Windows 编程简明教程

刘振安　编著

安徽科学技术出版社

WINDOWS 编程简明教程

刘振安 编著

安徽科学技术出版社

（皖）新登字 02 号

责任编辑：胡正义
封面设计：王国亮

内 容 提 要

本书采用多文件方式编程，并通过程序的变换说明它们的特点。不仅介绍最基础的概念，而且选用实例进一步说明问题，把涉及到的函数之用法也介绍清楚。尤其是通过程序的变化，揭示编程及消息处理原理，以便读者举一反三，触类旁通。所举程序均调试通过，源程序及运行文件均存储在磁盘上以方便教学与学习。

本书可作为大专院校研究生、高年级学生选修课及各种培训班的教材，尤其适合广大工程技术人员作为自学 Windows 编程的读本。

Windows 编程简明教程
刘振安 编著

*

安徽科学技术出版社出版
（合肥市九州大厦八楼）
邮政编码：230063
安徽省新华书店经销 铁四局印刷厂印刷

*

开本：787×1092 1/16 印张：15 字数：37 万
1995 年 7 月第 1 版 1995 年 7 月第 1 次印刷
印数：8 000
ISBN7-5337-1210-2/TP·20 定价：13.00 元
本书配有 5 寸盘一张 单价：15.00 元

前　言

Windows 正风靡世界，众多的计算机工作者及用户都被它漂亮的外表所折服。它是当今最受欢迎的软件开发环境之一，许多著名的软件公司都在 Windows 开发环境下开发软件。有些软件公司为了适应 Windows 的发展，都推出在 Windows 环境下运行的软件版本。有人说"只要具有 Windows 编程经验，就不愁找不到工作"，可见 Windows 流行之广，程序设计人员必须熟悉和掌握 Windows 环境下新的程序设计思想和方法，以适应软件发展潮流的需要。

倘若仅以一个简单的 Windows 应用程序为例，介绍 Windows 应用程序设计的一些基本概念，说明 Windows 应用程序的主要组成部分，则只能使读者对 Windows 应用程序有初步的了解。如果只是罗列一大堆详尽资料，又会使人陷入茫茫大海。程序设计教学最好的方法是仔细地选择和描述标准的例子，最初的学习应建立在模仿上以加深理解。想用一个例子就能引导读者熟悉 Windows 应用程序的开发是不可能的。这里把它分类，由简入繁，以实用为前题，逐步深入到各个部分，并通过程序的变换说明它们的特点。

本书采用多文件方式编程，这既能把 Windows 的变化更详细地展现出来，又能使每章的程序有大部分公用文件；既能熟悉大程序的编制方法，又能深入理解多个文件共用变量的实现方法；既能向读者提供很多实用程序，又能节省篇幅。

本书目的是尽快帮助读者入门，所以主要讲授基础知识。所谓"简明"，就是不仅介绍最基础的概念，而且选用实例进一步说明问题，把涉及到的函数用法也介绍清楚。尤其是通过程序的变化，揭示编程及消息处理原理，以便读者举一反三，触类旁通。

所举程序均调试通过并整理在磁盘上，以期省去读者输入程序之苦，把精力集中于理解 Windows 应用程序的构成方法及其工作原理上，以尽快迈进 Windows 大门。程序清单见附录一，需要的读者可以与出版社联系。

本书共分七章。第一章是 Windows 程序的基本代码结构，通过 C 语言多文件编程与 Windows 典型多文件程序的比较，不涉及程序语句作用的前题下分析几个典型 Windows 程序的结构，阐述 Windows 应用程序开发的特点、开发过程及编程方法；第二章是画笔、画刷和字体，通过分析例题，解释了 Windows 编程基础，说明 Windows 应用程序各个部分的基本构成；第三章是菜单，给出了应用广泛的菜单设计与处理原理；第四章是对话框，给出了 Windows 下的对话及处理方法；第五章是资源与加速键，在综述各种资源的利用方法及介绍了菜单与加速键配合方法的同时，改变了应用程序的文件结构，引入新的变化；第六章是计时器，给出新的窗口形式，它与第五章一样，均以新的面貌出现以揭示 Windows 程序的内核及多文件之间变量互相引用的问题；第七章是键盘与鼠标器，目的是介绍利用键盘与鼠标器进行人机对话的基本处理方法。

