北京希望电脑公司 Windows 技术丛书

- 面向窗口程序开发
- 突破窗口编程难点
- 提高窗口编程水平

Microsoft Windows 3.0
编程实例与技巧

博山 尚琼 编译
希望 审校

海洋出版社

Microsoft Windows 3.0
编程实例与技巧

北京希望电脑公司 Windows 技术丛书

Windows 程序员必备

Microsoft Windows 3.0

编程实例与技巧

博山　尚琼　编译

希望　　审校

- 面向 Windows 程序开发
- 突破 Windows 编程难点
- 提高 Windows 编程水平

海洋出版社

内容提要

本书讲述了 Windows 程序设计的基本技巧，着重讨论了窗口管理、子窗口和图形设备接口(GDI)等基本问题，并给出了 200 多个高级实用例程。利用书中提供的例程，用户就不必强迫自己掌握许多全新的概念，就可以编写出自己的 Windows 应用程序。书中还提供了 8 个样板程序，以便用户尽快地掌握 Windows 的基本工作机理，提高 Windows 程序设计水平。

本书是 Windows 程序设计人员必备的参考书，供大专院校计算机专业师生参考，也适合广大计算机用户阅读。

欲购本书的用户，请直接与北京 8721 信箱联系，邮政编码 100080，电话 2562329。

MICROSOFT WINDOWS 3.0

编程实例与技巧

博山　尚琼　编译

希　望　审校

责任编辑：阎世尊

*　　*　　*

海洋出版社(北京市复兴门外大街 1 号)

海洋出版社发行　双青印刷厂印刷

*　　*　　*

开本：787×1092 毫米 1/16　印张：29.25　字数：650 千字

1991 年 5 月第一版　1991 年 5 月第一次印刷

印数：1—3000 册　定价：18.00 元

ISBN7-5027-2206-8/TP・34

前　言

Microsoft Windows 是一个多任务环境，它覆盖了 DOS 的全部功能，提供了图形用户界面、高级应用程序编程接口和丰富的软件开发工具包(SDK)，实现了动态数据交换、模块动态连接、自动内存管理等功能；Microsoft Windows 是一个完善的软件开发环境，它采用了面向对象的程序设计技术，可以对开发任务进行自动处理，提供了直观优美的公用用户界面，充分发挥了微型计算机的潜力。

自从 Microsoft Windows 诞生以来，它就受到了广大用户的欢迎，目前在 Windows 上开发运行的优秀软件不胜枚举。在我国，Windows 也受到了广大用户的青睐，许多用户和开发人员都准备开发支持 Windows 的应用程序，但是，由于 Windows 引入了许多全新的概念，例如，消息、资源、窗口管理和图形设备接口等，它是一个基于消息、面向函数调用的系统，所以，Windows 环境被公认为是难于编程的。

在此，我们特别向广大 Windows 程序员推荐《Microsoft Windows 3.0 编程实例与技巧》。本书(带配套磁盘)试图让用户更容易地学会和使用 Windows 的软件开发工具包(SDK)。有经验的 Windows 程序员都知道，SDK 是 Windows 环境的主要编辑界面。它提供了 500 个 Winows 界面控制函数，涉及从窗口管理到动态数据交换等许多内容。

本书中的 200 多个实例旨在使 Windows 程序设计变得更为容易。例如，想要创建一个窗口时，只需利用书中提供的例程，而不必强迫自己掌握窗口风格的基本概念(当然，最终您自然就会明白这些概念)。总之，Primer 库让您仅在理解为数不多的概念之后，就可以开始着手 Windows 程序设计。

本书的实例由 Primer 库中的例程和示例程序中的过程组成。利用这些例程，您可以了解 Windows 的工作机理。大多数实例说明了如何完成一个特定的工作，在编程实践中，您可以在应用程序中使用这些例程。某些例程完成某些特殊任务，而其他例程则是大多数应用程序中经常使用的例程。

本书包含了关于 Windows 3.0 版的大量编程专题，包括位图菜单、组合对话框和自绘按钮等。此外，书中还着重讨论了 Windows 程序设计的基本问题，如窗口管理、子窗口和图形设备接口(GDI)。

最后，我们希望读者发现此书非常有用，并祝您在 Windows 环境中编程时获得美好的享受。

译者于北京

1991 年 5 月

目　录

第六章 GDI 正文工具

第七章 GDI 图形

第八章 其它 GDI 例程

导　论

在 DOS 世界的边沿徘徊了六年之后，Microsoft Windows 终于登堂入室了。其成功的主要原因与硬件有关，有了高速硬件和大量内存，运行图形用户界面就容易得多了。当然，Windows 3.0 本身也为程序员和用户进行了许多改进。

