Windows 3.X
程序设计入门

邓 锐 编著

北京航空航天大学出版社

WINDOWS 3.X
程序设计入门

邓 锐 编著

北京航空航天大学出版社

内 容 提 要

本书是一本讲述 Windows 应用程序设计入门的书籍。书中用大量的实例，从几个方面详细地阐述了 Windows 3.X 版本程序设计的基本方法。

全书共分六章：第零章介绍建立工作环境；第一章讲述 Windows 应用程序设计的基本概念；第二、三、四章分别介绍了如何设计资源文件、Windows 的输入及窗口的控制；第五章介绍了图形设备接口(GDI)，并列举了一个例子——简易绘图系统。

书中列举的程序均能在 Windows 环境下运行，同时对程序清单中的关键语句都附有中文注释，对没有 Windows 应用程序设计经验的初学者尤有帮助；对已具有相当水平的 Windows 应用程序设计人员也有一定的参考价值。

图书在版编目(CIP)数据

WINDOWS 3.X 程序设计入门/邓锐编著. —北京：北京航空航天大学出版社，1995.12
ISBN 7-81012-590-7

Ⅰ.W… Ⅱ.邓… Ⅲ.窗口(软件)-程序设计-基本知识 Ⅳ.TP311

中国版本图书馆 CIP 数据核字(95)第 08850 号

- 书　　名：Windows 3. X 程序设计入门
  Windows 3. X CHENGXU SHEJI RUMEN
- 编 著 者：邓　锐
- 责任编辑：冯学民
- 原出版者：台湾全华科技图书股份有限公司
- 出 版 者：北京航空航天大学出版社
- 地　　址：北京市海淀区学院路 37 号 (100083，发行科电话 2015720)
- 印 刷 者：朝阳科普印刷厂
- 发　　行：新华书店总店科技发行所
- 经　　售：全国各地新华书店
- 开　　本：787×1092　1/16
- 印　　张：19.25
- 字　　数：492 千字
- 印　　数：5000 册
- 版　　次：1995 年 12 月第一版
- 印　　次：1995 年 12 月第 1 次印刷
- 书　　号：ISBN 7-81012-590-7/TP・174
- 版 权 号：图字：01-95-679 号
- 定　　价：26.00 元

序

自从 1990 年 5 月 Microsoft 发行 Windows3. X 版以来，Windows 的销售量呈直线上升，可见其魄力惊人。目前，市面上中文版的 Windows 程序设计的书籍及资料并不多。因此，作者将在研究所二年期间使用 Windows 所得到的一些浅见汇集在一起与读者共享。本书共分五章，第一章介绍 Windows 程序设计应具备的基本概念；第二章告诉你如何设计资源文件；第三章介绍输入装置；第四章介绍 Windows 提供的控制类别的用法；第五章介绍 GDI 的用法及制作——简易绘图系统。使用 Windows 作为程序设计工具对初学者的确显得杂乱无章，感到无从入手，但一旦熟悉了 Windows 程序设计的框架，你会发觉它是个功能强大的工具，使用它会节省不少时间。为了节省键入程序代码的时间，书后附有文中程序的磁盘。

本书得以顺利完成，首先要感谢恩师吴传嘉先生的教诲，感谢父母的栽培，对师兄林银河及诸位师弟洪振顺、郑文昌、张仁宇、林瑞安、傅韦睿对本书的校正及意见，在此一并表示感谢。

虽然相当仔细地编著这本书并力校再三，但是错误之处仍恐难免，尚祈读者不吝赐教，深表感激。

邓 锐 谨识于
国立台湾工业技术学院

编辑部序

系统编辑是我们的编辑方针，我们所提供给读者的绝不只是一本书，而是关于这门学问的所有知识，它们由浅入深，循序渐进。

本书是作者钻研 Windows 软件的心得，而非一般翻译手册。本书介绍该功能强大的软件给读者，以期帮助初学者或已有一定基础者早日成为 Windows 程序设计的能手。书末附有文中范例程序的磁盘，若能配合实际的上机操作，必可达到事半功倍的学习效果。

