# 学习大纲

|  |
| --- |
|  |

# API用法说明

## CreateWindow函数

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 创建重叠、弹出窗口或子窗口。 它指定窗口类、窗口标题、窗口样式和窗口的初始位置和大小（可选）。 该函数还指定窗口的父或所有者（如果有）和窗口的菜单。  若要除了 **CreateWindow**支持的样式之外，还要使用扩展窗口样式，请使用 [CreateWindowEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-createwindowexa) 函数。 语法 C++复制  HWND CreateWindowA(  [in, optional] LPCSTR lpClassName,  [in, optional] LPCSTR lpWindowName,  [in] DWORD dwStyle,  [in] int x,  [in] int y,  [in] int nWidth,  [in] int nHeight,  [in, optional] HWND hWndParent,  [in, optional] HMENU hMenu,  [in, optional] HINSTANCE hInstance,  [in, optional] LPVOID lpParam  ); 参数 [in, optional] lpClassName  类型：**LPCSTR**  **null**-terminated 字符串或上一次调用 [RegisterClass](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-registerclassa) 或 [RegisterClassEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-registerclassexa) 函数创建的类原子。 原子必须位于 *lpClassName*的低序单词中;高序单词必须为零。 如果 *lpClassName* 是字符串，则指定窗口类名。 类名可以是注册到 **RegisterClass** 或 **RegisterClassEx**的任何名称，前提是注册该类的模块也是创建窗口的模块。 类名也可以是任何预定义的系统类名。 有关系统类名的列表，请参阅“备注”部分。  [in, optional] lpWindowName  类型：**LPCSTR**  窗口名称。 如果窗口样式指定标题栏，*lpWindowName* 指向的窗口标题将显示在标题栏中。 使用 **CreateWindow** 创建控件（如按钮、复选框和静态控件）时，请使用 *lpWindowName* 指定控件的文本。 使用 **SS\_ICON** 样式创建静态控件时，请使用 *lpWindowName* 指定图标名称或标识符。 若要指定标识符，请使用语法“#*num*”。  [in] dwStyle  类型：**DWORD**  正在创建的窗口的样式。 此参数可以是 [窗口样式值](https://learn.microsoft.com/zh-cn/windows/desktop/winmsg/window-styles)的组合，以及“备注”部分中指示的控件样式。  [in] x  类型：**int**  窗口的初始水平位置。 对于重叠窗口或弹出窗口，*x* 参数是窗口左上角的初始 x 坐标，以屏幕坐标表示。 对于子窗口，*x* 是窗口左上角相对于父窗口工作区左上角的 x 坐标。 如果此参数设置为 **CW\_USEDEFAULT**，系统将选择窗口左上角的默认位置，并忽略 *y* 参数。 **CW\_USEDEFAULT** 仅适用于重叠窗口;如果为弹出窗口或子窗口指定，则 *x* 和 *y* 参数设置为零。  [in] y  类型：**int**  窗口的初始垂直位置。 对于重叠窗口或弹出窗口，*y* 参数是窗口左上角的初始 y 坐标，以屏幕坐标表示。 对于子窗口，*y* 是子窗口左上角相对于父窗口工作区左上角的初始 y 坐标。 对于列表框，*y* 是列表框工作区左上角相对于父窗口工作区左上角的初始 y 坐标。  如果使用 **WS\_VISIBLE** 样式位设置创建重叠窗口，并将 *x* 参数设置为 **CW\_USEDEFAULT**，则 *y* 参数将确定窗口的显示方式。 如果 *y* 参数 **CW\_USEDEFAULT**，则窗口管理器会在创建窗口后使用 **SW\_SHOW** 标志调用 [ShowWindow](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-showwindow)。 如果 *y* 参数是一些其他值，则窗口管理器会调用 **ShowWindow**，该值作为 *nCmdShow* 参数。  [in] nWidth  类型：**int**  窗口的宽度（以设备单位为单位）。 对于重叠窗口，*nWidth* 是窗口的宽度、屏幕坐标或 **CW\_USEDEFAULT**。 如果 *nWidth* 为 **CW\_USEDEFAULT**，则系统会选择窗口的默认宽度和高度;默认宽度从初始 x 坐标扩展到屏幕右边缘，默认高度从初始 y 坐标扩展到图标区域的顶部。 **CW\_USEDEFAULT** 仅适用于重叠窗口;如果为弹出窗口或子窗口指定了 **CW\_USEDEFAULT**，*nWidth*，*nHeight* 设置为零。  [in] nHeight  类型：**int**  窗口的高度（以设备单位为单位）。 对于重叠窗口，*nHeight* 是窗口的高度（以屏幕坐标为单位）。 如果 nWidth 设置为 CW\_USEDEFAULT，则系统将忽略 nHeight。  [in, optional] hWndParent  类型：**HWND**  正在创建的窗口的父窗口或所有者窗口的句柄。 若要创建子窗口或拥有的窗口，请提供有效的窗口句柄。 对于弹出窗口，此参数是可选的。  若要创建 [仅消息窗口](https://learn.microsoft.com/zh-cn/windows/desktop/winmsg/window-features)，请向现有仅消息窗口提供 **HWND\_MESSAGE** 或句柄。  [in, optional] hMenu  类型：**HMENU**  菜单的句柄，或根据窗口样式指定子窗口标识符。 对于重叠或弹出窗口，*hMenu* 标识要与窗口一起使用的菜单;如果要使用类菜单，它可以 **NULL**。 对于子窗口，*hMenu* 指定子窗口标识符，这是对话框控件用来通知其父级事件的整数值。 应用程序确定子窗口标识符;对于具有相同父窗口的所有子窗口，它必须是唯一的。  [in, optional] hInstance  类型：**HINSTANCE**  要与窗口关联的模块实例的句柄。  [in, optional] lpParam  类型：**LPVOID**  指向通过 [CREATESTRUCT](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/ns-winuser-createstructa) 结构（**lpCreateParams** 成员）指向的值的指针，该值由 [WM\_CREATE](https://learn.microsoft.com/zh-cn/windows/desktop/winmsg/wm-create) 消息的 *lParam* 参数指向。 此消息在返回之前由此函数发送到创建的窗口。  如果应用程序调用 **CreateWindow** 来创建 MDI 客户端窗口，*lpParam* 应指向 [CLIENTCREATESTRUCT](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/ns-winuser-clientcreatestruct) 结构。 如果 MDI 客户端窗口调用 **CreateWindow** 创建 MDI 子窗口，*lpParam* 应指向 [MDICREATESTRUCT](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/ns-winuser-mdicreatestructa) 结构。 如果不需要其他数据，*lpParam* 可能会 **NULL**。 返回值 类型：**HWND**  如果函数成功，则返回值是新窗口的句柄。  如果函数失败，则返回值 **NULL**。 若要获取扩展的错误信息，请调用 [GetLastError](https://learn.microsoft.com/zh-cn/windows/desktop/api/errhandlingapi/nf-errhandlingapi-getlasterror)。 言论 在返回之前，**CreateWindow** 向窗口过程发送 [WM\_CREATE](https://learn.microsoft.com/zh-cn/windows/desktop/winmsg/wm-create) 消息。 对于重叠、弹出窗口和子窗口，**CreateWindow** 向窗口发送 **WM\_CREATE**、[WM\_GETMINMAXINFO](https://learn.microsoft.com/zh-cn/windows/desktop/winmsg/wm-getminmaxinfo)和 [WM\_NCCREATE](https://learn.microsoft.com/zh-cn/windows/desktop/winmsg/wm-nccreate) 消息。 **WM\_CREATE** 消息的 *lParam* 参数包含指向 [CREATESTRUCT](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/ns-winuser-createstructa) 结构的指针。 如果指定了 **WS\_VISIBLE** 样式，**CreateWindow** 发送激活和显示窗口所需的所有消息。  如果创建的窗口是子窗口，则其默认位置位于 Z 顺序的底部。 如果创建的窗口是顶级窗口，则其默认位置位于 Z 顺序的顶部（但除非创建窗口本身是最顶层的窗口）的顶部。  有关控制任务栏是否显示已创建窗口的按钮的信息，请参阅 [管理任务栏按钮](https://learn.microsoft.com/zh-cn/windows/desktop/shell/taskbar)。  有关删除窗口的信息，请参阅 [DestroyWindow](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-destroywindow) 函数。  可以在 *lpClassName* 参数中指定以下预定义的系统类。 请注意可在 *dwStyle* 参数中使用的相应控件样式。  展开表   |  |  | | --- | --- | | **系统类** | **意义** | | **按钮** | 指定一个小矩形子窗口，表示用户可以单击的按钮以打开或关闭它。 按钮控件可以单独使用，也可以在组中使用，并且可以标记或显示无文本。 当用户单击按钮控件时，按钮控件通常会更改外观。 