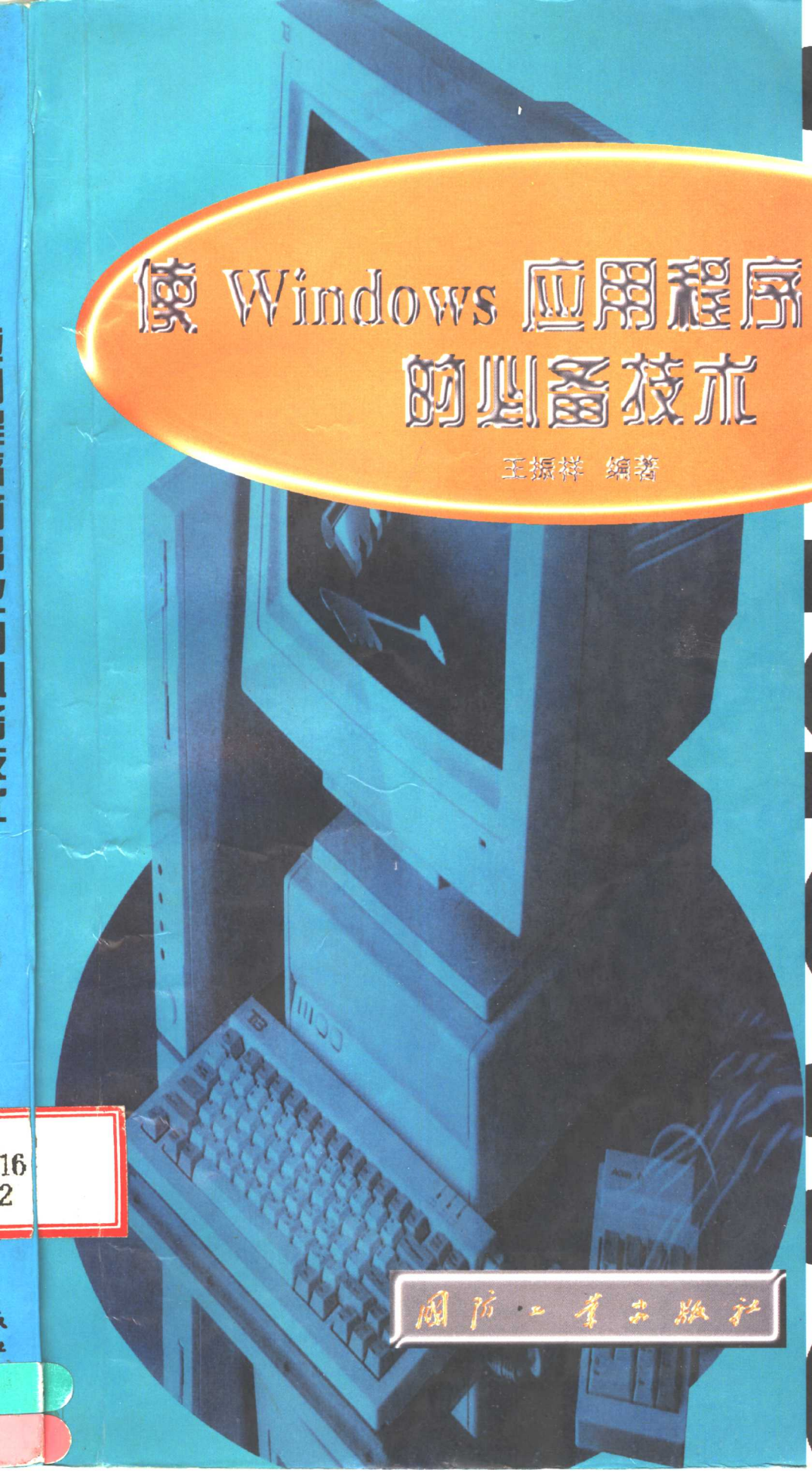
使 Windows 应用程序商品化的必备技术

王振祥 编著



国防工业出版社

使 Windows 应用程序商品化的必备技术

王振祥　编著

国防工业出版社

图书在版编目(CIP)数据

使 Windows 应用程序商品化的必备技术/王振祥编著. —
北京:国防工业出版社,1997.1
ISBN 7-118-01571-7

Ⅰ.使… Ⅱ.王… Ⅲ.应用程序-窗口(软件)-程序设计
Ⅳ.TP319

中国版本图书馆 CIP 数据核字(95)第 23676 号

国防工业出版社出版发行
(北京市海淀区紫竹院南路 23 号)
(邮政编码 100044)
北京怀柔新华印刷厂印刷
新华书店经售
*
开本 787×1092 1/16 印张 27¼ 630 千字
1997 年 1 月第 1 版 1997 年 1 月北京第 1 次印刷
印数:1—4000 册 定价:33.70 元