由于本人才疏学浅，错误在所难免，敬请读者批评指正。

刘振安

1995 年于中国科学技术大学(合肥)

目　录

第一章　Windows 程序的基本代码结构

本章将通过实例说明 Windows 的特点，不要把熟悉 Windows 函数作为学习编程的重点，应把学习基本的代码结构和程序设计方法作为重点。对大多数程序员来说，学习 Windows 程序设计的最大障碍在于必须打破传统的程序设计方法并全心全意地接受 Windows 程序设计思想。如果以一种开放的思维去学习 Windows 编程，就会很快地适应这个新的程序设计的概念和方法。

Windows 推荐用 C 语言编程，所以读者既应该注意把 C 语言编程技巧应用到 Windows 的编程中去，又要注意到 Windows 是面向对象的编程方法，因而采用的学习方法也应与学习传统的面向过程的语言的学习方法有所不同。

本章的目的主要是介绍 Windows 应用程序的特点，应用程序所涉及的对象及应用程序开发的过程。目的不是要读者弄懂程序及程序中语句的用法，而是比较这些程序并分析它们的特点。在以后的章节中，将再次深入分析这些程序。本章对初学者来说，有很多新的概念，但本章也只是让读者先接触这些新概念，以便讨论一下开始学习 Windows 编程时的难点。

1.1　C 程序多文件编制方法

这里复习一下在 Boloand C++集成环境下，用工程文件实现编译和连接的方法。如果读者过去没有编制多文件的经验，应该练习编制多文件及工程文件的编制方法。因为本书是采取多文件编程以突出 Wndows 程序的变化特点。

1.1.1　多文件编程实例

我们编制一个求函数 $10x^2-9x+2$ 在区间 [0, 1]内 x 以 0.01 的增量变化的最小值的程序。方法是把主函数编在文件 MC.C 中，定义的函数编在 MC1.C 中，使用头部文件 MC.H 及工程文件 MC.PRJ。编制步骤如下：

1. 编辑头部文件 MC.H

```
#include    <stoid.h>
#define     s1          0.0
#define     s2          1.0
#define     step        0.01
double      func(double);
double      value(double(*f)());
```

2. 编辑文件 MC.C

```
/*  主函数 main()  */
main ()
```

```
{
    double ( * p)( );
    p=func;     /*  指向目标函数  */
    printf("最小值是:%2.3f\n", value(p));
}
```

3. 编辑文件 MC1.C

```
/*  目标函数  */
double func(x);
double x;
{
    return (10 * x * x-9 * x+2);
}
/*  定义求最小值函数,它包括函数指针  */
double value(double( * f)());
{
    double x, y=( * f)(x);
    while( x <= s2 ) {
            if( y > ( * f)(x) )
                y=( * f)(x );
            x += step;
        }
    return y;
}
```

4. 编辑工程文件 MC.PRJ

MC.C

MC1.C

注意:工程文件不是在编辑环境下编辑,它的产生一 ‘ 见下节。

1.1.2 多个C文件的连接

在 BORLAND C++ V3.1 集成环境下编辑、编译和连接上述应用程序包含以下步骤:

(1) 假定在 BIN 目录下建立个人目录 LIU 并将输出目录设置为:

\BORDLAND\BIN\LIU

(2) 选择 Options 并进入 Application 菜单。这时出现 Set Application Options 对话框,用 Tab 键或鼠标选择 DOS Stabdard 或 DOS Overlay 环境项(前两个选择项)。