有关详细信息，请参阅 [按钮](https://learn.microsoft.com/zh-cn/windows/desktop/Controls/buttons)  有关可在 *dwStyle* 参数中指定的按钮样式表，请参阅 [按钮样式](https://learn.microsoft.com/zh-cn/windows/desktop/Controls/button-styles)。 | | **COMBOBOX** | 指定由列表框和类似于编辑控件的选择字段组成的控件。 使用此样式时，应用程序应随时显示列表框或启用下拉列表框。 如果列表框可见，请在选择字段中键入字符会突出显示与键入的字符匹配的第一个列表框项。 相反，选择列表框中的项将显示选定字段中的选定文本。  有关详细信息，请参阅 [组合框](https://learn.microsoft.com/zh-cn/windows/desktop/Controls/combo-boxes)。 有关可以在 *dwStyle* 参数中指定的组合框样式表，请参阅 [组合框样式](https://learn.microsoft.com/zh-cn/windows/desktop/Controls/combo-box-styles)。 | | **EDIT** | 指定一个矩形子窗口，用户可以在其中键入键盘中的文本。 用户选择该控件，并通过按 Tab 键将其移动到该控件，并为其提供键盘焦点。 当编辑控件显示闪烁插入点时，用户可以键入文本;使用鼠标移动光标、选择要替换的字符，或将光标定位为插入字符;或使用 BACKSPACE 键删除字符。 有关详细信息，请参阅 [编辑控件](https://learn.microsoft.com/zh-cn/windows/desktop/Controls/edit-controls)。  有关可在 *dwStyle* 参数中指定的编辑控件样式表，请参阅 [编辑控件样式](https://learn.microsoft.com/zh-cn/windows/desktop/Controls/edit-control-styles)。 | | **LISTBOX** | 指定字符串列表。 每当应用程序必须提供名称列表（如文件名）时，即可指定此控件，用户可以从中选择这些名称。 用户可以通过单击字符串来选择字符串。 突出显示所选字符串，并将通知消息传递到父窗口。 有关详细信息，请参阅 [列表框](https://learn.microsoft.com/zh-cn/windows/desktop/uxguide/ctrl-list-boxes)。  有关可在 *dwStyle* 参数中指定的列表框样式表，请参阅 [列表框样式](https://learn.microsoft.com/zh-cn/windows/desktop/Controls/list-box-styles)。 | | **MDICLIENT** | 指定 MDI 客户端窗口。 此窗口接收控制 MDI 应用程序的子窗口的消息。 建议的样式位 **WS\_CLIPCHILDREN** 和 **WS\_CHILD**。 指定 **WS\_HSCROLL** 和 **WS\_VSCROLL** 样式以创建 MDI 客户端窗口，允许用户将 MDI 子窗口滚动到视图中。  有关详细信息，请参阅 [多个文档界面](https://learn.microsoft.com/zh-cn/windows/desktop/winmsg/multiple-document-interface)。 | | **RichEdit** | 指定 Microsoft Rich Edit 1.0 控件。 此窗口允许用户使用字符和段落格式查看和编辑文本，并且可以包括嵌入的组件对象模型 （COM） 对象。 有关详细信息，请参阅 [富编辑控件](https://learn.microsoft.com/zh-cn/windows/desktop/Controls/rich-edit-controls)。  有关可在 *dwStyle* 参数中指定的富编辑控件样式表，请参阅 [Rich Edit Control Styles](https://learn.microsoft.com/zh-cn/windows/desktop/Controls/rich-edit-control-styles)。 | | **RICHEDIT\_CLASS** | 指定 Microsoft Rich Edit 2.0 控件。 此控件允许用户使用字符和段落格式查看和编辑文本，并且可以包括嵌入的 COM 对象。 有关详细信息，请参阅 [富编辑控件](https://learn.microsoft.com/zh-cn/windows/desktop/Controls/rich-edit-controls)。  有关可在 *dwStyle* 参数中指定的富编辑控件样式表，请参阅 [Rich Edit Control Styles](https://learn.microsoft.com/zh-cn/windows/desktop/Controls/rich-edit-control-styles)。 | | **SCROLLBAR** | 指定一个矩形，该矩形包含滚动框，并在两端具有方向箭头。 每当用户单击控件时，滚动条就会向其父窗口发送通知消息。 父窗口负责根据需要更新滚动框的位置。 有关详细信息，请参阅 [滚动条](https://learn.microsoft.com/zh-cn/windows/desktop/Controls/scroll-bars)。  有关可在 dwStyle 参数 中指定的滚动条控件样式表，请参阅 滚动条控件样式。 | | **STATIC** | 指定用于标记、框或其他控件的简单文本字段、框或矩形。 静态控件不采用任何输入并提供任何输出。 有关详细信息，请参阅 [静态控件](https://learn.microsoft.com/zh-cn/windows/desktop/Controls/static-controls)。  有关可以在 *dwStyle* 参数中指定的静态控件样式表，请参阅 [静态控件样式](https://learn.microsoft.com/zh-cn/windows/desktop/Controls/static-control-styles)。 |     **CreateWindow** 作为对 [CreateWindowEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-createwindowexa) 函数的调用来实现，如下所示。  syntax复制  #define CreateWindowA(lpClassName, lpWindowName, dwStyle, x, y, nWidth, nHeight, hWndParent, hMenu, hInstance, lpParam)\  CreateWindowExA(0L, lpClassName, lpWindowName, dwStyle, x, y, nWidth, nHeight, hWndParent, hMenu, hInstance, lpParam)  #define CreateWindowW(lpClassName, lpWindowName, dwStyle, x, y, nWidth, nHeight, hWndParent, hMenu, hInstance, lpParam)\  CreateWindowExW(0L, lpClassName, lpWindowName, dwStyle, x, y, nWidth, nHeight, hWndParent, hMenu, hInstance, lpParam)  #ifdef UNICODE  #define CreateWindow CreateWindowW  #else  #define CreateWindow CreateWindowA  #endif 例子 有关示例，请参阅 [使用窗口类](https://learn.microsoft.com/zh-cn/windows/desktop/winmsg/using-window-classes)。  **备注**  winuser.h 标头将 CreateWindow 定义为别名，该别名根据 UNICODE 预处理器常量的定义自动选择此函数的 ANSI 或 Unicode 版本。 将非中性编码别名与非非编码的代码混合使用可能会导致编译或运行时错误不匹配。 有关详细信息，请参阅函数原型的 约定。 要求  |  |  | | --- | --- | | **标头** | winuser.h （包括 Windows.h） |  另请参阅 [关于多个文档界面](https://learn.microsoft.com/zh-cn/windows/desktop/winmsg/about-the-multiple-document-interface)  [常见控制窗口类](https://learn.microsoft.com/zh-cn/windows/desktop/Controls/common-control-window-classes)  **概念**  [CreateWindowEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-createwindowexa)  [DestroyWindow](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-destroywindow)  [EnableWindow](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-enablewindow)  **其他资源**  **参考**  [RegisterClass](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-registerclassa)  [RegisterClassEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-registerclassexa)  [ShowWindow](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-showwindow)  [WM\_COMMAND](https://learn.microsoft.com/zh-cn/windows/desktop/menurc/wm-command)  [WM\_CREATE](https://learn.microsoft.com/zh-cn/windows/desktop/winmsg/wm-create)  [WM\_GETMINMAXINFO](https://learn.microsoft.com/zh-cn/windows/desktop/winmsg/wm-getminmaxinfo)  [WM\_NCCREATE](https://learn.microsoft.com/zh-cn/windows/desktop/winmsg/wm-nccreate)  [WM\_PAINT](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/wm-paint)  [Windows](https://learn.microsoft.com/zh-cn/windows/desktop/winmsg/windows) |