出版者注

我们仅将书单独出版。如读者需要范例程序的软盘，可向本社发行科魏兰英函购（邮编：100083；地址：北京市学院路 37 号；电话：2015720）。直接购买，每份 25 元；如需邮寄，另加邮购费 8 元。

试读结束，需要全本PDF请购买 www.ertongbook.com

目 录

第零章　工作环境的建立

在你尚未设计任何 Windows 应用程序之前，必须确定下列工具是否已安装在你的计算机内。这些必备之工具为：

(1) Microsoft C 5.1 以上版本；

(2) Windows 3.X 版本(中、英文版均可)；

(3) Windows SDK (开发工具)。

如果你已经安装了上述工具，但仍无法正常地编译 Windows 应用程序时，请检查系统环境参数的设定值。可以使用"set"命令检查系统环境参数。以下是本书在设计文中应用程序时的环境参数值(仅作参考用)：

```
PATH=C:\C600\BINB;C:\C600\BIN;C:\WIN;C:\WINDEV
LIB=C:\C600\LIB;C:\WINDEV\LIB
INCLUDE=C:\C600\INCLUDE;C:\WINDEV\INCLUDE
HELPFILES=C:\C600\HELP\*.HLP
INIT=C:\C600\INIT
TEMP=C:\WIN\TEMP
```

如果你发现环境参数值设定错误或少设了某些值时，可使用你最熟悉的编辑工具重新编辑 AUTOEXEC.BAT 的内容。修改完 AUTOEXEC.BAT 的内容后重新开机(在键盘上同时按下 Ctrl+Alt+Del 即可)。此时，系统的环境参数就设定为上次你修改的值。

在尚未探讨 Windows 应用程序设计之前，让我们先了解一个窗口(window)的组成元素。一个窗口是由下列元素组合而成：

(1) 系统菜单(System menu bar)

(2) 标题栏(Caption bar)

(3) 菜单条(Menu bar)

(4) 缩小按钮(Minimize box)

(5) 放大按钮(Maximize box)

(6) 垂直滚动条(Vertical scroll bar)

(7) 水平滚动条(Horizontal scroll bar)

(8) 窗口边界(window border)

(9) 窗口工作区(client area)