(3) 在集成环境下选 File 菜单,分别编辑 mc.c 和 mc1.c 源程序。

(4) 选择 Proect 并进入 Open 菜单。键入文件名 mc.prj(使用扩展名.prj)。如果是第一次编辑 mc.prj,按小键盘上的 Insert 键进入 Project Name 对话框。使用 Insert 和 Delete 键可以进行增删,Done 键结束编辑。本程序在多文件方式下加入如下文件:

mc.c

mc1.c

(5) 选择 Cmpile 并进入 Bulid all 菜单，自动编辑并连接出 mc.exe 文件。

运行文件 mc，得出如下结果：

最小值是：- 0.025

1.2 Windows 程序实例

1.2.1 实例程序总体设计思想

本节采取与处理 C 程序一样，把源文件编成两个文件。不过，它们除了要求一个头文件和工程文件之外，还增加一个模块定义文件 *.DEF。【例 1.4】则更特殊一些，尚需增加一个后缀为.RC 的资源文件。

像 C 语言一样，我们可以为这 4 个例子设计一个公共的头文件。为了简单，我们还为它们设计了公共的模块定义文件和工程文件(其实是为第一章～第三章的全部实例设计的公共文件)。这些公共文件的名字如下：

(1) 文件 W13Com.C 仅有 WinMain 函数；

(2) 文件 W13Com.H 包含对这 2 个函数的说明及引用的头文件；

(3) 文件 W13Com.DEF 是模块定义文件；

(4) 文件 W13Com.PRJ 包括如下文件：

W13Com.DEF

W13Com.C

Wnd13.C

W13Com.RC

虽然只有【例 1.4】才用到资源文件 W13Com.RC，但我们让工程文件都含有它。目的是为了使读者对它有个认识，并借助它说明查错的简单方法。

为了方便读者，我们提供了全部实例的源程序和执行文件。它们的名字与例题号相对应。例如本节的【例 1.1】～【例 1.4】的文件除上述公共文件之外，在磁盘里分别具有 W11.C～W14.C 的程序名及 W11.EXE～W14.EXE 的执行文件名。在本节中，它们都以 Wnd13.C (仅含 WndProc 函数)的文件名给出以求得工程文件及模块定义文件名的统一。读者自己进行编译时，只要把实例文件改一下文件名即可。例如：

```
copy   W13.c   Wnd13.c        实验【例 1.3】
copy   W14.c   Wnd13.c        实验【例 1.4】
```

我们给它们的运行程序名均为 W13Com.EXE，含义是第一章到第三章均是如上方式。本节主要是让读者注意 Wnd13.C 的变化方法，程序里具体语句的用法放在第二章及第三章的相应章节中讲授。

1.2.2 程序编辑、编译与连接

本节程序在 BORLAND C++ V3.1 集成环境下编辑、编译和连接的步骤如下：

(1) 假定在 BIN 目录下建立用户目录 LIU。

(2) 保证在根目录中的 autoexec.bat 文件设置的路径包括 C:\BORLANDC\BIN。

(3) 在目录 LIU 中运行 BC 后，选 Options 菜单并进入 Directories 菜单，将包含目录、库目录、输出目录和源程序目录分别设置如下：

Include Directories

C:\BORLANDC\INCLUDE

Include Directories

C:\BORLANDC\LIB

Include Directories

C:\BORLANDC\BIN\LIU

Include Directories

C:\BORLANDC\BIN\LIU

(4) 选择 Options 并进入 Application 菜单。这时出现 Set Application Options 对话框，用 Tab 键或鼠标选择 Windows APP 项。

(5) 选择 File 菜单，分别编辑各个文件的源程序。

(6) 选择 Proect 并进入 Open 菜单。键入文件名 W13Com.prj(使用扩展名.prj)。如果是第一次编辑 W13Com.prj，按小键盘上的 Insert 键进入 Project Name 对话框。Insert 和 Delete 键可以进行增删，Esc 或 Done 键结束编辑。本程序在多文件方式下加入如下文件：