## GetMessage函数

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 从调用线程的消息队列中检索消息。 函数调度传入的已发送消息，直到已发布的消息可供检索。  **GetMessage** 函数类似于 [PeekMessage](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-peekmessagea)，但是， **GetMessage** 会阻止，直到发布消息后再返回。 语法 C++  BOOL GetMessage(  [out] LPMSG lpMsg,  [in, optional] HWND hWnd,  [in] UINT wMsgFilterMin,  [in] UINT wMsgFilterMax  ); 参数 [out] lpMsg  类型： **LPMSG**  指向 [MSG](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/ns-winuser-msg) 结构的指针，该结构从线程的消息队列接收消息信息。  [in, optional] hWnd  类型：**HWND**  要检索其消息的窗口的句柄。 窗口必须属于当前线程。  如果 *hWnd* 为 **NULL，GetMessage** 将检索属于当前线程的任何窗口的消息，以及当前线程的消息队列中 **hwnd** 值为 **NULL** 的任何消息， () 看到 [MSG](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/ns-winuser-msg) 结构。 因此，如果 hWnd 为 **NULL**，则同时处理窗口消息和线程消息。  如果 *hWnd* 为 -1，**则 GetMessage** 仅检索当前线程的消息队列中 **hwnd** 值为 **NULL** 的消息，即当 *hWnd* 参数为 **NULL**) 或 [PostThreadMessage](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-postthreadmessagea) 时，[PostMessage](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-postmessagea) (发布的线程消息。  [in] wMsgFilterMin  类型： **UINT**  要检索的最低消息值的整数值。 使用 **WM\_KEYFIRST** (0x0100) 指定第一条键盘消息， **或使用WM\_MOUSEFIRST** (0x0200) 指定第一条鼠标消息。  在此处和 *wMsgFilterMax* 中使用[WM\_INPUT](https://learn.microsoft.com/zh-cn/windows/desktop/inputdev/wm-input)仅指定**WM\_INPUT**消息。  如果 *wMsgFilterMin* 和 *wMsgFilterMax* 均为零， **则 GetMessage** 将返回所有可用消息 (即不) 执行范围筛选。  [in] wMsgFilterMax  类型： **UINT**  要检索的最高消息值的整数值。 使用 **WM\_KEYLAST** 指定最后一条键盘消息， **WM\_MOUSELAST** 指定最后一条鼠标消息。  在此处和 *wMsgFilterMin* 中使用[WM\_INPUT](https://learn.microsoft.com/zh-cn/windows/desktop/inputdev/wm-input)，仅指定**WM\_INPUT**消息。  如果 *wMsgFilterMin* 和 *wMsgFilterMax* 均为零， **则 GetMessage** 将返回所有可用消息 (即不) 执行范围筛选。 返回值 类型： **BOOL**  如果函数检索 [WM\_QUIT](https://learn.microsoft.com/zh-cn/windows/desktop/winmsg/wm-quit)以外的消息，则返回值为非零值。  如果函数检索 [WM\_QUIT](https://learn.microsoft.com/zh-cn/windows/desktop/winmsg/wm-quit) 消息，则返回值为零。  如果出现错误，则返回值为 -1。 例如，如果 *hWnd* 是无效的窗口句柄或 *lpMsg* 是无效的指针，则该函数将失败。 要获得更多的错误信息，请调用 GetLastError。  由于返回值可以是非零、零或 -1，因此请避免使用如下所示的代码：  复制  while (GetMessage( lpMsg, hWnd, 0, 0)) ...  如果 hWnd 是无效参数 (（例如引用已销毁) 窗口），则返回值可能为 -1，这意味着此类代码可能会导致严重的应用程序错误。 请改用如下所示的代码：  复制  BOOL bRet;  while( (bRet = GetMessage( &msg, hWnd, 0, 0 )) != 0)  {  if (bRet == -1)  {  // handle the error and possibly exit  }  else  {  TranslateMessage(&msg);  DispatchMessage(&msg);  }  } 注解 应用程序通常使用返回值来确定是否结束main消息循环并退出程序。  **GetMessage** 函数检索与*由 hWnd* 参数或其任何子级标识的窗口关联的消息（由 [IsChild](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-ischild) 函数指定），并在 *wMsgFilterMin* 和 *wMsgFilterMax* 参数给出的消息值范围内。 请注意，应用程序只能使用 *wMsgFilterMin* 和 *wMsgFilterMax* 参数中的低字;高字是为系统保留的。  请注意，无论为 *wMsgFilterMin 和 wMsgFilterMax* 指定哪些值，**GetMessage** 始终检索[WM\_QUIT](https://learn.microsoft.com/zh-cn/windows/desktop/winmsg/wm-quit)消息。  在此调用期间，系统将传送挂起的非排队消息，即使用 [SendMessage、SendMessageCallback](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-sendmessage)、[SendMessageTimeout](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-sendmessagetimeouta) 或 [SendNotifyMessage](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-sendnotifymessagea) 函数发送到调用线程拥有的窗口的消息。 然后检索与指定筛选器匹配的第一个排队消息。 系统还可以处理内部事件。 如果未指定筛选器，则按以下顺序处理消息：   * 已发送消息 * 已发布的消息 * 输入 (硬件) 消息和系统内部事件 * 再次 () 发送消息 * [WM\_PAINT](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/wm-paint) 消息 * [WM\_TIMER](https://learn.microsoft.com/zh-cn/windows/desktop/winmsg/wm-timer) 消息   若要在发布消息之前检索输入消息，请使用 *wMsgFilterMin* 和 *wMsgFilterMax* 参数。  **GetMessage** 不会从队列中删除 [WM\_PAINT](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/wm-paint) 消息。 消息一直保留在队列中，直到处理。  如果顶级窗口停止响应消息超过几秒钟，系统会认为窗口没有响应，并将其替换为具有相同 z 顺序、位置、大小和视觉属性的虚影窗口。 这允许用户移动它、调整其大小，甚至关闭应用程序。 但是，这些是唯一可用的操作，因为应用程序实际上没有响应。 在调试器模式下，系统不会生成虚影窗口。 DPI 虚拟化 此 API 不参与 DPI 虚拟化。 输出处于消息所面向的窗口的模式。 不考虑调用线程。 示例 有关示例，请参阅 [创建消息循环](https://learn.microsoft.com/zh-cn/windows/desktop/winmsg/using-messages-and-message-queues)。 要求  |  |  | | --- | --- | | **标头** | winuser.h (包括 Windows.h) | | **Library** | User32.lib | | **DLL** | User32.dll | | **API 集** | 在 Windows 8) 中引入的 ext-ms-win-ntuser-message-l1-1-0 ( |  请参阅 **概念性**  [IsChild](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-ischild)  [MSG](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/ns-winuser-msg)  [消息和消息队列](https://learn.microsoft.com/zh-cn/windows/desktop/winmsg/messages-and-message-queues)  [PeekMessage](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-peekmessagea)  [PostMessage](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-postmessagea)  [PostThreadMessage](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-postthreadmessagea)  **引用**  [WaitMessage](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-waitmessage) |