图 0-1 所示为各元素在窗口内的位置图。

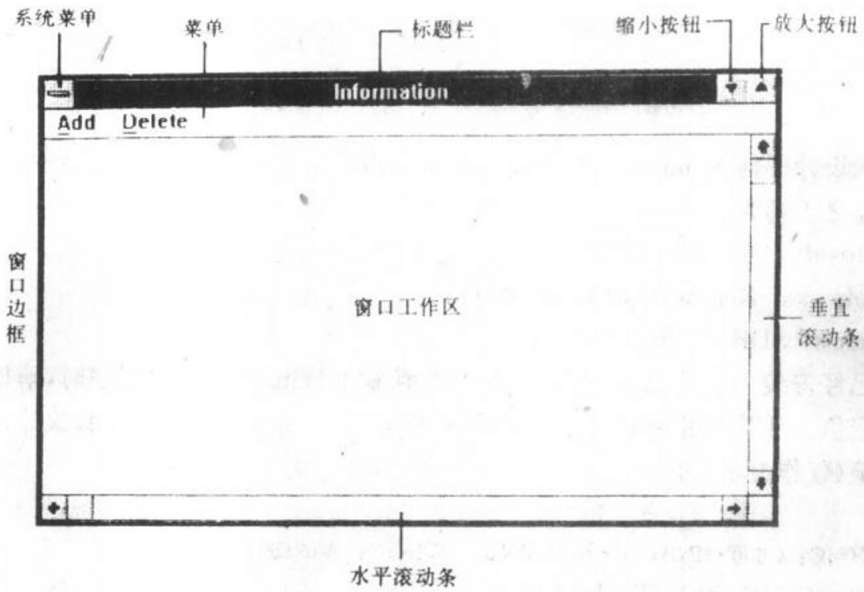

图 0-1 窗口组成元素

各个元素的功能说明如下：

(1) **系统菜单**：在这个菜单内允许

(a) 选择 Restore 命令可以将一个图标(ICON)还原至窗口原来大小；

(b) 选择 Move 命令可以任意移动窗口；

(c) 选择 Size 命令可以任意改变窗口的大小；

(d) 选择 Minimize 命令可以将窗口缩小成代表该窗口的图标；

(e) 选择 Maximize 命令可以将窗口放大成占满整个屏幕；

(f) 选择 Close 命令可以将窗口关闭——即关闭应用程序；

(g) 选择 Switch To 命令允许将控制权切换至不同的窗口；

(2)**标题栏**：除了显示应用程序的名称外，它具有另一个功能——移动窗口。

(3) **菜单条**：用户选择窗口命令(command)的区域。

(4) **缩小按钮**：将窗口缩小成代表该窗口的图标。

(5) **放大按钮**：将窗口放大到占满整个屏幕。

(6) **垂直滚动条**：允许窗口内容(文字或图象)向上或向下滚动。

(7) **水平滚动条**：允许窗口内容(文字或图象)向左或右滚动。

(8) **窗口边框**：可分为 WS_BORDER 及 WS_THICKFRAME 两种型式；对 WS_THICKFRAME 而言，用户可以改变窗口的宽度或长度。对于 WS_BORDER 而言，则不能改变窗口宽度或长度。

(9) **窗口工作区**：应用程序输出的区域。

窗口的上述特征在程序设计中均可由程序员自行控制。下一章我们将讨论如何控制这些特征。

第一章 Windows 应用程序设计的基本概念

80 年时代 DOS 的确是功能相当强大的操作系统(OS)。但由于近年来用户对个人计算机的性能要求愈来愈高,例如微处理器的速度要愈快愈好、没有内存的限制、多任务环境……。这些需要除去硬件因素外很多是 DOS 本身无法提供的,当然,未来的 DOS 版本可能会朝着这个方面努力。Windows 提供了许多 DOS 无法处理的功能,Windows 所提供的功能有:

(1) 一致性的图形用户接口(GUI);

(2) 它是个多任务系统(multitasking);

(3) 没有内存的限制;

(4) 提供独立性的绘图功能(device-independent graphics)。

了解了 Windows 所提供的功能后,我们将仔细地介绍设计 Windows 应用程序的步骤及注意事项。设计一个 Windows 应用程序可分为以下几个步骤:

(1) 向 Windows 注册一个或数个(依据程序需求而定)窗口类(Window class);

(2) 从已注册的窗口类(可以是应用程序中已注册的窗口类或 Windows 系统所提供的窗口类)中选择一个窗口类作为基类(base class)而创建一个窗口;

(3) 进入消息循环(message loop),等待一个消息发生;

(4) 设计应用程序中要使用的资源形式或种类:ICON、BITMAP、MENU、对话框(dialog)在 Windows 中均视为资源;

(5) 设计消息处理函数(callback function)。

如同一般 C 语言程序设计的 main()一样,Windows 应用程序亦有一个程序入口——WinMain()。请注意 W 与 M 是大写。WinMain()的主要目的是完成上述步骤中前三个步骤,故 WinMain()的格式如下所示:

```
int PASCAL WinMain(HANDLE, HANDLE, LPSTR, int)
   HANDLE   hInstance;
   HANDLE   hPreInstance;
   LPSTR    lpCmdLine;
   int      nCmdShow;
   {
      if(! hPreInstance)
      注册窗口类;
      创建窗口;
      进入消息循环;
   }
```

对于 WinMain()而言,应注意的事项有:

(1) PASCAL 保留字是指明当 Windows 系统调用 WinMain()时参数传递次序和

PASCAL 一样——传递次序由右至左，与一般 C 语言参数传递次序相反。传递过程均经由堆栈完成。Windows 规定任何被 Windows 系统调用的函数，它的前面要加上 PASCAL 保留字。

