## 第一章开始

Windows运作机制的核心是一个称作“动态连结”的概念。

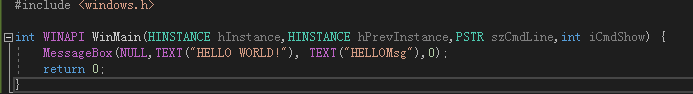
动态连结程式库（dynamic linking library）DLL

Windows的主要部分仅通过三个动态连结程式库实作。这代表了windows的三个主要子系统，他们被称作kernel,user和GDI

Kernel处理所有在传统上由作业系统核心处理的事物——记忆体管理、档案I/O和多工管理

Use指使用者界面，实作所有视窗运作机制。

GDI是一个图形装置界面，允许程式在荧幕和印表机上显示文字和图形。

这是第一个程序

Windows.h是主要的含入档案，它包含了其他windows表头档案，这些表头档案中最重要和最基本的是：

Windef.h,基本形态定义

winnt.h,支援Unicode的型态定义

winbase.h,kernel函式

winuser.h,使用者界面函式

wingdi.h,图形装置界面函式

正如在C程式中的进入点是函数main一样，windows程式的进入点是winmain

变量字首i表示int，sz表示“以零结束的字串”

Winmain函式宣告为返回一个int值。Winapi识别字在windef.h定义，语句为:

#define WINAPI \_\_stdcall

该语句指定了一个呼叫约定，包括如何生产机械码以在堆叠中放置函式呼叫的参数，许多windows函式呼叫宣告为WINAPI

Winmain的第一个参数被称作“执行体代号”，在windows程式设计中，代号仅是一个应用程式用来识别某些东西的数字，需要它在其他windows函式呼叫中作为参数。

在32位元windows版本中，传给winmain的第二个参数总是NULL。

Winmain的第三个参数是用于执行程式的命令列。某些windows应用程式利用它在程式启动时将档案载入记忆体。

Winmain的第四个参数指出程式最初显示的方式，可以是正常的或者是最大化地充满整个画面，或者是最小化显示在工作列中。

MessageBox函式：

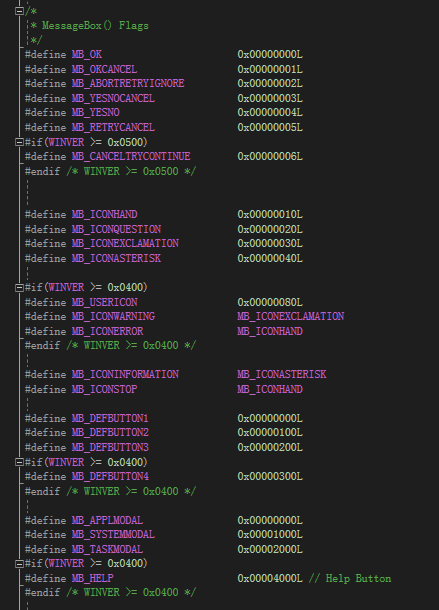
用于显示短资讯——对话方块

第一个参数通常是视窗代号

第二个参数是在讯息方块主体中显示的字串

第三个参数是出现在讯息方块标题上的字串，这些文字字串的每一个都被封装在一个TEXT巨集中。

MessageBox的第四个参数可以使在winuser.h中定义的一组字首MB\_开始的常数的组合，



## 第二章 Unicode简介

问题：世界上的书写预言不能简单地用256 个8位元代码表示。

Unicode使用“宽字元集”Unicode中的每个字元都是16位宽而不是8位宽。前128个Unicode字元就是ASCII字元，而接下来的128个unicode字元是ISO 8859-1对ASCII的扩展

宽字符：

C中的宽字符基于wchar\_t数据形态，它在几个表头文件包括WCHAR.H中都有定义

Typedef unsigned short wchar\_t

因此wchar\_t数据形态与无符号短整数形态相同，都是16位宽。

Strlen函数的宽字符版是wcslen,并且在string.h和wchar.h中均有说明：

Size\_t \_\_cdecl strlen(const char\*)

Size\_t\_\_codecl wcslen(const wchar\_t \*)