## LoadBitmap函数

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**LoadBitmap** 可用于“要求”部分中指定的操作系统。 它可能在后续版本中变更或不可用。 请改用 [LoadImage](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-loadimagea) 和 [DrawFrameControl](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-drawframecontrol)。]  **LoadBitmap** 函数从模块的可执行文件加载指定的位图资源。 语法 C++  HBITMAP LoadBitmapA(  [in] HINSTANCE hInstance,  [in] LPCSTR lpBitmapName  ); 参数 [in] hInstance  模块实例的句柄，该模块的可执行文件包含要加载的位图。  [in] lpBitmapName  指向以 null 结尾的字符串的指针，该字符串包含要加载的位图资源的名称。 或者，此参数可以包含低序字中的资源标识符和高序字中的零。 [MAKEINTRESOURCE](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-makeintresourcea) 宏可用于创建此值。 返回值 如果函数成功，则返回值是指定位图的句柄。  如果函数失败，则返回值为 NULL。 注解 如果 *lpBitmapName* 参数指向的位图不存在或内存不足，无法加载位图，则函数将失败。  **LoadBitmap** 创建显示器的兼容位图，该位图不能在打印机上选择。 若要将可以选择的位图加载到打印机，请调用 [LoadImage](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-loadimagea) 并指定LR\_CREATEDIBSECTION以创建 DIB 节。 可以为任何设备选择 DIB 部分。  应用程序可以使用 **LoadBitmap** 函数访问预定义的位图。 为此，应用程序必须将 *hInstance* 参数设置为 **NULL** ，并将 *lpBitmapName* 参数设置为以下值之一。  展开表   |  |  | | --- | --- | | **位图名称** | **位图名称** | | OBM\_BTNCORNERS | OBM\_OLD\_RESTORE | | OBM\_BTSIZE | OBM\_OLD\_RGARROW | | OBM\_CHECK | OBM\_OLD\_UPARROW | | OBM\_CHECKBOXES | OBM\_OLD\_ZOOM | | OBM\_CLOSE | OBM\_REDUCE | | OBM\_COMBO | OBM\_REDUCED | | OBM\_DNARROW | OBM\_RESTORE | | OBM\_DNARROWD | OBM\_RESTORED | | OBM\_DNARROWI | OBM\_RGARROW | | OBM\_LFARROW | OBM\_RGARROWD | | OBM\_LFARROWD | OBM\_RGARROWI | | OBM\_LFARROWI | OBM\_SIZE | | OBM\_MNARROW | OBM\_UPARROW | | OBM\_OLD\_CLOSE | OBM\_UPARROWD | | OBM\_OLD\_DNARROW | OBM\_UPARROWI | | OBM\_OLD\_LFARROW | OBM\_ZOOM | | OBM\_OLD\_REDUCE | OBM\_ZOOMD |     以 OBM\_OLD 开头的位图名称表示早于 3.0 的 16 位版本的 Windows 使用的位图。  若要使应用程序使用任何OBM\_常量，必须在包含 Windows.h 头文件之前定义常量 OEMRESOURCE。  应用程序必须调用 [DeleteObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-deleteobject) 函数才能删除 **LoadBitmap** 函数返回的每个位图句柄。 示例 有关示例，请参阅 [Using Menus](https://learn.microsoft.com/zh-cn/windows/desktop/menurc/using-menus) 中的 Menu-Item 位图示例。  **备注**  winuser.h 标头将 LoadBitmap 定义为别名，该别名根据 UNICODE 预处理器常量的定义自动选择此函数的 ANSI 或 Unicode 版本。 将非特定编码别名与非非特定编码的代码混合使用可能会导致不匹配，从而导致编译或运行时错误。 有关详细信息，请参阅 [**函数原型的约定**](https://learn.microsoft.com/zh-cn/windows/win32/intl/conventions-for-function-prototypes)。 要求 展开表   | **要求** | **值** | | --- | --- | | **最低受支持的客户端** | Windows 2000 Professional [仅限桌面应用] | | **最低受支持的服务器** | Windows 2000 Server [仅限桌面应用] | | **目标平台** | Windows | | **标头** | winuser.h (包括 Windows.h) | | **Library** | User32.lib | | **DLL** | User32.dll | | **API 集** | 在 Windows 8.1) 中引入的 ext-ms-win-ntuser-draw-l1-1-1 ( |  另请参阅 [位图函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/bitmap-functions)  [位图概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/bitmaps)  [CreateBitmap](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createbitmap)  [DeleteObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-deleteobject)  [DrawFrameControl](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-drawframecontrol)  [LoadCursor](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-loadcursora)  [LoadIcon](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-loadicona)  [LoadImage](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-loadimagea)  [MAKEINTRESOURCE](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-makeintresourcea) |

## LoadIcon函数

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| LoadIconA 函数 (winuser.h)  * 项目 * 2024/03/04   反馈 本文内容  1. [语法](https://learn.microsoft.com/zh-cn/windows/win32/api/winuser/nf-winuser-loadicona#syntax) 2. [参数](https://learn.microsoft.com/zh-cn/windows/win32/api/winuser/nf-winuser-loadicona#parameters) 3. [返回值](https://learn.microsoft.com/zh-cn/windows/win32/api/winuser/nf-winuser-loadicona#return-value) 4. [注解](https://learn.microsoft.com/zh-cn/windows/win32/api/winuser/nf-winuser-loadicona#remarks)   显示另外 2 个  从与应用程序实例关联的可执行 (.exe) 文件加载指定的图标资源。  **备注**  此函数已被 [**LoadImage**](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-loadimagea) 函数 (取代，) 设置了 **LR\_DEFAULTSIZE** 和 **LR\_SHARED** 标志。 语法 C++复制  HICON LoadIconA(  [in, optional] HINSTANCE hInstance,  [in] LPCSTR lpIconName  ); 参数 [in, optional] hInstance  类型： **HINSTANCE**  DLL 或可执行文件的模块的句柄， (.exe) 包含要加载的图标的文件。 有关详细信息，请参阅 [GetModuleHandle](https://learn.microsoft.com/zh-cn/windows/desktop/api/libloaderapi/nf-libloaderapi-getmodulehandlea)。  若要加载预定义的系统图标，请将此参数设置为 **NULL**。  [in] lpIconName  类型： **LPCTSTR**  如果 *hInstance* 为非 **NULL**， *则 lpIconName* 按名称或序号指定图标资源。 必须使用 [MAKEINTRESOURCE](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-makeintresourcew) 宏打包此序号。  如果 *hInstance* 为 **NULL**， *则 lpIconName* 将指定标识符 (从要加载的预定义系统图标的 [IDI\_前缀) 开始](https://learn.microsoft.com/zh-cn/windows/win32/menurc/about-icons)。 返回值 类型： **HICON**  如果函数成功，则返回值是新加载图标的句柄。  如果函数失败，则返回值为 NULL。 要获得更多的错误信息，请调用 GetLastError。 注解 仅当图标资源尚未加载时，**LoadIcon** 才会加载该资源;否则，它将检索现有资源的句柄。 函数在图标资源中搜索最适合当前显示的图标。 图标资源可以是颜色或单色位图。  **LoadIcon** 只能加载大小符合 **SM\_CXICON** 和 **SM\_CYICON** 系统指标值的图标。 使用 [LoadImage](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-loadimagea) 函数加载其他大小的图标。  **备注**  winuser.h 标头将 LoadIcon 定义为别名，该别名根据 UNICODE 预处理器常量的定义自动选择此函数的 ANSI 或 Unicode 版本。 将非特定编码别名与非非特定编码的代码混合使用可能会导致不匹配，从而导致编译或运行时错误。 有关详细信息，请参阅 [**函数原型的约定**](https://learn.microsoft.com/zh-cn/windows/win32/intl/conventions-for-function-prototypes)。 要求  |  |  | | --- | --- | | **标头** | winuser.h (包括 Windows.h) | | **Library** | User32.lib | | **DLL** | User32.dll | | **API 集** | 在 Windows 8) 中引入的 ext-ms-win-ntuser-gui-l1-1-0 ( |  请参阅 **概念性**  [CreateIcon](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-createicon)  [图标](https://learn.microsoft.com/zh-cn/windows/desktop/menurc/icons)  [LoadImage](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-loadimagea)  [MAKEINTRESOURCE](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-makeintresourcea)  [IS\_INTRESOURCE](https://learn.microsoft.com/zh-cn/windows/win32/api/winuser/nf-winuser-is_intresource) |