(2) 传递参数声明中，hInstance 是表示目前句柄(HANDLE)值，而 hPreInstance 表示前面同一个应用程序句柄值。如果这个应用程序执行两次以上，则对第一次执行应用程序而言，hPreInstance＝NULL，hInstance＝100(100 是为了说明而自己假设的；事实上，这个值是 Windows 系统设定的)；而对第二次执行应用程序而言，hPreInstance＝100，hInstance＝系统设定值。请注意在函数主体中有一个语句(statement)：

```
if(! hPrehInstance)     注册窗口类;
```

Windows 是多任务系统，它允许用户多次执行同一个应用程序，且每次执行时会指定一个唯一(unique)句柄值给 hInstance 以便于区别。因此，一个应用程序只要在第一次执行时注册了它所需要的窗口类，以后执行该相同应用程序时直接使用这个已注册过的窗口类就可以了。这里的 if 语句就是为了防止应用程序重复注册相同的窗口类。

(3) 参数 lpCmdLine 是个长指标，指向一个命令行。nCmdShow 是个整型数，它用来指明窗口显示形式。

1.1 Windows 的数据类型

在探讨 Windows 应用程序设计概念前，先了解一下 Windows 的数据类型。Windows 在表头文件 Windows.h 内定义了一些数据类型。本节将说明一些常用数据类型的意义，如表 1-1 所示。

表 1-1 常用数据类型的意义

数据类型	意　　义
BYTE	无符号(unsigned)8 位整数值
WORD	无符号 16 位整数值
DWORD	无符号 32 位整数值
LONG	有符号(signed)32 位整数值
HANDLE	无符号 16 位整数值，用来指明一个句柄
HWND	无符号 16 位整数值，用来指明一个窗口代码
LPSTR	指向字符串的远指针
PSTR	指向字符串的近指针
FARPROC	指向函数的长指针

RECT	设定矩形的左上角及右下角坐标值
	typedef struct tagRECT
	{
	int left; //矩形的左上角 x 坐标值
	int top; //矩形的左上角 y 坐标值
	int right; //矩形的右下角 x 坐标值
	int bottom; //矩形的右下角 y 坐标值
	}RECT;

表 1-1 (续)

数据类型	意 义
MSG	消息结构 typedef struct tagMSG { HWND Hwnd; WORD message; WORD wParam; LONG lParam; DWORD time; POINT Pt; }MSG; hwnd 是指接受消息的窗口代码,message 指明何种消息,wParam 及 lParam 依据 message 的形式有不同的意义,time 是指消息发生的时间,Pt 是指当消息发生时光标的 x,y 坐标值。

PAINTSTRUCT	指明窗口的工作区中哪一部分需要更新
	typedef struct tagPAINTSTRUCT
	{
	HDC hdc;
	//指明更新时的显示结构(display context)
	BOOL fErase; //指明更新时背景是否要重画
	RECT rcPaint; //指明更新的区域
	BOOL fRestore; //Windows 保留
	BOOL fIncUpdate; //Windows 保留
	BYTE rgbReserved[16]; //Windows 保留
	}PAINTSTRUCT;

以上所讨论是在设计 Windows 应用程序中常用的数据类型,但由于 Windows 所提供的数据类型极多,无法一一列举,因此,当我们在程序中遇到新的数据类型时,再说明该数据类型的意义。

1.2 如何向 Windows 注册一个新窗口类

Windows 提供了一个窗口类结构给程序员使用,使得应用程序内可以很容易地向 Windows 注册新窗口类。窗口类结构的定义如下所示:

```
typedef struct tagWNDCLASS{
    WORD        style;
    long        (FAR PASCAL *PfnWndProc)();
    int         cbClsExtra;
    int         cbWndExtra;
```