\_tcslen，\_tprintf在两种字符集中均可使用。

TCHAR在两种字符集中均可定义字符串变量。

一个基于WCHAR数据形态的变量可在前面附加上字母wc以说明一个宽字符。

WINNT.h表头文件进而定义了可用做8位字符串指针的六种数据形态和四个可用作const8位字符串指针的数据形态。

Typedef char \* PCHAR,\*LPCH,\*PCH,\*NPSTR,\*LPSTR,\*PSTR;

Typedef const char \* LPCCH,\*PCCH,\*LPSTR,\*PCSTR;

前缀N和L表示near 和long,在win32中near和long指标没有区别。

类似地，WINNT.H定义了六种可作为16位字符串指针的数据形态和四中可作为const16位字符串指针的数据形态。

Typedef wchar \*PWCHAR,\*LPWCH,\*PWCH,\*NWPSTR,\*LPWSTR,\*PWSTR;

Typedef const wchar \*LPCWCH,\*PCWCH,\*LPCWSTR,\*PCWSTR;

Windows函数呼叫

Int WINAPI MessageBox(HWND,LPCSTR,LPCSTR,UNIT);

函数的第二个，第三个参数是指向常数字符串指针。当编译连结一个win16程序时，windows并不处理messagebox呼叫。程序.exe文件中的表格，允许windows将该程序的呼叫与user中的messageBox函数动态链接起来。

除了含有与16位兼容的user.exe外，还含有一个称为user32.dll的动态链接库，该动态链接库含有32位使用者借口函数的进入点，包括32位的messageBox。实际上在user32.dll中没有32位messagebox函数的进入点，但有一个名为messageboxA和messageboxW的进入点。但并需要担心使用问题，只需要使用messagebox即可。

Windows的字符串函数

ILength = lstrlen (pString) ;

pString = lstrcpy (pString1, pString2) ;

pString = lstrcpyn (pString1, pString2, iCount) ;

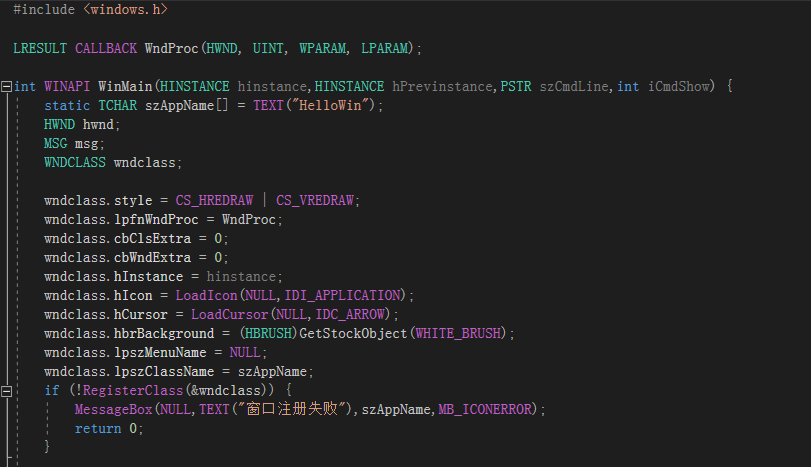
pString = lstrcat (pString1, pString2) ;

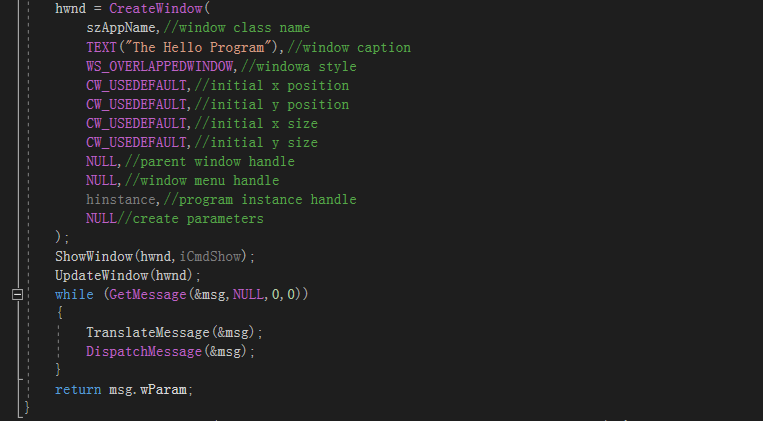
iComp = lstrcmp (pString1, pString2) ;