## loadCursor函数

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 从与应用程序实例关联的可执行文件 (.EXE) 文件中加载指定的游标资源。  **备注**  此函数已被 [**LoadImage**](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-loadimagea) 函数 (取代，) 设置了 **LR\_DEFAULTSIZE** 和 **LR\_SHARED** 标志。 语法 C++复制  HCURSOR LoadCursorA(  [in, optional] HINSTANCE hInstance,  [in] LPCSTR lpCursorName  ); 参数 [in, optional] hInstance  类型： **HINSTANCE**  DLL 或可执行文件 (.exe 模块的句柄，) 包含要加载的游标的文件。 有关详细信息，请参阅 [GetModuleHandle](https://learn.microsoft.com/zh-cn/windows/desktop/api/libloaderapi/nf-libloaderapi-getmodulehandlea)。  若要加载预定义的系统游标，请将此参数设置为 **NULL**。  [in] lpCursorName  类型： **LPCTSTR**  如果 *hInstance* 为非 **NULL**， *则 lpCursorName* 按名称或序号指定游标资源。 必须使用 [MAKEINTRESOURCE](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-makeintresourcew) 宏打包此序号。  如果 *hInstance* 为 **NULL**， *则 lpCursorName* 将指定标识符 (从要加载的预定义系统游标的 [IDC\_前缀) 开始](https://learn.microsoft.com/zh-cn/windows/win32/menurc/about-cursors)。 返回值 类型： **HCURSOR**  如果函数成功，则返回值是新加载的游标的句柄。  如果函数失败，则返回值为 NULL。 要获得更多的错误信息，请调用 GetLastError。 注解 仅当游标资源尚未加载时， **LoadCursor** 函数才会加载游标资源;否则，它将检索现有资源的句柄。 仅当 *lpCursorName* 参数是指向游标资源的指针时，此函数才返回有效的游标句柄。 如果 *lpCursorName* 是指向除游标 (（例如图标) ）以外的任何类型的资源的指针，则返回值不是 **NULL**，即使它不是有效的游标句柄也是如此。  **LoadCursor** 函数搜索最适合当前显示设备的游标资源。 光标资源可以是颜色位图或单色位图。 DPI 虚拟化 此 API 不参与 DPI 虚拟化。 返回的输出不受调用线程的 DPI 影响。 示例 有关示例，请参阅 [创建游标](https://learn.microsoft.com/zh-cn/windows/desktop/menurc/using-cursors)。  **备注**  winuser.h 标头将 LoadCursor 定义为别名，该别名根据 UNICODE 预处理器常量的定义自动选择此函数的 ANSI 或 Unicode 版本。 将非特定编码别名的使用与非非特定编码的代码混合使用可能会导致不匹配，从而导致编译或运行时错误。 有关详细信息，请参阅 [**函数原型的约定**](https://learn.microsoft.com/zh-cn/windows/win32/intl/conventions-for-function-prototypes)。 要求  |  |  | | --- | --- | | **标头** | winuser.h (包括 Windows.h) | | **Library** | User32.lib | | **DLL** | User32.dll |  请参阅 **概念性**  [光标](https://learn.microsoft.com/zh-cn/windows/desktop/menurc/cursors)  [LoadImage](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-loadimagea)  [MAKEINTRESOURCE](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-makeintresourcea)  [IS\_INTRESOURCE](https://learn.microsoft.com/zh-cn/windows/win32/api/winuser/nf-winuser-is_intresource)  **引用**  [SetCursor](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-setcursor)  [SetCursorPos](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-setcursorpos)  [ShowCursor](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-showcursor) |

## LoadImage函数,可以代替上面这几个函数

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 加载图标、光标、动画光标或位图。 语法 C++复制  HANDLE LoadImageA(  [in, optional] HINSTANCE hInst,  [in] LPCSTR name,  [in] UINT type,  [in] int cx,  [in] int cy,  [in] UINT fuLoad  ); 参数 [in, optional] hInst  类型： **HINSTANCE**  包含要加载的图像的 DLL 或可执行文件 (.exe) 的模块的句柄。 有关详细信息，请参阅 [GetModuleHandle](https://learn.microsoft.com/zh-cn/windows/desktop/api/libloaderapi/nf-libloaderapi-getmodulehandlea)。 请注意，从 32 位 Windows 开始，实例句柄 (**HINSTANCE**) ，例如 [由 WinMain](https://learn.microsoft.com/zh-cn/windows/desktop/api/winbase/nf-winbase-winmain) 的系统函数调用公开的应用程序实例句柄，而模块句柄 (**HMODULE**) 也是如此。  若要 (图标、光标或位图文件) 加载预定义图像或独立资源，请将此参数设置为 **NULL**。  [in] name  类型： **LPCTSTR**  要加载的图像。  如果 *hInst* 参数为非 **NULL** 且 *fuLoad* 参数省略 **LR\_LOADFROMFILE**， *则 name* 指定 *hInst* 模块中的图像资源。  如果要按名称从模块加载图像资源， *则 name* 参数是指向包含映像资源名称的以 null 结尾的字符串的指针。  如果要从模块按序号加载图像资源，请使用 [MAKEINTRESOURCE](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-makeintresourcea) 宏将图像序号转换为可传递给 **LoadImage** 函数的形式。  如果 *hInst* 参数为 **NULL** 且 *fuLoad* 参数省略 **LR\_LOADFROMFILE** 值并包含 **LR\_SHARED**，则 *名称* 将指定要加载的预定义图像。  预定义的图像标识符在 中 Winuser.h 定义，并具有以下前缀：  展开表   | **前缀** | **含义** | | --- | --- | | **OBM\_** | OEM 位图。 使用 [MAKEINTRESOURCE](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-makeintresourcea) 宏传递这些。 | | **OIC\_** | OEM 图标。 使用 [MAKEINTRESOURCE](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-makeintresourcea) 宏传递这些。 | | **Ocr\_** | OEM 游标。 使用 [MAKEINTRESOURCE](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-makeintresourcea) 宏传递这些。 | | **IDI\_** | [标准图标](https://learn.microsoft.com/zh-cn/windows/win32/menurc/about-icons) | | **Idc\_** | [标准游标](https://learn.microsoft.com/zh-cn/windows/win32/menurc/about-cursors) |   若要将 OEM 图像标识符常量传递给 **LoadImage** 函数，请使用 [MAKEINTRESOURCE](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-makeintresourcea) 宏。 例如，若要加载**OCR\_NORMAL**游标，请将 作为 *name* 参数传递MAKEINTRESOURCE(OCR\_NORMAL)，**将 NULL** 作为 *hInst* 参数传递，**并将LR\_SHARED**作为标志之一传递给 *fuLoad* 参数。  如果 *hInst* 参数为 **NULL** 且 *fuLoad* 参数包含**LR\_LOADFROMFILE**值，*则 name* 是包含独立资源 (图标、光标或位图文件) 的文件的名称，例如 。 c:\myicon.ico  有关详细信息，请参阅下面的“备注”部分。  [in] type  类型： **UINT**  要加载的图像的类型。  此参数可能是以下值之一：  展开表   | **值** | **含义** | | --- | --- | | **IMAGE\_BITMAP** | 加载位图。 | | **IMAGE\_CURSOR** | 加载游标。 | | **IMAGE\_ICON** | 加载图标。 |   [in] cx  类型： **int**  图标或光标的宽度（以像素为单位）。 如果此参数为零且 *fuLoad* 参数 **为LR\_DEFAULTSIZE**，则函数使用 **SM\_CXICON** 或 **SM\_CXCURSOR** 系统指标值来设置宽度。 如果此参数为零且未使用 **LR\_DEFAULTSIZE** ，则函数使用实际资源宽度。  [in] cy  类型： **int**  图标或光标的高度（以像素为单位）。 如果此参数为零且 *fuLoad* 参数 **为LR\_DEFAULTSIZE**，则函数使用 **SM\_CYICON** 或 **SM\_CYCURSOR** 系统指标值来设置高度。 如果此参数为零且未使用 **LR\_DEFAULTSIZE** ，则函数使用实际资源高度。  [in] fuLoad  类型： **UINT**  此参数可使用以下一个或多个值。  展开表   |  |  | | --- | --- | | **值** | **含义** | | **LR\_CREATEDIBSECTION**  0x00002000 | 当 *uType* 参数指定 **IMAGE\_BITMAP**时，会导致函数返回 DIB 节位图而不是兼容的位图。 此标志可用于加载位图而不将其映射到显示设备的颜色。 | | **LR\_DEFAULTCOLOR**  0x00000000 | 默认标志;它不执行任何工作。 它的意思是“不 **LR\_MONOCHROME**”。 | | **LR\_DEFAULTSIZE**  0x00000040 | 如果 *cxDesired* 或 *cyDesired* 值设置为零，则使用游标或图标的系统指标值指定的宽度或高度。 如果未指定此标志，并且 *cxDesired* 和 *cyDesired* 设置为零，则函数将使用实际资源大小。 如果资源包含多个图像，则 函数使用第一个图像的大小。 | | **LR\_LOADFROMFILE**  0x00000010 | 从 *名称* (图标、光标或位图文件指定的文件) 加载独立图像。 | | **LR\_LOADMAP3DCOLORS**  0x00001000 | 在颜色表中搜索图像，并将以下灰色底纹替换为相应的三维颜色。   * Dk 灰色，RGB (128,128,128) 与 **COLOR\_3DSHADOW** * 灰色，RGB (192,192,192) ，带 **COLOR\_3DFACE** * Lt Gray，RGB (223,223,223) 与 **COLOR\_3DLIGHT**   如果要加载颜色深度大于 8bpp 的位图，请不要使用此选项。 | | **LR\_LOADTRANSPARENT**  0x00000020 | 检索图像中第一个像素的颜色值，并将颜色表中的相应条目替换为默认窗口颜色 (**COLOR\_WINDOW**) 。 图像中使用该条目的所有像素都将成为默认的窗口颜色。 此值仅适用于具有相应颜色表的图像。  如果要加载颜色深度大于 8bpp 的位图，请不要使用此选项。  如果 *fuLoad* 同时包含 **LR\_LOADTRANSPARENT** 值和 **LR\_LOADMAP3DCOLORS** 值， **LR\_LOADTRANSPARENT** 优先。 但是，颜色表条目将替换为 **COLOR\_3DFACE** 而不是 **COLOR\_WINDOW**。 | | **LR\_MONOCHROME**  0x00000001 | 加载黑白图像。 | | **LR\_SHARED**  0x00008000 | 如果多次加载映像，则共享映像句柄。 如果未设置 **LR\_SHARED** ，则对同一资源的第二次 **LoadImage** 调用将再次加载映像并返回不同的句柄。  使用此标志时，系统会在不再需要资源时销毁该资源。  对于具有非标准大小、加载后可能会更改或从文件加载的图像，请勿使用 **LR\_SHARED** 。  加载系统图标或光标时，必须使用 **LR\_SHARED** 否则函数将无法加载资源。  无论请求的大小如何，此函数都会查找缓存中具有请求的资源名称的第一个映像。 | | **LR\_VGACOLOR**  0x00000080 | 使用真正的 VGA 颜色。 |  返回值 类型： **HANDLE**  如果函数成功，则返回值是新加载的图像的句柄。  如果函数失败，则返回值为 NULL。 要获得更多的错误信息，请调用 GetLastError。 注解 如果 [IS\_INTRESOURCE](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-is_intresource) (*名称*) 为 **TRUE**，则 *name* 指定给定资源的整数标识符。 否则，它是指向以 null 结尾的字符串的指针。 如果字符串的第一个字符是井号 (#) ，则其余字符表示一个指定资源的整数标识符的十进制数。 例如，字符串“#258”表示标识符 258。  使用完未指定 **LR\_SHARED** 标志的情况下加载的位图、光标或图标后，可以通过调用下表中的函数之一释放其关联的内存。  展开表   |  |  | | --- | --- | | **资源** | **Release 函数** | | Bitmap | [DeleteObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-deleteobject) | | 游标 | [DestroyCursor](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-destroycursor) | | 图标 | [DestroyIcon](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-destroyicon) |     当创建这些资源的进程终止时，系统会自动删除这些资源;但是，调用相应的函数可节省内存并减小进程的工作集的大小。 示例 有关示例，请参阅 [使用窗口类](https://learn.microsoft.com/zh-cn/windows/desktop/winmsg/using-window-classes)。  **备注**  winuser.h 标头将 LoadImage 定义为别名，该别名根据 UNICODE 预处理器常量的定义自动选择此函数的 ANSI 或 Unicode 版本。 将非特定编码别名与非非特定编码的代码混合使用可能会导致不匹配，从而导致编译或运行时错误。 有关详细信息，请参阅 [**函数原型的约定**](https://learn.microsoft.com/zh-cn/windows/win32/intl/conventions-for-function-prototypes)。 要求  |  |  | | --- | --- | | **标头** | winuser.h (包括 Windows.h) | | **Library** | User32.lib | | **DLL** | User32.dll | | **API 集** | windows 8 中引入的 ext-ms-win-ntuser-gui-l1-1-0 () |  请参阅 **概念性**  [CopyImage](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-copyimage)  [GetSystemMetrics](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-getsystemmetrics)  [LoadBitmap](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-loadbitmapa)  [LoadCursor](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-loadcursora)  [LoadIcon](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-loadicona)  **其他资源**  **引用**  [资源](https://learn.microsoft.com/zh-cn/windows/desktop/menurc/resources) |