试读结束,需要全本PDF请购买 www.ertongbook.com

```
HANDLE          hInstance;
HICON           hIcon;
HCURSOR         hCursor;
HBRUSH          hbrBackground;
LPSTR           lpszMenuName;
LPSTR           lpszClassName;
}WNDCLASSS;
```

窗口类结构中各字段所代表的意义如下：

(1) style 指明窗口类的形式，它可以是下列常数的任意组合(使用 OR 运算符)：

(a) CS_DBLCLKS：送一个 double-click 消息到窗口内

(b) CS_HREDRAW：如果窗口的水平大小(即宽度)改变则更新整个窗口内容

(c) CS_VERDRAW：如果窗口的垂直大小(即高度)改变则更新整个窗口内容

(d) CS_NOCLOSE：系统菜单中 close 选择项被禁止使用(disable)

上述四个选择项是程序员所常用的，对其它没有探讨的选择项请参阅 Windows SDK 使用手册。

(2) (FAR PASCAL *lpfnWndProc)()是个指针函数，它指向一个函数且这个函数专门处理窗口类内所接收到的消息。我们称这个函数为消息处理函数(callback function)。

(3) cbClsExtra 指明在窗口类内是否要多分配一些空间。

(4) cbWndExtra 指明当创建窗口时是否多分配一些空间给窗口使用。

(5) hInstance 说明拥有此窗口类的句柄。

(6) HICON 是个图标代码的数据类型，对资源表(resource table)而言，它是 16 位索引值。参数 hIcon 指明当窗口缩小(minimize)时系统要使用哪一个图标。

(7) HCURSOR 是个光标代码的数据类型，对资源表格而言，它是 16 位索引值。参数 hCursor 指明在窗口内光标显示的形式。

(8) HBRUSH 是画刷(brush)代码的数据类型，对 GDI(Graphics Device Independence)实际绘图对象而言，它是 16 位索引值。参数 hbrBackground 指明采用哪种画刷做为窗口的背景颜色。

(9) lpszMenuName 指向一个菜单名称，这个菜单名称必须存在于资源文件(resource file)——即.RC 文件内。

(10) lpszClassName 指向一个表示窗口类名称的字符串。

了解了窗口类结构内各字段的意义后，我们要使用这个结构向 Windows 注册。Windows 提供了一个函数 RegisterClass(WNDCLASS * WC)，使用户很容易地向 Windows 注册一个新的窗口类。如果注册成功，函数返回 TRUE；否则，即注册失败，返回 FALSE。因此，在程序中可以依据 RegisterClass()传回的值来判别窗口类是否注册成功。现举一个实际例子如下：

```
if (! hPrevInstance)//防止应用程序向 windows 注册相同窗口类
  {
  wc.style=NULL; //不设定任何窗口形式
  wc.lpfmWndProc=InformationProc;//处理此窗口类内任何消息的函数
```

```
wc.cbClsExtra=0; //类内不保留额外空间
wc.cbWndWxtra=0; //窗口内不保留额外空间
wc.hInstance=hInstance; //拥有此类的句柄
wc.hIcon=LoadIcon(NULL,IDC_APPLICATION);//设定内设图标
wc.hCursor=LoadCursor(NULL,IDC_ARROW);//设定光标形式
wc.hbrBackground=GetStockObject(WHITE_BRUSH);//背景设定成白色
wc.lpszMenuName=NULL;//在.RC 文件内 Menu 名称,这个例子不使用 Menu,故声明为
                     NULL
wc.lpszClassName="InformationWclass";//窗口类名称
if (! RegisterClass(&wc)) //注册窗口类
  return FALSE;
}
```