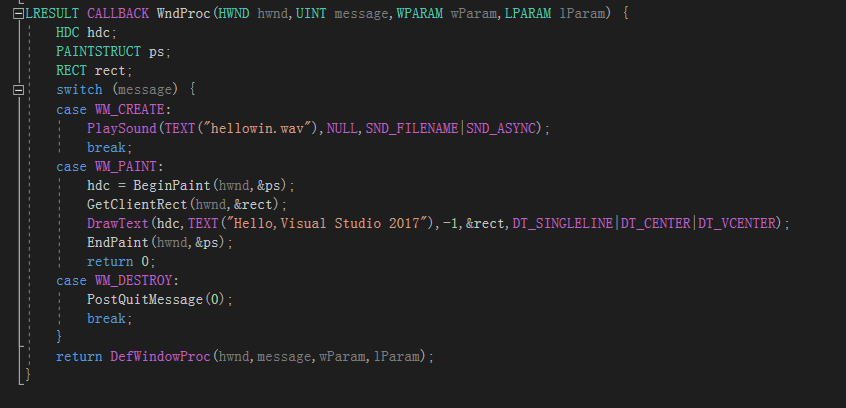
iComp = lstrcmpi (pString1, pString2) ;

## 第三章窗口和消息

进行windows程序设计，实际上是在进行一种对象导向的程序设计（OOP）。







Windows函数呼叫

HelloWin至少呼叫了18个windows函数。

LoadIcon加载图标供程序使用。

LoadCursor加载鼠标光标供程序使用。

GetStockObject取得一个图形对象（在这个例子中，是取得回执窗口背景的画刷对象）。

RegisterClass为程序窗口注册窗口类别。

MessageBox显示消息框。

CreateWindow根据窗口类别建立一个窗口。

ShowWindow在屏幕上显示窗口。

UpdateWindow指示窗口自我更新。

GetMessage从消息队列中取得消息。

TranslateMessage转译某些键盘消息。

DispatchMessage将消息发送给窗口消息处理程序。

PlaySound播放一个声音文件。

BeginPaint 开始绘制窗口

GetClientRect 取得窗口显示区域的大小。

DrawText 显示字符串。

EndPaint 结束绘制窗口。

PostQuitMessage在消息队列中插入一个退出程序消息。

DefWindowProc执行内定的消息处理。

大写字母标识符

CS 窗口类别样式

CW 建立窗口

DT 绘制文字

IDI 图示ID

IDC 游标ID

MB 消息框

SND 声音

WM 窗口消息

WS 窗口样式

新的数据形态

MSG 消息结构

WNDCLASS 窗口类别结构

PAINTSTRUCT 绘图结构

RECT 矩形结构

句柄简介

HINSTANCE 执行实体（程序自身）句柄

HWND 窗口句柄

HDC设备内容句柄

句柄是一个整数，它代表一个对象。Windows中的句柄类似传统C或者MS\_DOC程序设计中使用的文件句柄。程序几乎总是通过呼叫windows函数取得句柄。程序在其他Windows 函数中使用这个句柄，以使用它代表的对象。代号的实际值对程序来说是无关紧要的。但是，向您的程序提供代号的windows模块知道如何利用它来使用相对应的对象。

匈牙利表示法

变量名以一个或者多个小写字母开始，这些字母表示变量的数据形态。

匈牙利表示法能够帮助程序写作者及早发现并避免程序中错误。由于变量名既描述了变量的作用，又描述了其数据形态，就比较容易避免产生数据形态不合的错误。

c char或WCHAR或TCHAR

by BYTE(无正负号字符)

n short

I int

X,y int分别用作x坐标和y坐标

Cx,cy int分别用作x长度和y长度；c代表计数器

b或fBOOL(int);f代表旗标

w word(无正负号短整数)

l long(长整数)

dw DWORD(无正负号长整数)

fn function(函数)

s string(字符串)

sz 以字节值0结尾的字符串

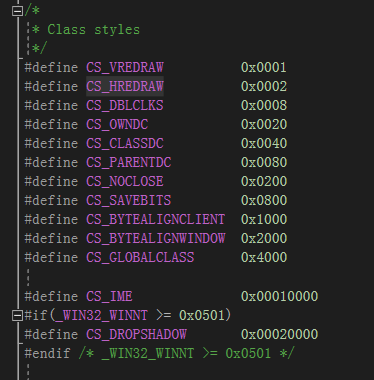
h 句柄

p 指针

注册窗口类别

Wndclass.style=CS\_HREDRAW|CS\_VREDRAW

使用C的位[或]运算子结合了两个[窗口类别样式]标识符.