## GetStockObject函数

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **GetStockObject** 函数检索其中一支股票笔、画笔、字体或调色板的句柄。 语法 C++  HGDIOBJ GetStockObject(  [in] int i  ); 参数 [in] i  常用对象的类型。 此参数的取值可为下列值之一：  展开表   |  |  | | --- | --- | | **值** | **含义** | | **BLACK\_BRUSH** | 黑色画笔。 | | **DKGRAY\_BRUSH** | 深灰色画笔。 | | **DC\_BRUSH** | 纯色画笔。 默认颜色为白色。 可以使用 [SetDCBrushColor](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-setdcbrushcolor) 函数更改颜色。 有关详细信息，请参见“备注”部分。 | | **GRAY\_BRUSH** | 灰色画笔。 | | **HOLLOW\_BRUSH** | 空心画笔 (等效于NULL\_BRUSH) 。 | | **LTGRAY\_BRUSH** | 浅灰色画笔。 | | **NULL\_BRUSH** | null 画笔 (等效于 HOLLOW\_BRUSH) 。 | | **WHITE\_BRUSH** | 白色画笔。 | | **BLACK\_PEN** | 黑色触笔。 | | **DC\_PEN** | 纯色笔颜色。 默认颜色为黑色。 可以使用 [SetDCPenColor](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-setdcpencolor) 函数更改颜色。 有关详细信息，请参见“备注”部分。 | | **NULL\_PEN** | null 触笔。 null 触控笔不绘制任何内容。 | | **WHITE\_PEN** | 白色触笔。 | | **ANSI\_FIXED\_FONT** | Windows 固定间距 (正) 系统字体。 | | **ANSI\_VAR\_FONT** | Windows 可变间距 (比例空间) 系统字体。 | | **DEVICE\_DEFAULT\_FONT** | 设备依赖字体。 | | **DEFAULT\_GUI\_FONT** | 用户界面对象（如菜单和对话框）的默认字体。 不建议使用DEFAULT\_GUI\_FONT或SYSTEM\_FONT来获取对话框和窗口使用的字体;有关详细信息，请参阅备注部分。  默认字体为 Tahoma。 | | **OEM\_FIXED\_FONT** | 原始设备制造商 (OEM) 依赖固定间距 (正方形) 字体。 | | **SYSTEM\_FONT** | 系统字体。 默认情况下，系统使用系统字体来绘制菜单、对话框控件和文本。 不建议使用DEFAULT\_GUI\_FONT或SYSTEM\_FONT来获取对话框和窗口使用的字体;有关详细信息，请参阅备注部分。  默认系统字体为 Tahoma。 | | **SYSTEM\_FIXED\_FONT** | 固定间距 (单调) 系统字体。 此库存对象仅用于与 3.0 之前的 16 位 Windows 版本兼容。 | | **DEFAULT\_PALETTE** | 默认调色板。 此调色板由系统调色板中的静态颜色组成。 |  返回值 如果函数成功，则返回值是请求的逻辑对象的句柄。  如果函数失败，则返回值为 NULL。 注解 不建议使用此方法来获取对话框和窗口使用的当前字体。 请改用带有 SPI\_GETNONCLIENTMETRICS 参数的 [SystemParametersInfo](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-systemparametersinfoa) 函数来检索当前字体。 [SystemParametersInfo](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-systemparametersinfoa) 将考虑当前主题，并提供标题、菜单和消息对话框的字体信息。  仅在具有CS\_HREDRAW和CS\_VREDRAW样式的窗口中使用DKGRAY\_BRUSH、GRAY\_BRUSH和LTGRAY\_BRUSH库存对象。 在任何其他窗口样式中使用灰色股票画笔可能会导致在移动窗口或调整大小后画笔图案不对齐。 无法调整股票画笔的来源。  HOLLOW\_BRUSH和NULL\_BRUSH库存对象是等效的。  不需要 (但) 调用 [DeleteObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-deleteobject) 删除库存对象并不有害。  DC\_BRUSH和DC\_PEN都可以与其他库存对象（如BLACK\_BRUSH和BLACK\_PEN）互换使用。 有关检索当前笔或画笔颜色的信息，请参阅 [GetDCBrushColor](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getdcbrushcolor) 和 [GetDCPenColor](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getdcpencolor)。 有关设置颜色的示例，请参阅 [设置笔或画笔](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/setting-the-pen-or-brush-color) 颜色。 参数为 DC\_BRUSH 或 DC\_PEN 的 **GetStockObject** 函数可与 [SetDCPenColor](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-setdcpencolor) 和 [SetDCBrushColor](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-setdcbrushcolor) 函数互换使用。 示例 有关示例，请参阅 [设置笔或画笔颜色](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/setting-the-pen-or-brush-color)。 要求  |  |  | | --- | --- | | **标头** | wingdi.h (包括 Windows.h) | | **Library** | Gdi32.lib | | **DLL** | Gdi32.dll |  另请参阅 [DeleteObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-deleteobject)  [设备上下文函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/device-context-functions)  [设备上下文概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/device-contexts)  [SelectObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-selectobject) |