前　言

目前，Windows 在计算机领域的重要性不言而喻。有人说，掌握了 Windows，就不愁找不到工作，此言确有其道理。在此情况下，学习和研究 Windows 以及为 Windows 开发应用程序具有明显的价值。

目前关于 Windows 的书籍很多，但是细心的读者会发现，这些书籍均讲述一些 Windows 的基本的编程技术。阅读这些书籍可以掌握 Windows 的基本编程技术，从而可以实现应用程序的基本功能，但是实现了基本功能的应用程序只能算作"试验室产品"，要成为真正的产品，尚有大量的工作要做。诸如安装、演示、培训、反跟踪、开发 Control Panel 应用程序实现配置等。

本书以一个 Windows 应用程序 ONSELL 为例，全面讲述使 Windows 应用程序从"试验室产品"变成真正的商品所需要进行的工作及相应的编程技术。第一章讲述一些预备知识；第二章讲述准备商品化的 Windows 应用程序 ONSELL 的功能及实现；第三章讲述 ONSELL 的安装技术；第四章讲述如何在 Progman Manager 中为 ONSELL 增加程序组及相应的程序图标；第五章讲述开发 Control Panel 应用程序实现对 ONSELL 的配置；第六章讲述编写演示程序的技术；第七章讲述教程程序的编程技术；第八章讲述几种反跟踪技术；第九章讲述建立帮助系统的三种不同方法；第十章讲述介绍开发人员的标准方法及编程技术；第十一章讲述两种进一步完善应用程序的技术，第一种技术用于开发一个完善的 STUB 程序，第二种技术则用于使程序同时支持 DOS 和 Windows；第十二章讲述如何控制对话框，建立丰富多彩的帮助系统。

本书是第一本讲述使 Windows 程序商品化的技术书籍，也是一本深入了解 Windows 系统、学习 Windows 编程技巧的书籍。本书涉及到许多 Windows 特殊的编程技术及许多普通 Windows 书籍很少涉及的 Windows 的概念，这些特殊的编程技术（如通知函数等）及概念（如钩子、超类等）在 Windows 中不常用，但却非常重要。

本书对所讲述的内容提供了示范程序，这些程序均在 Borland C++ V3.1 下调试通过。在本书的许多示范程序中出现了长字符串。为了排版方便，将长字符串分为若干行，但在实际使用时，应当作为一行出现。本书的内容集技术性、技巧性和实用性于一体。阅读本书，不但可以全面掌握使 Windows 应用程序成为商品所涉及的各种技术，也可学习到许多 Windows 的特殊编程技术，另外还可对 Windows 更深入地了解。

在本书的编写过程中，本书的责任编辑陈子玉老师付出了大量的劳动，在此表示衷心的感谢。

能向读者奉献这样一本集技术、技巧与实用于一体的书籍，作者感到很高兴，但是由于水平所限，错误和不当之处在所难免，希望读者不吝指正。

王振祥

内容简介

本书是第一本讲述使 Windows 程序商品化的技术书籍。

本书以一个 Windows 应用程序 ONSELL 为例，全面讲述了使 Windows 应用程序从“试验室产品”变成真正的商品所需要进行的工作及相应的编程技术，如程序的安装、演示、教程、反跟踪、在 Program Manager 中增加图标、在 Control Panel 中增加应用程序项等。

本书是一本深入了解 Windows 系统、学习 Windows 编程技巧的书籍。

本书涉及到许多 Windows 的特殊的编程技术及许多普通 Windows 书籍很少涉及的 Windows 的概念，这些特殊的编程技术（如通知函数等）及概念（如钩子、超类等）在 Windows 中不常用但却非常重要。

本书适用于 Windows 的开发人员及希望对 Windows 进一步了解的人员学习和使用，也可作为大中专学生、研究生及一切对 Windows 感兴趣者的参考资料。

目　录

第一章　预备知识

1.1　引　　言

本章集中讲述一些本书在以后章节中经常使用的预备知识，比如调试动态连接库的方法、运行调试程序时屏幕出现混乱时的解决方法、使用其它程序的位图、光标、图标资源的方法等；本章将还介绍 Windows 的子类与超类；还列举了一些禁止作为程序名字使用的字符串。