if 语句防止重复向 Windows 注册相同的窗口类。函数 LoadCursor(HANDLE hInstance, LPSTR lpCursorName)将执行文件内 lpCursorName 光标资源加载到内存,且该执行文件的模块由 hInstance 指定;如果 hInstance=NULL,表示加载的光标资源是 Windows 提供的。Windows 所提供的光标资源有:IDC_ARROW(标准箭头光标)、IDC_CROSS(十字光标)、IDC_IBEAM(I 型文字光标)、IDC_ICON(空图标)、IDC_UPARROW(重直向上光标)、IDC_WAIT(漏斗型光标)……。函数 GetStockObject(int index)用来加载已存在于系统的画笔(pen)、画刷(brush)、字型(fonts)。我们使用这个函数加载白色画刷作为窗口类背景颜色。LoadIcon(HANDLE hInstance, LPSTR lpIconName)将执行文件 lpIconName 图标资源加载到内存且执行文件的模块由 hInstance 指定,如果 hInstance=NULL 时,表示加载的图标资源由 Windows 提供。Windows 提供的图标资源有:IDI_APPLICATION(空白图标)、IDI_ASTERISK(信息图标)、IDI_EXCLAMATION(惊叹号图标)、IDI_QUESTION(问号图标)、IDI_HAND(stop 图标)。

1.3 创建窗口

窗口类注册成功后,我们就可以使用它来创建窗口了。Windows 提供了一个函数 CreatWindow()允许用户创建窗口。CreatWindow()的声明情形如下:

```
HWND CreateWindow(LPSTR     lpClassName,
                  LPSTR     lpWindowsName,
                  DWORD     dwStyle,
                  int       X,
                  int       Y,
                  int       nWidth,
                  int       nHeight,
                  HWND      hWndParent,
                  HMENU     hMenu,
                  HANDLE    hInstance,
```

```
LPSTR        lpParam);
```

各个传递参数所代表的意义如下：

(1) 参数 lpClassName 说明窗口是继承哪一个已注册的窗口类名称。窗口类名称可以是由程序员自己定义的或是下列七个 Windows 提供的控制类(control class)：

(a) BUTTON

(b) COMBOBOX

(c) EDIT

(d) LISTBOX

(e) MDICLIENT

(f) SCROLLBAR

(g) STATIC

在第四章中将会详细介绍这些内设控制类的用法。

(2) 参数 lpWindowName 指向代表窗口名称的字符串。

(3) 参数 dwStyle 说明窗口以什么形式显示在屏幕上，一般常用的形式如表 1-2 所示。

表 1-2

窗口显示形式值	意　　义
WS_HSCROLL	创建一个具有水平滚动条的窗口
WS_VSCROLL	创建一个具有垂直滚动条的窗口
WS_MINIMIZEBOX	创建一个具有缩小按钮的窗口
WS_MAXIMIZEBOX	创建一个具有放大按钮的窗口
WS_CAPTION	创建一个具有标题栏的窗口
WS_BORDER	创建一个具有边框的窗口
WS_THICKFRAME	创建一个具有框架的窗口
WS_CHIDE	创建一个子窗口
WS_ICONIC	创建一个图标窗口
WS_MINIMIZE	创建一个最小尺寸的窗口
WS_MAXIMIZE	创建一个最大尺寸的窗口
WS_OVERLAPPED	创建一个具有标题栏及边框的窗口
WS_SYSMENU	创建一个具有系统菜单的窗口
WS_OVERLAPPEDWINDOW	创建一个具有标题栏、边框、系统菜单、放大按钮及缩小按钮的窗口
WS_POPUP	创建一个 pop-up 窗口

(4) 参数 X 对 overlapped 及 pop-up 窗口而言，是窗口左上角的 X 坐标值；对子窗口而言，是相对于父窗口工作区的窗口左上角 X 坐标值。如果 X 设定成 CW_USEDEFAULT 则 Windows 会自己设定 X 坐标值。

(5) 参数 Y 对 overlapped 及 pop-up 窗口而言，是窗口左上角的 Y 坐标值；对子窗口而言，是相对于父窗口工作区的窗口左上角 Y 坐标值。如果 Y 设定成 CW_USEDEFAULT 则 Windows 会自己设定 Y 坐标值。