## CreateSolidBrush函数

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **CreateSolidBrush** 函数创建具有指定纯色的逻辑画笔。 语法 C++复制  HBRUSH CreateSolidBrush(  [in] COLORREF color  ); 参数 [in] color  画笔的颜色。 若要创建 [COLORREF](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/colorref) 颜色值，请使用 [RGB](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-rgb) 宏。 返回值 如果函数成功，则返回值将标识逻辑画笔。  如果函数失败，则返回值为 NULL。 注解 如果不再需要 **HBRUSH** 对象，请调用 [DeleteObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-deleteobject) 函数将其删除。  实心画笔是系统用于绘制填充形状内部的位图。  应用程序通过调用 **CreateSolidBrush** 创建画笔后，可以通过调用 [SelectObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-selectobject) 函数将画笔选择到任何设备上下文中。  若要使用系统颜色画笔进行绘制，应用程序应使用 GetSysColorBrush (nIndex) 而不是 CreateSolidBrush(GetSysColor(nIndex))，因为 [GetSysColorBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-getsyscolorbrush) 返回缓存的画笔，而不是分配新的画笔。  **Icm：** 创建画笔时不执行颜色管理。 但是，将画笔选入启用了 ICM 的设备上下文时，将执行颜色管理。 示例 有关示例，请参阅 [创建彩色笔和画笔](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/creating-colored-pens-and-brushes)。 要求  |  |  | | --- | --- | | **标头** | wingdi.h (包括 Windows.h) | | **Library** | Gdi32.lib | | **DLL** | Gdi32.dll |  另请参阅 [画笔函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/brush-functions)  [画笔概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/brushes)  [COLORREF](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/colorref)  [CreateDIBPatternBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createdibpatternbrush)  [CreateDIBPatternBrushPt](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createdibpatternbrushpt)  [CreateHatchBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createhatchbrush)  [CreatePatternBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createpatternbrush)  [DeleteObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-deleteobject)  [GetSysColorBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-getsyscolorbrush)  [RGB](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-rgb)  [SelectObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-selectobject) |

## CreatePatternBrush 函数

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **CreatePatternBrush** 函数使用指定的位图模式创建逻辑画刷。 位图可以是由 **CreateDIBSection** 函数创建的 DIB 节位图，也可以是依赖于设备的位图。 语法 C++  HBRUSH CreatePatternBrush(  [in] HBITMAP hbm  ); 参数 [in] hbm  要用于创建逻辑画笔的位图的句柄。 返回值 如果函数成功，则返回值标识逻辑画笔。  如果函数失败，则返回值为 NULL。 注解 图案画笔是系统用于绘制填充形状内部的位图。  应用程序通过调用 **CreatePatternBrush** 创建画笔后，可以通过调用 [SelectObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-selectobject) 函数在任何设备上下文中选择该画笔。  可以使用 [DeleteObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-deleteobject) 函数删除图案画笔，而不会影响关联的位图。 因此，可以使用此位图创建任意数量的图案画笔。  使用单色 (每像素 1 位) 位图创建的画笔具有绘制到的设备上下文的文本和背景色。 用当前文本颜色绘制由 0 位表示的像素;用当前背景色绘制由 1 位表示的像素。  **Icm：** 创建画笔时不执行任何颜色。 但是，当画笔被选入启用了 ICM 的设备上下文时，将执行颜色管理。 示例 有关示例，请参阅 [使用画笔](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/using-brushes)。 要求  |  |  | | --- | --- | | **标头** | wingdi.h (包括 Windows.h) | | **Library** | Gdi32.lib | | **DLL** | Gdi32.dll |  另请参阅 [画笔函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/brush-functions)  [画笔概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/brushes)  [CreateBitmap](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createbitmap)  [CreateBitmapIndirect](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createbitmapindirect)  [CreateCompatibleBitmap](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createcompatiblebitmap)  [CreateDIBPatternBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createdibpatternbrush)  [CreateDIBPatternBrushPt](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createdibpatternbrushpt)  [CreateDIBSection](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createdibsection)  [CreateHatchBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createhatchbrush)  [DeleteObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-deleteobject)  [GetBrushOrgEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getbrushorgex)  [LoadBitmap](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-loadbitmapa)  [SelectObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-selectobject)  [SetBrushOrgEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-setbrushorgex) |