由于每个标识符都可以在一个复合值设置一个位的值所以按这种方式定义的标识符通常称为位旗标.

Wndclass.lpfnWndProc=WndProc

这个过程将处理依据这个窗口类别建立的所有窗口的全部消息.在C语言中,像这样在结构中使用函数名时,真正提供的是指向函数的指针.

Wndclass.cbClsExtra=0;

Wndclass.cbWndExtra=0;

程序可以根据需要来使用预留的空间.

Wndclass.hinstance=hinstance;

程序的执行实体句柄.

Wndclass.hicon=LoadIcon(NULL,IDI\_APPLICATION);

为所有依据这个窗口类别建立的窗口设置一个图标.

要取得预先定义图示的句柄,可以将第一个参数设定为NULL来呼叫LoadIcon.在加载程序写作者自订的图标时,这个参数应该被设定为程序的执行实体句柄hInstance.第二个参数代表图示.对于预先定义图示,此参数是以IDI开始的标识符,标识符在WINUSER.H中定义.

Wndclass.hCursor=LoadCursor(NULL,IDC\_ARROW);

同上.

Wndclass.hbrBackground=GetStockObject(WHITE\_BRUSH);

画刷是一个绘图词汇,指用来填充一个区域的着色样式.windows有几个标准画刷,也称为备用画刷.

Wndclass.lpszMenuName=NULL;

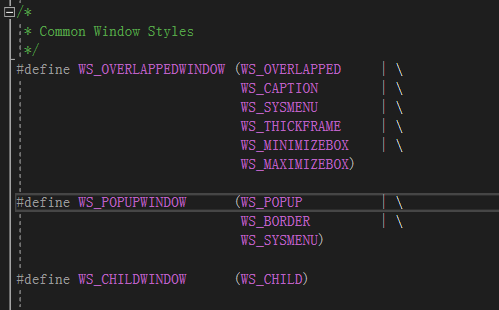
指定窗口类别菜单

Wndclass.lpszClassName=szAppName;

给出一个类别名称.类别名称可以与程序名相同.

建立窗口:

此程序建立的窗口是一个普通的重叠式窗口.它包含有一个标题列,标题列左边有一个系统菜单按钮,标题列右边有缩小放大关闭图示,四周还有一个表示窗口大小的边框.这是一个标准样式的窗口,名为WS\_OVERLAPPEDWINDOW,出现在CreateWindow的窗口样式参数中.



NULL

如果建立一个最上层窗口,如应用程序窗口时,注释为父窗口句柄的参数设定为NULL.通常,如果窗口之间存在有父子关系,则子窗口总是出现在父窗口上面.应用程序窗口出现在桌面窗口的上面.

NULL

因为窗口没有菜单,所以窗口菜单句柄也设定为NULL.程序执行实体句柄设定为执行实体句柄,他是作为winMain的参数传递给这个程序的.最后,建立参指标设定为NULL,可以用这个参数存取稍后程序中可能引用到的数据.

CreateWindow传回被建立的窗口的句柄，该句柄存放在变量hwnd中，后者被定义为HWND型态（「窗口句柄型态」）。Windows中的每个窗口都有一个句柄，程序用句柄来使用窗口。许多Windows函数需要使用hwnd作为参数，这样，Windows才能知道函数是针对哪个窗口的。如果一个程序建立了许多窗口，则每个窗口均有一个句柄。窗口句柄是Windows程序所处理最重要的句柄之一。

显示窗口:

在CreateWindow呼叫传回之后，Windows内部已经建立了这个窗口。这就是说，Windows已经配置了一块内存，用来保存在CreateWindow呼叫中指定窗口的全部信息跟一些其它信息，而Windows稍后就是依据窗口句柄找到这些信息的。

然而，光是这样子，窗口并不会出现在视讯显示器上。您还需要两个函数呼叫:

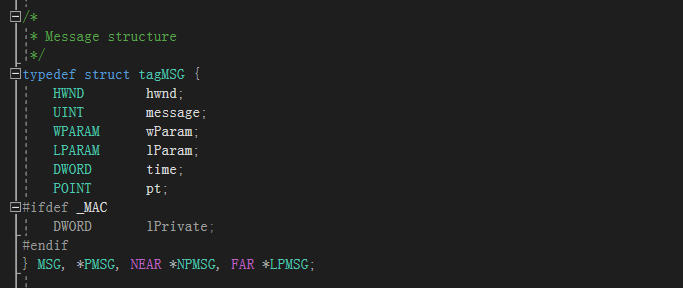
ShowWindow(hwnd,iCmdShow);

UpdateWindow(hwnd);

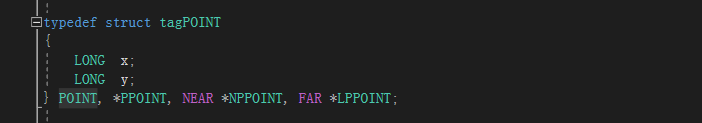
第一个参数是刚刚用CreateWindow建立的窗口句柄.第二个参数是作为参数传递给winMain的iCmdShow.它确定最初如何在屏幕上显示窗口,是一般大小,最小化还是最大化.在开始菜单中安装程序时,使用者可能做出最佳选择.如果窗口按一般大小显示,那么WinMain接收到后传递给ShowWindow的就是SW\_SHOWNORMAL;如果窗口是最大化显示的,则为SW\_SHOWMAXIMIZED.而如果窗口只显示在工作列上,则是SW\_SHOWMINNOACTIVE.

消息循环:

MSG形态在WINUSE.H中定义如下:



POINT数据形态也是一个结构,它在WINDEF.H中定义如下:



消息循环以GetMessage呼叫开始,它从消息队列中取出一个消息:

GetMessage(&msg,NULL,0,0);

这个呼叫传给windows一个指标,指向名为msg的MSG结构.第二,第三和第四个参数设定为NULL或者0,表示程序接收它自己建立的所有窗口的所有消息,Windows用从消息队列中取出的下一个消息来填充消息结构的各个字段,结构的各个字段包括:

Hwnd接收消息的窗口句柄.在hellowin程序中,这一参数与createWindow传回的hwnd值相同,因为这是该程序拥有的唯一窗口.

Message消息标识符.这是一个数值,用以标识消息.对于每个消息,均有一个对应的标识符,这些标识符定义于windows表头文件,以前缀WM开头.

WParam一个32位的message parameter,其含义和数值根据消息的不同而不同.

LParam一个32位的消息参数,其值与消息有关.

Time 消息放入消息队列中的时间.

Pt消息放入消息队列时的鼠标坐标.

只要从消息队列中取出消息的message字段不为WM\_QUIT（其值为0x0012），GetMessage就传回一个非零值。WM\_QUIT消息将导致GetMessage传回0。

TranslateMessage (&msg) ;

将msg结构传给Windows，进行一些键盘转换

DispatchMessage (&msg) ;

又将msg结构回传给Windows。然后，Windows将该消息发送给适当的窗口消息处理程序，让它进行处理。这也就是说，Windows将呼叫窗口消息处理程序。在HELLOWIN中，这个窗口消息处理程序就是WndProe函数。处理完消息之后，WndProc传回到Windows。此时，Windows还停留在DispatchMessage呼叫中。在结束DispatchMessage呼叫的处理之后，Windows回到HELLOWIN，并且接着从下一个GetMessage呼叫开始消息循环。

窗口消息处理程序:

窗口消息处理程序总是定义为如下形式：

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)

注意，窗口消息处理程序的四个参数与MSG结构的前四个字段是相同的。

第一个参数hwnd是接收消息的窗口的句柄，它与CreateWindow函数的传回值相同。对于与HELLOWIN相似的程序（只建立一个窗口），这个参数是程序所知道的唯一窗口句柄。如果程序是依据同一窗口类别（同时也是同一窗口消息处理程序）建立多个窗口，则hwnd标识接收消息的特定窗口。