1.2　Windows 的静态连接库、动态连接库及引入库

Windows 的库实际上可分为三种库：静态连接库、动态连接库和引入库。它们有着不同的功能和作用。静态连接库提供了函数的完整的目标代码，如果程序调用静态连接库中的函数，则在进行连接时连接程序将静态连接库中所包含的该函数的代码拷贝至运行文件中（比如 Borland C＋＋的 CWS. LIB 就属于静态连接库，诸如 chdir 等函数就包括在 CWS. LIB 中）。动态连接库也包含了其所提供的函数的目标代码，但是在程序调用动态连接库中的函数时，连接程序并不将包含在动态连接库中的该函数的目标代码拷贝至运行文件，而只是简单地记录了该函数的位置信息（即包含于哪个动态连接库中以及在动态连接库中的位置）。只要记录了这些信息，程序在执行时，即可得到函数的目标代码，因为只有在执行时才得到真正的连接，因此才称为动态连接。而提供函数在动态连接库中位置的信息存放在一个独立的文件中，这个文件就是引入库。也就是说，引入库提供了程序与动态连接库中的连接信息。

图 1－1 以应用程序调用函数 CreateWindow 为例，示意说明了应用程序、动态连接库、引入库以及运行文件的关系。

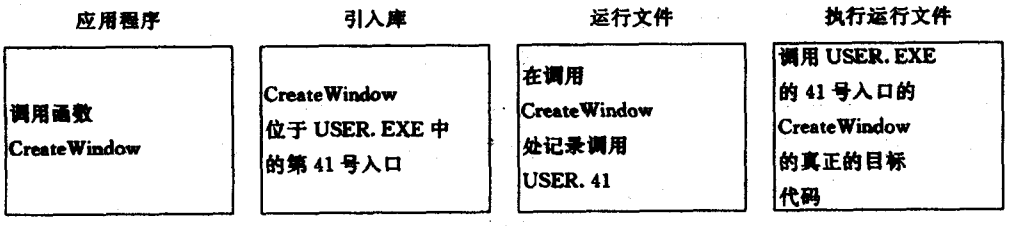


图 1－1　应用程序、动态连接库、引入库、运行文件的关系

综前所述，引入库的作用是实现动态连接，引入库中记录了函数在动态连接库中的位置，应用程序在连接时与引入库进行连接，从而也就与动态连接库实现了连接，引入库中包含的函数均是动态连接库中的引出函数。

1.3 使用未公开的 Windows 函数的三种方法

Windows 为程序员提供了许多函数，但是也有许多函数未被 Windows 公开，而这些未公开的函数在某些情况下有特殊的用处。本节讲述了使用未公开的 Windows 函数的三种方法。

Windows 许多动态连接库（如 USER.EXE、KRNL386.EXE、GDI.EXE 等）均有许多未公开的函数。这些未公开的函数虽然未被公开，但是这些函数实际上已经存在于 Windows 所提供的动态连接库中。但是对于不同的编译器，其所提供的引入库是不相同的，有的编译器将未公开的函数列在其引入库中（比如 Borland C++所提供的引入库 IMPORT.LIB 就是如此），而有的编译器则将这些未公开的函数强行从引入库中剔除（比如 Microsoft 的 SDK 所提供的引入库 LIBW.LIB 就是如此）。

对于未公开的函数，有以下三种不同的使用方法。

一、生成并连接引入库

为了从动态连接库生成引入库，几乎所有的编译器均提供了相应的应用程序（如 Borland C++的应用程序 IMPLIB），这种程序可以根据 DLL、EXE 甚至 DEF 文件生成引入库。Microsoft 提供的完成同样功能的应用程序的名字也为 IMPLIB。

下面的命令示例用于从 GDI.EXE 文件（扩展名虽然为 EXE，实际上是动态连接库）产生其相应的引入库 GDI.LIB：

```
IMPLIB GDI.LIB GDI.EXE
```

只要有了引入库，在连接时与引入库直接进行连接即可。

二、通过 Windows 的特定函数直接使用未公开的函数

Windows 提供了直接调用动态连接库中的函数的函数，这就是 LoadLibrary（或 GetModuleHandle）和 GetProcAddress。其中 LoadLibrary 用于取得动态连接库的句柄，如果动态连接库尚未装入内存，则将其装入，如果已经装入内存，则将其使用计数器加 1。而函数 GetModuleHandle 则用于取得一个模块的句柄，所谓模块是指动态连接库、应用程序等所有 NE 文件的内存版本的统称，也就是说，函数 GetModuleHandle 只能用于取得已经装入内存的模块的句柄。在取得模块句柄之后，可以用 GetProcAddress 得到未公开的函数的地址，然后即可实现函数的调用。