W13Com.def

W13Com.c

Wnd13.c

W13Com.rc

(7) 选择 Cmpile 并进入 Bulid all 菜单，自动编辑并连接出 W13Com.exe 文件。

在 Windows 环境下直接输入如下命令，程序运行，得到窗口图形。

win W13Com

使用工程文件 *.prj 很方便，本书将全部使用这种方法。另外，可以不编制模块定义文件 *.def 而使用默认的 *.def 文件，这只要在 *.prj 中不包括 *.def 即可。当然，系统会给出一个无 *.def 文件的警告，同时告诉你系统自动采用了默认值。

假如未使用自定义模块文件，编译时显示的信息如下：

Linker Warning: No module definition file specified: using defaults

我们仍然给出 *.def 文件以减少编译时出现的警告信息。

1.2.3 实例

【例 1.1】编制一个在窗口的标题条上显示“Chapter One Example 1.1”信息的 Windows 程序。

```
/*  文件:  W13Com.c  */
#include <windows.h>  /*  所有 Windows 程序都要包含它  */
#include <W13Com.h>
/*  名字唯一的主函数 WinMain()  */
int PASCAL WinMain(hInstance, hPrevInstance, lpCmdLine, nCmdShow)
HANDLE   hInst;
HANDLE   hPrevInstance;
LPSTR    lpCmdLine;
int      nCmdShow;
{
    HWND     hWnd;
    MSG      msg;
    WNDCLASS wc;

    if (! hPrevInstance)
    {                                   /*  定义窗口类      */
        wc.style = CS_HREDRAW|CS_VREDRAW;
        wc.lpfnWndProc = WndProc;  /*  窗口过程函数    */
        wc.cbClsExtra = 0;
        wc.cbWndExtra = 0;
        wc.hInstance = hInst;
        wc.hIcon = LoadIcon(NULL, IDI_APPLICATION);
        wc.hCursor = LoadCursor(NULL, IDC_ARROW);
        wc.hbrBackground = GetStockObject(WHITE_BRUSH);
        wc.lpszMenuName = szMenuName;
        wc.lpszClassName = szClassName;
        RegisterClass(&wc);             /*  登录窗口类      */
    }

    hWnd = CreateWindow(                /*  创建用户窗口    */
        szClassName,
        szAppTitle,                     /*  信息在这里      */
        WS_OVERLAPPEDWINDOW|
        WS_VSCROLL|WS_HSCROLL,
```

```
            CW_USEDEFAULT,
            CW_USEDEFAULT,
            CW_USEDEFAULT,
            CW_USEDEFAULT,
            NULL,
            NULL,
            hInst,
            NULL
            );

    ShowWindow(hWnd, nCmdShow);   /*  显示窗口        */
    UpdataWindow(hWnd);           /*  更新窗口        */
        while (GetMessage(&msg,   /*  消息循环        */
                NULL,
                NULL,
                NULL))
        {
    TranslateMessage(&msg);       /*  检索消息        */
    DispatchMessage(&msg);        /*  发送消息        */
        }
    return (msg.wParam);
}

/*  文件:Wnd13.c  */
#include <windows.h>   /*  所有 Windows 程序都要包含它  */
#include <W13Com.h>

extern char szClassName[ ]="W13Com";
extern char szMenuName[ ]="NULL";
extern char szAppTitle[ ]="Chapter One Example 1.1";

/*  窗口过程函数 WndProc ( )  */
long FAR PASCAL WndProc(hWnd, message, wParam, lParam)
HWND            hWnd;
unsigned        message;
WORD            wParam;
LONG            lParam;
{
    return (DefWindowProc(hWnd, message, wParam, lParam));
```

```
}

/*   文件 W13Com.h   */
#include <stdio.h>
#include <string.h>

#define  IDM_SMALL     200   /*  【例 1.4】的菜单 ID 定义   */
#define  IDM_MEDIUM    201
#define  IDM_LARGE     202

extern char szClassName[ ];
extern char szMenuName[ ];
extern char szAppTitle[ ];

int PASCAL WinMain(HANDLE, HANDLE, LPSTR, int);
long FAR PASCAL WndProc(HWND, unsigned, WORD, LONG);
/*   文件:W13Com.def   */
NAME            W13Com
DESCRIPTION     'Simple Example'
EXETYPE         WINDOWS
STUB            'WINSTUB.EXE'
CODE PRELOAD MOVEABLE DISCARDABLE
DATA PRELOAD MOVEABLE MULTIPLE
HEAPSIZE        1024
STACKSIZE       5120
EXPORTS         WndProc
/*   资源文件 W13Com.RC   */
/*   【例 1.4】使用的菜单资源   */
#include "W13Com.h"

W13Com MENU
BEGIN
      POPUP "Select_Pie_Size"
        {
            MENUITEM   "&Small",    IDM_SMALL
            MENUITEM   "&Medium",   IDM_MEDIUM
            MENUITEM   "&Large",    LARGE
        }
END
```

