大小发生了改变
WM_TIMER	定时器的时间间隔已到
WM_VSCROLL	单击垂直滚动棒

Windows程序没有必要对发送给它的每一条消息都进行处理。实际上，多数的Windows程序会忽略上述大部分消息，而只处理那些有关的消息。通过使用Spy这类的实用程序可以侦察监视系统中往来的消息，图1.2给出了Spy的窗口，该窗口显示了当用户使用鼠标器双击选择窗口左上角的系统菜单从而终止了某个应用程序时发送给该应用程序的全部消息。大多数用户在首次使用Spy时，看到系统中有如此大的消息传输量会感到很吃惊。但要记住这样一件重要的事情：传给应用程序的绝大多数消息尽管对Windows很有意义，但在用户程序的代码中并不必对它们进行处理。在本书给出的Windows实用程序中，大多数实用程序处理的消息都不多于十种。

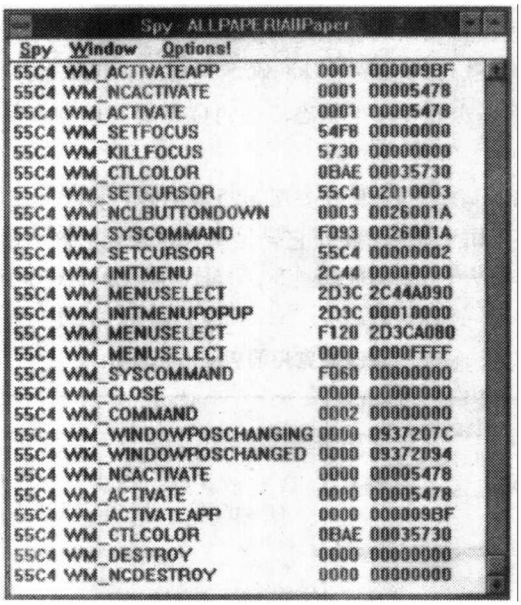

图1.2 Spy程序的窗口

Windows程序的组成

Windows程序的C源代码文件主要由两部分组成：一个WinMain函数和一个窗口过程。其中WinMain相当于C的主函数。当应用程序被启动后其WinMain函数就得到了控制权，而在WinMain函数返回时该应用程序也就结束了。窗口过程是负责接受和处理发给应用程序的消息的函数。每当WinMain中的一个子例程从应用程序的消息队列中检索到一条消息并将它分派给相应的窗口过程时，就要调用这个函数。另外，通常在绕过应用程序的消息队列而直接将消息发送给应用程序时，也要调用这个函数。了解WinMain函数和窗口过程是理解Windows程序工作原理的第一步，也是最基本的一步。

WinMain函数 Windows程序开始于WinMain函数也结束于WinMain函数。WinMain函数主要完成三个重要任务：

- 建立同外部环境接口的窗口。
- 从应用程序消息队列中检索消息并将其分派给相应的窗口过程进行处理，这部分程