本书及配套磁盘试图让用户更容易地学会和使用 Windows 的软件开发工具包(SDK)。有经验的 Windows 程序员都知道，SDK 是 Windows 环境的主要编辑界面。它提供了 500 个 Winows 界面控制函数，涉及从窗口管理到动态数据交换。由于 Windows 是一个基于消息、面向函数调用的系统，所以，Windows 环境被认为是难于编程的。

本书中的 200 多个例程旨在使窗口程序设计变得更为容易。例如，想到要创建一个窗口，只需利用书中提供的例程，而不必强迫自己掌握窗口风格的基本概念(当然，最终您自然就会明白这些概念)。总之，Primer 库让您仅在理解为数不多的概念之后，就可以开始着手窗口程序设计。

本书的例程由 Primer 库中的例程和示例程序中的过程组成。通过这些例程，您可以了解 Windows 是如何工作的。在大多数情况下，例程说明了如何完成一个给定的工作，在编程实践中，您可以在应用程序中使用这些例程。某些例程完成某些特殊任务，而其他例程则是大多数应用程序中经常使用的例程。

本书除了包含了大量关于 Windows 3.0 版的编程专题，包括位图菜单例程、组合对话框和自绘按钮等。此外，书中还着重讨论了 windows 程序设计的基本问题，如窗口管理、子窗口和图形设备接口(GDI)。

使用本书例程的软件需求与 Windows 3.0 编程相同。包括：Windows 3.0、Windows 3.0 软件开发工具包和 Microsoft C 编译程序(5.1 或 6.0 版)。

除软件需要之外，开发 Windows 应用程序还需要一个性能可靠的计算机。作者建议采用 80386 或 80486。目前，计算机系统的内存和磁盘容量可以不同，但作者建议至少有 2M 内存和 60 兆的硬盘空间，以获得较高的性能。当然，少一点也不要紧，但是，在硬件性能价格比较高的今天，作者以为多花几个美元来购买高性能的硬件是值得的。

最后，我们希望读者发现此书非常有用，并祝您在 Windows 环境中编程时获得美好的享受。

第一章 Windows 入门

本书中的例程旨在使您能够尽快地进行 Windows 编程。这些例程您可以调用，也可写到自己的函数中。为了让您方便地使用这些函数，我们还提供了配套磁盘（需要本书配套磁盘的读者，可与本书封底的地址联系）。

例程的设计采用了结构化的代码组织技术，包括例程的可读性、例程的细节和返回值的测试等。使用这些函数有两种方式：

1. 直接用在程序中。例如，若程序中正好缺少控制字体的函数，您就可以利用本书给出的字体例程之一。

2. 当需要进一步完善程序时，您可以根据需要改变例程。若对例程的性能感到满足，则不必对它们进行修改。

例程尽可能面向应用程序开发。若不用这些例程，您也可以花费时间来开发自己的例程，但如果您修改了例程，则需考虑本书中的许多用到该例程的地方。

用于初始化与创建窗口、菜单和对话框的例程具有较少的独立性，程序中可以重新利用这些例程，例如，例程 CreateXXListbox 可以调用许多次。

1.1 例程名

本书中的命名约定与 Microsoft 和 Windows 程序所用的约定基本相同，但也有略微不同的地方，这主要是受 Presentation Manager 编程的影响。例如，窗口句柄的标准名为 hWnd，而本书却使用了 PM 方式，hwnd。

所有例程根据其用途来命名（这种方法出现在 Windows 工具包函数之后）。一般来说，例程是 2 个或 3 个英文字的组合，在第一个字之后是 1～3 个字头组成的助记符。

助记符对函数进行了说明。其后是函数名中的最后一个字，它用于区分特殊函数。例如，CreateODicon 以图标形式创建一个标准窗口，OD 表示窗口是一个重叠的缺省窗口，icon 表示窗口将以图标形式创建。

1.2 例程的组织

例程是按照专题来组织的。每章讨论一个专题及相应的函数。在一章中，函数按字为顺序列出。

若专题要求对已有例程进行改进，则改进原例程并给出新例程。例如，当要为一个动态数据交换（DDE）程序创建一个标准窗口时，需要一个与通常标准窗口例程显著不同的例程。在这种情况下，我们就设计一个新例程。