关于这几个函数的使用方法，可参看相关资料。

三、在程序中直接使用未公开的函数

使用本方法需要进行以下三种工作。

1. 在 DEF 文件中用 IMPORTS 将调用的函数名显示地引入。

比如：需要在程序 SAMPLE 中使用 USER. EXE 中的未公开的函数 DragDetect，则需要在文件 SAMPLE. DEF 中使用如下语句：

```
IMPORTS
        USER. DragDetect
```

在 USER. DragDetect 中同时提供了动态连接库的名字(USER)及函数的名字(DragDetect)。当然也可以不使用函数名，而直接使用其入口号。比如下面的语句可实现同样的功能：

```
IMPORTS
        USER. 465
```

说明 465 为函数 DragDetect 在 USER 中的入口号。

(2)在 C 文件(或 H 文件)中声明未公开函数的原型。

仍以使用 USER 中的 DragDetect 为例，说明该函数原型为 BOOL FAR PASCAL DragDetect(HWND,LPPOINT)，因此需要在 C 文件(或 H 文件)中作如下声明：

```
BOOL FAR PASCAL DragDetect(HWND,LPPOINT);
```

(3)在 C 程序中直接使用该函数。

仍以 SAMPLE 为例。比如在 SAMPLE. C 中，可直接使用如下语句：

```
...
DragDetect();
...
```

下面对这三种方法进行比较，其中第一种方法需要建立新的引入库(因为 Windows 已经提供了相应的引入库，实际上必须对 Windows 所提供的引入库进行更新)，另外还必须在连接时指定库名，这给程序员带来了麻烦；第二种方法需要调用三个不同的函数，并且在调用时需要对这三个不同函数的返回值进行检查，从而增加了程序量；而第三种方法与其它两种方法相比较具有直观、方便的优点。

本书中在使用未公开的函数时，均使用第三种方法。

四、使用未公开函数的示范程序

本节以实例示范了前述使用未公开函数的第二种和第三种用法。在例子程序中使用的是第二种方法(用于第三种方法的语句作为注释出现)。如果使用第三种方法，可将用于第二种方法的语句作为注释，而将现程序中用于第三种方法的语句的注释清除。

例子程序包括 UNDOC. C 和 UNDOC. DEF 两个文件。

(1)文件 UNDOC.C

```
//
//  使用未公开的函数的示例程序
//
#define         STRICT
#include        <windows.h>
#include        <stdlib.h>
#include        <string.h>
#include        "Undoc.h"
int     PASCAL      WinMain(HINSTANCE,HINSTANCE,LPSTR,int);
long    CALLBACK    UndocMainWndProc(HWND,int,WPARAM,LPARAM);
// 使用第三种方法时,需要使用下面的一行语句。
// void FAR PASCAL   FillWindow(HWND,HWND,HDC,HBRUSH);
// 使用第二种方法时,需要使用下面的一行语句。
void (FAR PASCAL * FillWindow)(HWND,HWND,HDC,HBRUSH);

HINSTANCE     hInst;
HWND          hWnd;

BOOL    InitApplication(HINSTANCE hInstance)
{
WNDCLASS      wc;

      wc.style      = CS_HREDRAW|CS_VREDRAW;
      wc.hCursor    = LoadCursor(NULL,IDC_CROSS);
      wc.cbWndExtra = 0;
      wc.cbClsExtra = 0;
      wc.hbrBackground = GetStockObject(GRAY_BRUSH);
      wc.lpszMenuName = NULL;
      wc.lpszClassName = "UndocClass";
      wc.lpfnWndProc = (WNDPROC)UndocMainWndProc;
      wc.hInstance = hInstance;

      if(RegisterClass(&wc))
            return TRUE;
      else
            return FALSE;
}
```

```
BOOL    InitInstance(HINSTANCE hInstance,int nCmdShow)
{
    hWnd = CreateWindow (
                        "UndocClass",
                        "Call Windows Undoc Function Sample",
                        WS_OVERLAPPEDWINDOW,
                        CW_USEDEFAULT,
                        CW_USEDEFAULT,
                        CW_USEDEFAULT,
                        CW_USEDEFAULT,
                        NULL,
                        NULL,
                        hInstance,
                        NULL);
    if(! hWnd)
        return  FALSE;
    UpdateWindow(hWnd);
    ShowWindow(hWnd,nCmdShow);
    return(TRUE);
}

int    PASCAL  WinMain (HINSTANCE hInstance,HINSTANCE hPrevInstance,
                        LPSTR lpCmdLine,int nCmdShow)
{
MSG     msg;
WORD    UserHandle;
        // 使用第二种方法时,需要使用下面的两行语句,这两行语句首先取得模块
        // USER 的句柄,然后取得函数 FillWindow 的地址。
        UserHandle = GetModuleHandle("USER");
        FillWindow= GetProcAddress(UserHandle,"FILLWINDOW");

        hInst = hInstance;
        if(! hPrevInstance)
            InitApplication(hInstance);

        InitInstance(hInstance,nCmdShow);

        while(GetMessage(&msg,NULL,NULL,NULL))
        {
```

```
            TranslateMessage(&msg);
            DispatchMessage(&msg);
        }
        return(msg.wParam);
}

long    CALLBACK    UndocMainWndProc (HWND hWnd,int message,
                                      WPARAM wParam,LPARAM lParam)
{
HDC     hDC;

        switch(message)
         {
            case    WM_CREATE:
            break;

            case    WM_CHAR:
            // 按下一字符键时,将窗口背景清除为白色。
            hDC = GetDC(hWnd);
            FillWindow(NULL,hWnd,hDC,GetStockObject
                                    (WHITE_BRUSH));
            ReleaseDC(hWnd,hDC);
            break;

            case    WM_DESTROY:
            PostQuitMessage(0);
            break;

            default:
            return(DefWindowProc(hWnd,message,wParam,lParam));
         }
       return(0);
}
```