我们所做的工作只是用 WinMain 函数注册了一个显示标题的窗口，这个显示的信息称为标题横条，左边的带阴影的矩形可以用鼠标激活或同时按住 Alt 和空格键选取它。选中后，弹出一个系统菜单，该系统菜单有 7 个菜单项，提供了恢复(Restore)、移动(Move)、改变框架大小(Size)、缩至最小(Minimize)、放至最大(Maximize)、关闭(Close)及切换(Next)应用程序等 7 项操作。如果读者使用的是中文 Windows 3.1 版本，这个菜单就是确定的中文信息；如果是西文 Windows 3.1，就显示英文信息。

右上角向下的箭头使窗口缩小为图标，向上的箭头使窗口放大充满全屏，但这两个选项只能通过鼠标实现。这些都是利用 Windows 系统提供的函数 DefWindowProc 实现的。

这个最简单的程序向读者演示了缺省窗口处理过程 DefWindowProc。通过它，实现了很复杂的操作，但用户可以不管它是如何实现的。Windows 的设计者再次向用户显示这样一个道理：将程序建立在“需要懂得为什么”基础上的思想并不一定行得通。

Windows 的封装、黑匣子和信息隐蔽都是正确的。如果没有事情是隐蔽的，任何事情都是公开的，我们将一事无成。

中国有句俗话：“知其然，不知其所以然”，在这里有了积极含义。在开始学习之初，应仔细选择和描述标准的例子，知道不要深究还是一个正确的入门方法。难怪“你只要管用句柄而不用知道它指向什么”这句鼓励不求甚解的话，到成了 Windows 程序设计者的“经文”。当然，如果具备了要“知其所以然”的条件，又作别论。