(6) 参数 nWidth 指定窗口宽度值。如果 nWidth 设定成 CW_USEDEFAULT 则 Windows 会自己设定窗口宽度值。

(7) 参数 nHeight 指定窗口高度值。如果 nHeight 设定成 CW _ USEDEFUAT 则 Windows 会自己设定窗口高度值。

(8) 参数 hWndParent 说明这个新创建窗口是否有父窗口。对 WS _ OVERLAPPED 窗口而言,hWndParent 设成 NULL,因为 WS _ OVERLAPPED 窗口没有父窗口。对 WS _ CHILD 窗口而言,hWndParent 就必须设定成拥有这个子窗口的父窗口的窗口代码。

(9) 参数 hMenu 说明这个新窗口要使用哪一个菜单或是代表子窗口的常数——即 hMenu 值有三种不同意义:

(a) hMenu=NULL 时,新窗口使用窗口类内定义的菜单。

(b) hMenu 设定成一数据类型为 HMENU 的变量时,新窗口使用自己定义的菜单,不使用窗口类的菜单。

(c) hMenu 设定成一常数时,新窗口必为子窗口且使用这个常数来和其它子窗口相区分的(因为父窗口可以同时拥有许多子窗口)。

(10) 参数 hInstance 说明拥有新窗口句柄实例(instance)。

(11) 参数 lpParam 只在 MDICLIENT 类时才有意义。对其它窗口类无意义,一般均设成 NULL。

了解了 CreateWindow()内各参数的意义及用法后,下面举一个实际例子:

```
xScreen=GetSystemMetrics (SM _ CXSCREEN);//取得屏幕的宽度
yScreen=GetSystemMetrics(SM _ CYSCREEN);//取得屏幕的长度
InfoWnd =CreateWindow(        //创建窗口
        "InformationWclass",//类名称
        "Information",//窗口标题名称
        WS _ OVERLAPPEDWINDOW|WS _ HSCROLL
        |WS _ VSCROLL //设定窗口形式
        0,                  //窗口左上角 X 坐标值
        0,                  //窗口左上角 Y 坐标值
        xScreen, //窗口宽度
        yScreen, //窗口长度
        NULL,//对 WS _ OVERLAPPEDWINDOW 窗口形式而言没有父窗口
        NULL,//使用类内所设的 Menu
        hInstance,//拥有此窗口的句柄
        NULL //这个字段只在 MDICLIENT 类时才使用
        );
    if (! InfoWnd)
      return (FALSE);
```

函数 GetSystemMetric(int index)可用来轻易地取得 Windows 各显示元素,例如图标、水平及垂直滚动条、缩小按钮、放大按钮等的宽度及高度。在这个例子里,我们使用 SM _ CXSCREEN 及 SM _ CYSCREEN 索引值取得屏幕的宽度及高度。我们使用已注册的窗口类 InformationWclass 当做基类来创建窗口。如果新窗口创建成功,则返回一个窗口代码;否则,即创建失败,CreatWindow()返回 NULL。程序中可以依据返回值判断窗口是否创建成功。

即使成功地创建了新窗口，但屏幕上仍未显示这个新窗口，为什么呢？CreateWindow()只负责帮创建新窗口，它并没有将窗口显示在屏幕上。因此，必须自己将窗口显示在屏幕上。Windows 提供了一个函数 ShowWindow(HWND hWnd, int nCmdShow)可随心所欲地选择窗口以什么形式显示在屏幕上。nCmdShow 控制显示类型，设定的值如表 1-3 所示。

表 1-3

nCmdShow 值	意　　义
SW _ HIDE	将窗口隐藏并将控制权交给其它窗口
SW _ MINIMIZE	将窗口缩小成图标并将控制权交给窗口管理队列中最上层窗口
SW _ SHOW	将窗口显示在当前位置上并拥有控制权
SW _ SHOWMINIMIZED	窗口以图标形式显示在屏幕上并拥有控制权
SW _ SHOWMAXIMIZED	窗口以最大(maximize)形式显示在屏幕上并拥有控制权
SW _ SHOWNORMAL	显示窗口并拥有控制权