1.3 安 装

安装例程最简单的方法是将它们拷贝到一个子目录中。配套磁盘上安装处理文件在\WINDEV 目录中创建一个子目录，若\WINDEV 不存在，则子目录建在根目录下。

例程按照章节放置在\WINDEV\ROUTINES 目录，每个文件对应于一个例程，文件出现的顺序与书中相同。含单个例程的文件以.C 为扩展名。每章有一个 CH??.H 文件（建议把该

文件用作头文件)，目录中的其他文件包含了该章的版本信息。

函数名尽可能产生便于理解的唯一文件名(最长不超过 8 个字符)。在许多情况下，函数名的第一个字被忽略，因此，函数如 CreateODicon 变成 ODicon. c。

1.4 应用程序模式

Windows 中最重要的概念是用户界面。除此之外，就是理解某些基本概念，如呈现空间(presentation)和消息，以及面向对象的程序设计。本章的其余部分综述了其中的某些概念，以及 Windows 的内容。Windows 的最重要部分之一是 SDK 工具包，包括其文件、数据类型、函数库、命名约定和函数说明。

Windows 中需要知道的不是其内部机制，而是 Winodws 可以管理多个过程。除此之外，就是构成 Windows 环境的许多有趣的部件，如对话框，消息队列和呈现空间。

Windows 获得用户输入的方法有两种，即标准键盘与鼠标输入和对话框。对应地，Windows 处理与系统有关的输入、定时器消息和交互式应用程序输入。

键盘和鼠标输入是异步的。它是通过消息队列来处理的，通过消息队列，输入被转换成应用程序可以识别和处理的消息。一旦键盘或鼠标消息离开了系统输入队列，它们就由程序员设计的相应函数进行处理。为了让该函数处理消息，程序必须安装一个应用程序消息队列。窗口的每个实例必须有一个相应的应用程序消息队列。一般来说，应用程序消息队列的代码放在 WinMain 中。

通过对话框获得的用户输入也可以被系统输入队列处理，但它将被转到处理该对话框的函数。对话框窗口过程是一个内部再分类的过程，但它仅处理可由各种对话框控制解释的输入，如按钮、列表框和输入域等。再分类是一种将消息传送到不同过程的技术，它允许不相关的过程处理这些消息。当对话框过程成功地处理了一个消息时，系统就向应用程序消息队列发送一个结果消息，该消息可被用户窗口过程或框架窗口的缺省消息处理循环来处理。

1.5 输出机制

设备描述表(device context)就象是 Windows 的一块画布。在画布上，可利用窗口管理库和大多数图形设备接口(GDI)库中的函数来绘制文本和图形。当绘制窗口、对话框或屏幕上的其它控制时，Windows 也使用设备描述表。

在大多数情况下，一个设备描述表可以认为是一个给定设备(包括屏幕)的独立操作。通过设备描述表，可以用象素坐标来指定文本和图形对象，Windows 将象素坐标自动转换成系统设备的坐标。这种方法为许多程序员提供了一种新的思路，从而不要求程序员理解实际设备是如何处理应用程序的输出的。设备描述表的职责如下：

- □ 定义颜色属性；
- □ 提供缺省属性；
- □ 恢复剪裁区域；
- □ 转换象素坐标；
- □ 定义字体和逻辑符号；
- □ 处理用户定义的线型；
- □ 保存图形数据段。

设备描述表是基于系统所知的设备。它包括监视器的设备驱动程序。为了支持非标准设备，应用程序开发人员必须用 SDK 的设备驱动程序组件来创建一个驱动程序。

1.6 数据类型与数据结构

在利用 Windows 的头文件(包括其中所包含的函数定义)之前，您必须理解 Windows 采用的数据类型与结构。在大多数情况下，Windows 不使用标准 C 数据类型，而是用 C 语言的 define 和 typedef 语句创建了一个自己的数据类型集。

头文件 WINDOWS. H 中包含了大多数创建 Windows 数据类型的 define 和 typedef 语句，数据类型均以大写形式出现，数据类型中不使用下划线字符(_)，以此来区别于系统定义的常量。Windows 的数据类型可以分为五个组：

- ☐ 对应于标准 C 数据类型的数据类型；
- ☐ 用作句柄的数据类型。句柄是为间接引用各种对象所赋的内部值；
- ☐ 为适应 80286 的分段内存结构而使用远指针的数据类型。