因为我们的目的是比较，所以读者还是不要急于弄懂上面的 Windows 程序的原理。

在编译程序时，因为程序 Wnd1.C 中定义

```
extern char szMenuName[ ]="NULL";
```

所以运行后的窗口图如 1.1 所示。图中还给出打开系统菜单的示意图。

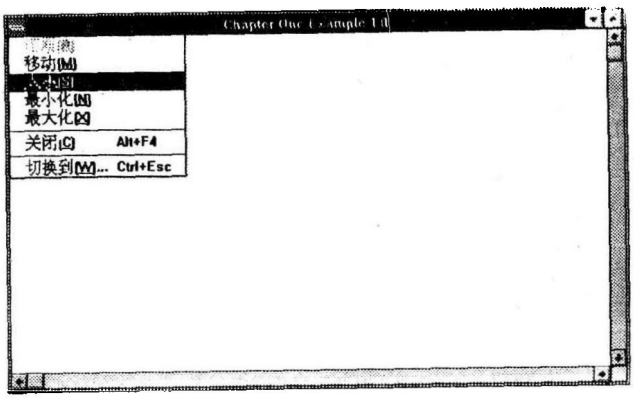

图 1.1 窗口及系统菜单信息示意图

假如在编译【例 1.1】的程序时，我们在工程文件中取消资源文件 W13Com.rc。取消的方法很简单，只要同时按下 Alt 键和代表工程文件的信息窗号(例如工程文件的信息窗代号是 5，则按 Alt+5。“+”号表示按住 Alt 键的之后不要松手，再按下数字键 5)，再用上、下箭头键把指示选中的亮度调到文件 W13Com.RC 处，按 Delete 键即把该文件从工程文件中删除。如果不知道工程文件信息窗的代号，按住 Alt 键后，再依次按数字键 1、2・・・，直到屏幕底部窗口中出现工程文件编辑窗。编译成功后出现图 1.2 的画面。

```
EXE File W13COM.EXE
LinKing:          \BORLANDC\LIB\CWS.LIB

                        Total     Link
          Lines compiled: 12082     PASS  2
        Warnings:         1         0
          Errors:         0         0

Available memory:         1143K
Sucess          :             Press ank key
```

图 1.2 编译、连接成功信息示意图

语句“Pess any key”不断闪烁。按任意键之后，出现一条显示在当前信息带内的信息为：

```
• Warning W13COM.C 64:Parameter 'lpCmdLine' is never used
```

这只是说我们的应用程序中没有使用参数“lpCmdLine”，而我们确实也用不到此参数。选 File 菜单的 Quit 项(或同时按 Alt 和 X 键)退出。如果有修改过而没有存储的文件，将给出提示信息由用户采取相应措施。

假如程序有错，将给出错误的数量。按回车后，将给出相应信息供查错。

由上可见，头文件的处理方法与 C 语言一样。

如果有兴趣，可以再选择工程文件编辑窗口增加文件 W13Com.RC，然后编译这个程序，编译完成后，出现如图 1.3 所示的信息。

```
Main File:W13COM.EXE

                        Total     Link
          Lines compiled: 12158     0
        Warnings:         1         0
          Errors:         0         0

Available memory:         1143K
Sucess          :             Press ank key
```

图 1.3 装入资源文件信息示意图

观看一下编译过程就会发现：这两种方法的编译过程虽然有异同，但运行结果却一样。这是为什么？

仅仅为了输出这样一句话，就花费了这么多的语句。是否比 DOS 下的 C 程序效率低得多？

【例 1.2】在上面的窗口里输出“How are you?”。

这个程序只要修改 Wnd13.c 文件的内容即可，程序的执行结果如图 1.2。修改后的源程序如下：

```
/*   文件 Wnd13.c( )   */
#include <windows.h>
#include "W13Com.h"

extern char szClassName[ ]="W13Com";
extern char szMenuName[ ]="NULL";
extern char szAppTitle[ ]="Chapter One Example 1.2";
/*   函数 WndProc( )   */
long   FAR   PASCAL   WndProc(hWnd, message, wParam, lParam)
HWND          hWnd;
unsigned      message;
WORD          wParam;
LONG          lParam;
{
     HDC   hDC;

     switch (message) {
          case WM_PAINT:
               hDC=GetDC(hWnd);
               TextOut(hDC,100,100,"How are you?",12);
               ReleaseDC(hWnd,hDC);
               break;
          case WM_DESTROY:
               PostQuitMessage(0);
               break;

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

这个程序在函数 WndProc 里增加了 switch 开关语句意处理 Windows 消息 WM_PAINT 和 WM_DESTROY。前者用来绘图，内容随对像而变化。后者用以退出程序，这里是标准方式。

在以下的例子中，我们均使用含有 W13Com.rc 文件的工程文件。编译程序在完成产生