因此，必须在窗口创建成功之后加上下面两条语句。

```
ShowWindow(InfoWnd,nCmdShow);// 将窗口显示在屏幕上
UpdateWindow(InfoWnd);//送 WM _ PAINT 消息给窗口 Infownd
```

此时，nCmdShow 值由 Windows 决定。一般，Windows 设定为 SW _ SHOWNORMAL。函数 updateWindow(HWDN hWnd)会不经过应用队列(application queue)送出一个 WM _ PAINT 消息给 hWnd 窗口。

1.4 进入消息循环

消息循环(message loop)主要功能是从应用队列中取得属于自己的消息，并将消息的虚拟代码转换成相对的字符代码，并将其送到窗口内。以下是一个实际例子：

```
//进入消息循环
//当 GetMessage()接收到 WM _ QUIT 消息时，while 循环结束
while(GetMessage(&msg,  //从应用程序队列内取出消息
                 NULL,
                 NULL,
                 NULL))
{
TranslateMessage(&msg);//将消息的虚拟代码转换成相应的字符代码
DispatchMessage(&msg);//将消息传送给窗口
}
```

消息循环是由一个 while 循环构成的。函数 GetMessage(LPMSG lpMsg, HWND hWnd, WORD wMsgfilterMin, WORD wMsgFilterMax)负责从应用队列接收消息，它的返回值(TURE 或 FALSE)决定是否结束应用程序的执行，当 GetMessage()接收到 WM _ QUIT 消

息时，它返回 FALUSE；对于其它非 WM _ QUIT 的消息，它返回 TURE。如果应用队列内没有属于自己的消息，那么 GetMessage()会将控制权交给其它应用程序使用，直到应用队列有属于自己的消息时才又取得控制权。参数 nMsgFilterMin、nMsgFilterMax 分别确定 hWnd 所能接收到消息的下限与上限，因为消息是按整数形式依次增加的。在这个例子中 hWnd、nMsgFilterMin、nMsgFilterMax 均设成 NULL，表示从应用队列中取得所有消息。函数 TranslateMessage()负责将消息的虚拟代码转换成相对应的字符代码，例如对 WM _ KEYDOWN/WM _ KEYUP 的组合转换成 WM _ CHAR/WM _ DECHAR 消息。函数 DispatchMessage()负责将接收的消息传送给窗口。

我们将 1.2 节、1.3 节和本节中三个实际例子组合起来就构成 WinMain()内的主要功能，将来读者会发现任何 Windows 应用程序的 WinMain()都使用这种框架。故将 WinMain()的框架及例子加以整理以加深读者的印象。

```
int PASCAL WinMain (HANDLE,HANDLE,LPSTR,int)
   HANDLE     hPreInstance;
   HANDLE     hInstance;
   LPSTRR     lpCmdLine;
   int        nCmdShow;
   {
   if (! hPreInstance)
     注册窗口类;
     创建窗口;
     进入消息循环;
   }
   HANDLE hImst;
   int PASCAL WinMain(hInstance,hPrevInstance,lpCmdLine,nCmdShow)
      HANDLE hInsstance; //当前句柄
      HANDLE hPrevInstance;//以前的句柄
      LPSTR lpCmdLine;//命令行
      int nCmdShow;//窗口显示形式
      {
      MSG msg; //声明消息变量
      WNDCLASS wc; //声明窗口类结构变量
      int xScreen,yScreen;
      if (! hPrevInstance) //防止应用程序向 windows 注册相同窗口类
        {
        wc. style=NULL;//不设定任何窗口形式
        //处理此窗口类内任何消息的函数
        wc. lpfmWmdProc=InformationProc;
        wc. cbClsExtra=0;//类内不保留额外空间
        wc. cbWmdExtra=0;//窗口内不保留额外空间
        wc. hImstance=hInstance;//拥有此类的句柄
        wc. hIcon=LoadIcon(NULL,IDI  APPLICATION);//设定内设图标
```

试读结束，需要全本PDF请购买 www.ertongbook.com