当您熟悉前三类数据类型之后，就不难掌握第四类数据类型。因为 Windows 中的大多数数据类型都有一个相应的远指针数据类型。远指针包含段地址和偏移量，允许访问 80286 中的所有内存段。

1.7 标准数据类型

重新定义标准 C 的数据类型的做法是相当普遍的。Microsoft 为新数据类型所选择的名字尽可能接近于工业标准。例如，在多个环境中编过程的程序员掌握 Windows 的 WORD 数据类型的时间可能比标准 C 中的等价数据类型 unsiged short 要早。表 1—1 列出了经常使用的 Windows 数据类型。

表 1—1 常用数据类型

数据类型	位数	等价的 C 数据类型
BOOL	16	unsigned short
BYTE	8	unsigned integer
char	8	char
DWORD	32	unsigned integer 或段地址
int	16	signed integer
LONG	32	signed integer
short	16	signed integer
void	—	函数的空值
WORD	16	unsigned integer

1.8 命名约定

在编写 Windows 应用程序时，Microsoft 建议严格遵循命名约定。这些约定适应于结构、结构中的域、常量、消息参数和函数参量。

由于 Windows 头文件中的定义使用了这些约定，所以，无论您是否在代码中使用它们，这些名字都是很好理解的。

这里所采用的命名约定是由 Microsoft 程序员 Charles Simonoyi 创造的 Hungarian 约定。Hungarian 约定的主要目的是提高代码的可读性。下列规则总结了 Hungarian 约定的用法：

1. 所有参量、参数和域可由三个元素——前缀、基本类型和限定符组成；
2. 基本类型标识了变量的数据类型，基本类型中的所有字头均为小写。
3. 前缀部分给出了该基本类型的附加信息。例如，它说明变量是一个指针。前缀中的字母均为小写；
4. 限定符指出变量的用处。限定符由一个或多个字组成，每个字的首字母必须大写。

Hungarian 约定确定提高了代码的可读性，但也使名字更长。当然，您可以遇到忽略了变量前缀，且限定符很短的程序，但是，Windows 函数和结构中的许多域名使用了一些较长的名字。

无论如何，Hungarian 约定是非常有用的。例如，名字 bQueue 立即告诉我们，它是一个队列的真或假值；若在 bQueue 之后加上 Object 一词，即 bQueueObject，那么可以推知该队列是一个对象窗口。

表 1-2 基本类型的含义：

数据类型	含义
b	布尔变量。TRUE 表示活动状态，而 0 值表示 FALSE。
c	单字节字符值。
f	位标志(压缩成 16 位整数)
dw	带符号的 32 位值
h	句柄(16 位)
l	32 位长整数
lp	32 位长指针
n	SHORT 或 int(16 位)
p	16 位短指针
pt	压缩成 DWORD 的象素坐标。
rgb	红、绿、蓝颜色值
w	WORD 值(16 位)

前缀约定对于阅读 Windows 程序是有帮助的。常用的前缀有 id(标识符)和 a(数组)。表 1-3 列出了全部前缀的含义。

表 1-3　前缀的含义

前缀	含义
a	表示一个数组，后跟基本类型和可选限定符。例如 achCustomer 表示一个存放顾客名的数组。
c	表示一个统计量，以后可由基本类型构成一个变量名。通常限定符是不必要的。例如，cch 表示字符的统计值。
h	表示一个句柄，例如，hwnd 表示一个窗口句柄。
i	表示一个数组下标。
id	表示一个标识符。例如，idWindow 表示某窗口的 ID，而 idDialog 则表示某对话框的标识符。
np	表示一个近指针。例如，nps 表示一个指向 SHORT 值的近指针。
off	表示偏移量，用于表示缓冲区或结构的偏移量。
p	表示一个远指针。例如 pch 表示一个字符远指针。

1.9　常　量

Windows 中的常量之多可能会使您十分惊讶！Windows 的头文件中定义了 1000 多个常量。

和变量类似，常量严格遵从 Hungarian 约定。常量均由大写组成，它分为两个部分：前缀部分和限定符，前缀之后跟一个下划线，但前缀长度可以不同。最重要的是，Windows 的系统常量一般带有两或三个字符的前缀，但最多可达九个字符。典型地，前缀名是从相关函数、对象或过程中分离出来的，但是这种关联有时不太清楚；常量的限定符通常指出与函数、对象或过程相关的动作或状态。

常量定义为 16 或 32 位数。在函数调用中使用常量时，可以对它们进行 OR 运算，以产生不同的结果，但有时你可能会发现它等价于另一个常量值。另一种做法是将经常使用的一系列常量定义一个独立的头文件中，然后，再使用自己定做的标识符。