第二个参数与MSG结构中的message字段相同，它是标识消息的数值。最后两个参数都是32位的消息参数，提供关于消息的更多信息。这些参数包含每个消息型态的详细信息。有时消息参数是两个存放在一起的16位值，而有时消息参数又是一个指向字符串或数据结构的指针。程序通常不直接呼叫窗口消息处理程序，窗口消息处理程序通常由Windows本身呼叫。

处理消息:

一般来说，Windows程序写作者使用switch和case结构来确定窗口消息处理程序接收的是什么消息，以及如何适当地处理它。窗口消息处理程序在处理消息时，必须传回0。窗口消息处理程序不予处理的所有消息应该被传给名为DefWindowProc的Windows函数。从DefWindowProc传回的值必须由窗口消息处理程序传回。

呼叫DefWindowProc来为窗口消息处理程序不予处理的所有消息提供内定处理，这是很重要的。不然一般动作，如终止程序，将不会正常执行。

播放声音文件:

PlaySound的第一个参数是声音文件的名称（它也可能是在Control Panel的Sounds中定义的一种声音的别名，或者是一个程序资源）。第二个参数只有当声音文件是一种资源时才被使用。第三个参数指定一些选项。在这个例子中，我指定第一个参数是一个文件名，并且异步地播放声音，即PlaySound函数呼叫在声音文件开始播放时立即传回，而不会等待它的完成。在这种方法下，程序能够继续初始化。

WM\_PAINT消息:

在使用者改变HELLOWIN窗口的大小后，显示区域的显示内容重新变得无效。读者应该还记得，HELLOWIN中wndclass结构的style字段设定为标志CS\_HREDRAW和CS\_VREDRAW，这样的格式设定指示Windows，在窗口大小改变后，就把整个窗口显示内容当成无效。然后，窗口消息处理程序将收到一条WM\_PAINT消息。

当使用者将HELLOWIN最小化，然后再次将窗口恢复为以前的大小时，Windows将不会保存显示区域的内容。在图形环境下，窗口显示区域涉及的数据量很大。因此，Windows令窗口无效，窗口消息处理程序接收一条WM\_PAINT消息，并自动恢复其窗口的内容。

对WM\_PAINT的处理几乎总是从一个BeginPaint呼叫开始：

hdc = BeginPaint (hwnd, &ps) ;

而以一个EndPaint呼叫结束：

EndPaint (hwnd, &ps) ;

如果窗口消息处理程序不处理WM\_PAINT消息（这是很罕见的），它们必须被传送给DefWindowProc。DefWindowProc只是依次呼叫BeginPaint和EndPaint，以使显示区域有效。

GetClientRect (hwnd, &rect) ;

第一个参数是程序窗口的句柄。第二个参数是一个指标，指向一个RECT型态的rectangle结构。该结构有四个LONG字段，分别为left、top、right和bottom。GetClientRect将这四个字段设定为窗口显示区域的尺寸。

DrawText ( hdc, TEXT ("Hello, Windows 98!"), -1, &rect, DT\_SINGLELINE | DT\_CENTER | DT\_VCENTER) ;

DrawText可以输出文字（正如其名字所表明的一样）。由于该函数要输出文字，第一个参数是从BeginPaint传回的设备内容句柄，第二个参数是要输出的文字，第三个参数是 -1，指示字符串是以字节0终结的。DrawText最后一个参数是一系列位旗标，它们均在WINUSER.H中定义.

WM\_DESTROY消息：

WM\_DESTROY消息是另一个重要消息。这一个消息指示，Windows正在根据使用者的指示关闭窗口。该消息是使用者单击Close按钮或者在程序的系统菜单上选择 Close时发生的。

PostQuitMessage (0) ;

该函数在程序的消息队列中插入一个WM\_QUIT消息。前面提到过，GetMessage对于除了WM\_QUIT之外的从消息队列中取出的所有消息都传回非0值。而当GetMessage得到一个WM\_QUIT消息时，它传回0。这将导致WinMain退出消息循环，并终止程序。然后程序执行下面的叙述：

return msg.wParam ;

结构的wParam字段是传递给PostQuitMessage函数的值（通常是0）。然后return叙述将退出WinMain并终止程序。

## 第三章输出文字

**WM\_PAINT消息:**

在何时窗口消息处理程序会接收到一个WM\_PAINT消息:

在使用者移动窗口或者显示窗口时,窗口中先前被隐藏的区域重新可见.