(2)文件 UNDOC.DEF

```
NAME            UNDOC
DESCRIPTION     'Windows Undocument function'
EXETYPE         WINDOWS
STUB            'WINSTUB.EXE'
DATA            PRELOAD MOVEABLE MULTIPLE
```

```
CODE            PRELOAD MOVEABLE DISCARDABLE
HEAPSIZE        8192
STACKSIZE       5120

;  使用下面第三种方法需要使用下面的两行语句。
;IMPORTS
;               USER.FILLWINDOW

EXPORTS
                UndocMainWndProc
```

1.4 如何开发动态连接库

一、开发动态连接库的基本知识

在大多数情况下，编写的应用程序均为普通的 Windows 可执行程序，但是在某些情况下，可能不得不编写动态连接库（比如 Windows 要求系统级钩子函数必须位于动态连接库中）。动态连接库主要为应用程序提供函数，因此动态连接库主要提供引出函数，即凡是提供给应用程序使用的函数均必须引出。动态连接库与应用程序有以下不同：

（1）其主过程为 LibMain，而非 WinMain。

（2）应当为动态连接库提供 WEP 函数（虽然该函数并不是必需的）。

（3）其 DEF 文件必须使用 LIBRARY，而非 NAME。

（4）其提供应用程序使用的函数必须为引出的。

Windows 引出函数的方法有以下两种：

（1）在 DEF 文件中将其列于 EXPORTS 中。

（2）在 C 文件中对需要引出的函数使用__export。比如示范程序 DLL.C 中可将需要引出的函数 Beep 作如下声明：

```
int  __export FAR PASCAL Beep(void)
```

本节的示范程序是一个极其简单的动态连接库程序 DLL.C。该动态连接库只有以下三个函数：LibMain、WEP 和 Beep，其中 Beep 是引出的，该函数由示范程序中的应用程序 CALLDLL 调用。

使用用户提供的动态连接库的方法与使用未公开的 Windows 函数的方法相同，即前面所述的三种使用未公开函数的方法与使用用户自己开发的动态连接库同样有效，因为从实质上说，Windows 所提供的函数均位于其动态连接库中，与读者自己开发的动态连接库所不同的是，Windows 为这些内建的动态连接库中已公开的函数提供了必要的引用信息（如函数的原型均声明于 Windows.h 文件中等），因此对这些公开的函数可直接使用，而对于那些未公开的函数则未提供这些必要的信息。也就是说，Windows 自身的未公开的函数的地位与读者自己开发的动态连接库中的函数的地位完全相同。因此使用读者自己开发的动态连接库中的函数的方法与使用未公开的 Windows 函数的方法是完全相