1.10　制作模板

本章中的例程可用于构造一个基本的 Windows 模板，所需做的工作是增加一个消息处理例程(即 WindowProcedure)。下面的程序例子提供了一个模板，这个模板可以运行本章中的例程。示例程序使用了一个资源描述，利用 Microsoft 的资源编译器，可将资源描述编译成可执行程序。

```
# TEMPLATE (Make File)
all: template.exe
```

```
template.res: template.rc template.h primer.ico
    rc -r template.rc

template.obj: template.c template.h
    cl -c -AS -Gsw -Oas -Zpe template.c

template.exe: template.obj template.def
    link /NOD template,,,libw slibcew, template.def
    rc template.res

template.exe: template.res
    rc template.res


; TEMPLATE.DEF (Module Definition File)

NAME            TEMPLATE
DESCRIPTION     'Sample template program'
EXETYPE         WINDOWS
STUB            'WINSTUB.EXE';
CODE            PRELOAD MOVEABLE DISCARDABLE
DATA            PRELOAD MOVEABLE MULTIPLE
HEAPSIZE        4096
STACKSIZE       9216

EXPORTS
                WindowProcedure @1
                About           @2


/* TEMPLATE.H  (Header file for template program) */

#define IDM_ABOUT 100

#define DEL_ID(x,y) ((x) & ~(y))
#define ADD_ID(x,y) ((x) | (y))

/* Functions in order of appearance */

int PASCAL WinMain(HANDLE, HANDLE, LPSTR, int);
long FAR PASCAL WindowProcedure(HWND, unsigned, WORD, LONG);
BOOL InitWin(HANDLE);
BOOL CreateODwindow(HANDLE, int);
BOOL FAR PASCAL About(HWND, unsigned, WORD, LONG);

DWORD dwStyle = WS_OVERLAPPEDWINDOW;
HANDLE hInst;

/* TEMPLATE.RC  (Resource script) */

#include "windows.h"
#include "template.h"

PrimerIcon ICON primer.ico
TemplateMenu MENU
BEGIN
        MENUITEM "&About", IDM_ABOUT
    END

AboutBox DIALOG 22, 17, 144, 75


STYLE DS_MODALFRAME | WS_CAPTION
CAPTION "About Template"
BEGIN
    CTEXT "Windows 3.0 Programming Primer"   -1, 0,  5, 144,  8
    CTEXT "Template Example Program"         -1, 0, 14, 144,  8
    CTEXT "(c) Alan Southerton"              -1, 0, 34, 144,  8
    DEFPUSHBUTTON "OK"                  IDOK, 53, 59, 32, 14, WS_GROUP
END

/* TEMPLATE.C  (Sample template program for use with routines) */

#include "windows.h"
#include "template.h"
```

```
int PASCAL WinMain(hInstance, hPrevInstance, lpCmdLine, nCmdShow)
HANDLE hInstance;                   /* current program instance */
HANDLE hPrevInstance;               /* previous program instance */
LPSTR lpCmdLine;                    /* command line string */
int nCmdShow;                       /* icon or window appearance */
{
     MSG msg;

     /* Call window registration routine */

     if(!hPrevInstance)
        if(!InitWin(hInstance))
            return(FALSE);

    /* Initialize new instance of a window */

     if(!CreateODwindow(hInstance, nCmdShow))
        return(FALSE);

    /* Message loop where Windows does it stuff */

    while (GetMessage(&msg, NULL, NULL, NULL))
    {
      TranslateMessage(&msg);
      DispatchMessage(&msg);
    }
    return(msg.wParam);
}


long FAR PASCAL WindowProcedure(hwnd, msg, wParam, lParam)
HWND hwnd;
unsigned msg;
WORD wParam;
LONG lParam;
{
    FARPROC lpProcAbout;


    switch(msg)
    {
        case WM_COMMAND:
            if(wParam==IDM_ABOUT)
            {
                lpProcAbout = MakeProcInstance(About, hInst);
                DialogBox(hInst, "AboutBox", hwnd, lpProcAbout);

                FreeProcInstance(lpProcAbout);
                break;
            }
            else
                return(DefWindowProc(hwnd, msg, wParam, lParam));

        case WM_DESTROY:
            PostQuitMessage(0);
            break;

        default:
            return(DefWindowProc(hwnd, msg, wParam, lParam));
    }
    return(NULL);
}
```