使用者改变窗口的大小(如果窗口类别样式有着CS\_HREDRAW和CS\_VREDRAW).

程序使用ScrollWindow或ScrollDC函数滚动显示区域的一部分.

程序使用InvalidateRect或InvalidateRgn函数刻意产生WM\_PAINT消息.

可能收到一个WM\_PAINT消息:

Windows擦除覆盖了部分窗口的对话框或消息框.

菜单下拉出来,然后被释放.

显示工具提示消息.

在某些情况下,windows总是保存它所覆盖的显示区域,然后恢复它:

鼠标光标穿越显示区域.

图标拖过显示区域.

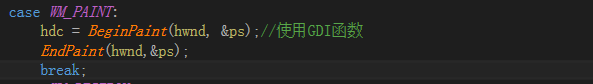
**GDI简介:**

TextOut (hdc, x, y, psText, iLength) ;

TextOut向窗口的显示区域写入字符串.psText参数是指向字符串的指针,iLength是字符串的长度.x,y参数定义了字符串在显示区域的开始位置.hdc参数是设备内容句柄,它是GDI的重要部分.实际上,每个GDI函数都需要将这个句柄作为函数的第一个参数.

**取得设备内容句柄:方法一**

一般的处理WM\_PAINT消息的形式如下:



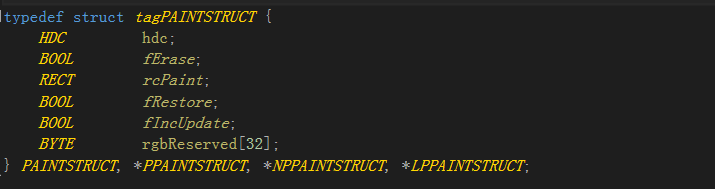
在处理WM\_PAINT消息时,必须成对地呼叫BeginPaint和EndPaint.

Case WM\_PAINT:

Return 0;

以上方法是错误的.windows将一个WM\_PAINT消息放到消息队列中,是因为现实区域中的一部分无效.如果不呼叫beginpaint和endpaint,则windows不会使该区域变为有效.相反,windows将发送另一个WM\_PAINT消息,且一直发送下去.

**绘图信息结构:**



在程序呼叫beginPaint时,windows会适当填入该结构的各个字段值.使用者程序只使用前三个字段.其他字段由windows内部使用.

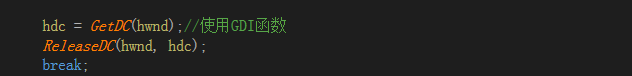
PAINTSTRUCT结构的rcPaint字段是RECT形态的结构.定义了无效矩形的边界,无效矩形是应该重画的区域.

在处理WM\_PAINT消息时,为了在更新的矩形外绘图,可以使用如下呼叫:

invalidateRect(hwnd,NULL,TRUE);

该呼叫在BeginPaint呼叫之前进行,它使整个显示区域变为无效,并擦除背景.,但是,如果最后一个参数等于False则不擦除背景,原有的东西将保留在原处.

**取得设备内容句柄:方法二**



与beginPaint传回设备内容句柄不同,GetDC传回的设备内容句柄具有一个剪取矩形,它等于整个显示区域.可以在显示区域的某一部分绘图,而不是在无效矩形上绘图.与BeginPaint不同,GetDC不会使任何无效区域变为有效的.如果需要使整个显示区域有效,可以呼叫

ValidateRect(hwnd,NULL);

与GetDC相似的函数是GetWindowDC.getDC传回用于写入窗口显示区域的设备内容句柄,而GetWindowDC传回写入整个窗口的设备内容句柄.例如,您的程序可以使用从GetWindowDC传回的设备内容句柄在窗口的标题列上写入文字.然而,程序同样也应该处理WM\_NCPAINT消息.

**TextOut:细节**

TextOut(hdc,x,y,psText,iLength);

第一个参数是设备内容句柄,设备内容的属性控制了被显示的字符串的特征.

PsText参数是指向字符串的指针,iLength是字符串中字符的个数.

**字符大小**

程序可以呼叫GetSystemMetrics函数以取使用者借口上各类视觉组件大小的信息.