## CreateHatchBrush 函数

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CreateHatchBrush** 函数创建具有指定阴影图案和颜色的逻辑画笔。 语法 C++复制  HBRUSH CreateHatchBrush(  [in] int iHatch,  [in] COLORREF color  ); 参数 [in] iHatch  [画笔的阴影样式](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/hatch-brush)。 此参数的取值可为下列值之一：  展开表   |  |  | | --- | --- | | **值** | **含义** | | **HS\_BDIAGONAL** | 从左到右阴影向上 45 度 | | **HS\_CROSS** | 横向缩放与纵向阴影线 | | **HS\_DIAGCROSS** | 45 度交叉截断 | | **HS\_FDIAGONAL** | 从左到右阴影下 45 度 | | **HS\_HORIZONTAL** | 水平阴影 | | **HS\_VERTICAL** | 垂直阴影 |   [in] color  用于阴影的画笔的前景色。 若要创建 [COLORREF](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/colorref) 颜色值，请使用 [RGB](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-rgb) 宏。 返回值 如果函数成功，则返回值将标识逻辑画笔。  如果函数失败，则返回值为 NULL。 注解 画笔是系统用于绘制填充形状内部的位图。  应用程序通过调用 **CreateHatchBrush** 创建画笔后，可以通过调用 [SelectObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-selectobject) 函数将画笔选择到任何设备上下文中。 它还可以调用 [SetBkMode](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-setbkmode) 来影响画笔的呈现。  如果应用程序使用阴影画笔以匹配的颜色填充父窗口和子窗口的背景，则必须在绘制子窗口的背景之前设置画笔原点。 可以通过调用 [SetBrushOrgEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-setbrushorgex) 函数来执行此操作。 应用程序可以通过调用 [GetBrushOrgEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getbrushorgex) 函数来检索当前画笔原点。  如果不再需要画笔，请调用 [DeleteObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-deleteobject) 函数将其删除。  **Icm：** 创建画笔时未定义任何颜色。 但是，将画笔选入启用了 ICM 的设备上下文时，将执行颜色管理。 示例 以下示例创建具有指定阴影图案和颜色的逻辑画笔。 还可以将阴影画笔背景设置为透明或不透明。  C++复制  #include <windows.h>  #include <stdlib.h>  #include <string.h>  #include <tchar.h>  #include <stddef.h>  #include <gdiplus.h>  #include <assert.h>  using namespace Gdiplus;  // Reference to the GDI+ static library).  #pragma comment (lib,"Gdiplus.lib")  // Global variables  // The main window class name.  static TCHAR szWindowClass[] = \_T("win32app");  // The string that appears in the application's title bar.  static TCHAR szTitle[] = \_T("Win32 Application Hatch Brush");  HINSTANCE hInst;  #define BTN\_MYBUTTON\_ID\_1 503  #define BTN\_MYBUTTON\_ID\_2 504  // Forward declarations of functions included in this code module:  LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);  int WINAPI WinMain(HINSTANCE hInstance,  HINSTANCE hPrevInstance,  LPSTR lpCmdLine,  int nCmdShow)  {  UNREFERENCED\_PARAMETER(lpCmdLine);  UNREFERENCED\_PARAMETER(hPrevInstance);  WNDCLASSEX wcex;  wcex.cbSize = sizeof(WNDCLASSEX);  wcex.style = CS\_HREDRAW | CS\_VREDRAW;  wcex.lpfnWndProc = WndProc;  wcex.cbClsExtra = 0;  wcex.cbWndExtra = 0;  wcex.hInstance = hInstance;  wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_APPLICATION));  wcex.hCursor = LoadCursor(NULL, IDC\_ARROW);  wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW+1);  wcex.lpszMenuName = NULL;  wcex.lpszClassName = szWindowClass;  wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_APPLICATION));  if (!RegisterClassEx(&wcex))  {  MessageBox(NULL,  \_T("Call to RegisterClassEx failed!"),  \_T("Win32 Guided Tour"),  NULL);  return 1;  }  hInst = hInstance; // Store instance handle in our global variable  // The parameters to CreateWindow:  // szWindowClass: the name of the application  // szTitle: the text that appears in the title bar  // WS\_OVERLAPPEDWINDOW: the type of window to create  // CW\_USEDEFAULT, CW\_USEDEFAULT: initial position (x, y)  // 500, 100: initial size (width, length)  // NULL: the parent of this window  // NULL: this application does not have a menu bar  // hInstance: the first parameter from WinMain  // NULL: not used in this application  HWND hWnd = CreateWindow(  szWindowClass,  szTitle,  WS\_OVERLAPPEDWINDOW,  CW\_USEDEFAULT, CW\_USEDEFAULT,  1000, 500,  NULL,  NULL,  hInstance,  NULL  );  if (!hWnd)  {  MessageBox(NULL,  \_T("Call to CreateWindow failed!"),  \_T("Win32 Guided Tour"),  NULL);  return 1;  }  // Create button controls.  CreateWindowEx(NULL, L"BUTTON", L"Transparent", WS\_VISIBLE | WS\_CHILD,  35, 35, 120, 20, hWnd, (HMENU)BTN\_MYBUTTON\_ID\_1, NULL, NULL);  CreateWindowEx(NULL, L"BUTTON", L"Opaque", WS\_VISIBLE | WS\_CHILD,  35, 65, 120, 20, hWnd, (HMENU)BTN\_MYBUTTON\_ID\_2, NULL, NULL);  // The parameters to ShowWindow:  // hWnd: the value returned from CreateWindow  // nCmdShow: the fourth parameter from WinMain  ShowWindow(hWnd,  nCmdShow);  UpdateWindow(hWnd);  // Main message loop:  MSG msg;  while (GetMessage(&msg, NULL, 0, 0))  {  TranslateMessage(&msg);  DispatchMessage(&msg);  }  return (int) msg.wParam;  }  /\*\*\*  \* This function creates the following rectangles:  \* 1. An outer rectangle using a solid brush with blue background.  \* 2. An inner rectangle using a hatch brush with red horizontal lines and yellow background.  \* It makes the background of the inner rectangle transparent or opaque in function of the user's input.  \* Inputs:  \* 1. hdc, the display device context.  \* 2. transparent, the hatch brush background user's value; true if transparent, false if opaque.  \*\*\*/  VOID SetHatchBrushBackground(HDC hdc, bool transparent)  {  // Define a brush handle.  HBRUSH hBrush;  // Create a solid blue brush.  hBrush = CreateSolidBrush (RGB(0, 0, 255));  // Associate the brush with the display device context.  SelectObject (hdc, hBrush);  // Draw a rectangle with blue background.  Rectangle (hdc, 400,40,800,400);  // Create a hatch brush that draws horizontal red lines.  hBrush = CreateHatchBrush(HatchStyleHorizontal, RGB(255, 0, 0));  // Set the background color to yellow.  SetBkColor(hdc, RGB(255, 255, 0));  // Select the hatch brush background transparency based on user's input.  if (transparent == true)  // Make the hatch brush background transparent.  // This displays the outer rectangle blue background.  SetBkMode(hdc, TRANSPARENT);  else  // Make the hatch brush background opaque.  // This displays the inner rectangle yellow background.  SetBkMode(hdc, OPAQUE);  // Associate the hatch brush with the current device context.  SelectObject(hdc, hBrush);  // Draw a rectangle with the specified hatch brush.  Rectangle(hdc, 500,130,700,300);  }  //  // FUNCTION: WndProc(HWND, UINT, WPARAM, LPARAM)  //  // PURPOSE: Processes messages for the main window.  //  // WM\_PAINT - Paint the main window  // WM\_DESTROY - post a quit message and return  //  //  LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)  {  PAINTSTRUCT ps;  HDC hdc;  TCHAR greeting[] = \_T("Select your brush background.");  TCHAR wmId;  TCHAR wmEvent;  switch (message)  {  case WM\_PAINT:  hdc = BeginPaint(hWnd, &ps);  // Start application-specific layout section.  // Just print the greeting string in the top left corner.  TextOut(hdc,  5, 5,  greeting, (int)\_tcslen(greeting));  // End application-specific layout section.  // Draw rectangles using hatch brush.  SetHatchBrushBackground(hdc, true);  EndPaint(hWnd, &ps);  break;  case WM\_COMMAND:  wmId = LOWORD(wParam);  wmEvent = HIWORD(wParam);  hdc = GetDC(hWnd);  switch (wmId) {  case BTN\_MYBUTTON\_ID\_1:  // Draw the inner rectangle using a hatch brush transparent background.  SetHatchBrushBackground(hdc, true);  MessageBox(hWnd, \_T("Hatch brush background is TRANSPARENT"), \_T("Information"), MB\_OK);  break;  case BTN\_MYBUTTON\_ID\_2:  // Draw the inner rectangle using a hatch brush opaque background.  SetHatchBrushBackground(hdc, false);  MessageBox(hWnd, \_T("Hatch brush background is OPAQUE"), \_T("Information"), MB\_OK);  break;  }  break;  case WM\_DESTROY:  PostQuitMessage(0);  break;  default:  return DefWindowProc(hWnd, message, wParam, lParam);  break;  }  return 0;  } 要求  |  |  | | --- | --- | | **标头** | wingdi.h (包括 Windows.h) | | **Library** | Gdi32.lib | | **DLL** | Gdi32.dll |  另请参阅 [画笔函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/brush-functions)  [画笔概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/brushes)  [COLORREF](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/colorref)  [CreateDIBPatternBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createdibpatternbrush)  [CreateDIBPatternBrushPt](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createdibpatternbrushpt)  [CreatePatternBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createpatternbrush)  [CreateSolidBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createsolidbrush)  [DeleteObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-deleteobject)  [GetBrushOrgEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getbrushorgex)  [RGB](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-rgb)  [SelectObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-selectobject)  [SetBkMode](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-setbkmode)  [SetBrushOrgEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-setbrushorgex) |

# 这一节主要学习window窗口的创建以及消息处理，主要又两个函数一个是WinMain，另外一个是WndProc我们用两种方式来学习

## 1.从零开始全部手写代码，其实可以用向导创建一个项目，然后我们移除它的代码，新建c文件，自己来写

|  |
| --- |
|  |

### MyWinApp的代码如下。

|  |
| --- |
| #include"stdafx.h"  #include "resource.h"  #define MAX\_LOADSTRING 100  // 全局变量:  HINSTANCE hInst; // 当前实例  TCHAR szTitle[MAX\_LOADSTRING]; // 标题栏文本  TCHAR szWindowClass[MAX\_LOADSTRING];  LRESULT CALLBACK WndProc(HWND hwnd,UINT uMsg,WPARAM wParam,LPARAM lParam);  int APIENTRY \_tWinMain(HINSTANCE hInstance,HINSTANCE hprevInstance,LPTSTR lpCmdLine,int nCmdShow)  {  MSG msg;    //从字符串表中加载字符串  LoadString(hInstance,IDS\_APP\_TITLE,szTitle,MAX\_LOADSTRING);  LoadString(hInstance, IDC\_LESSION6CREATEWIN,szWindowClass,MAX\_LOADSTRING);  //1.设计窗口类  WNDCLASS wcex;  wcex.style = CS\_HREDRAW | CS\_VREDRAW;  wcex.lpfnWndProc = WndProc;  wcex.cbClsExtra = 0;  wcex.cbWndExtra = 0;  wcex.hInstance = hInstance;  wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_LESSION6CREATEWIN));  wcex.hCursor = LoadCursor(NULL, IDC\_ARROW);  wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW+1);  wcex.lpszMenuName = MAKEINTRESOURCE(IDC\_LESSION6CREATEWIN);  wcex.lpszClassName = \_T("MyWin");  //2.注册窗口类  RegisterClass(&wcex);    //3.创建窗口  HWND hwnd;  hwnd = CreateWindow(  \_T("MyWin"),  \_T("Lession6"),  WS\_OVERLAPPEDWINDOW,  CW\_USEDEFAULT,  CW\_USEDEFAULT,  CW\_USEDEFAULT,  CW\_USEDEFAULT,  NULL,  NULL,  hInstance,  NULL);  if (!hwnd)  {  MessageBox(NULL,\_T("创建窗口失败"),\_T("错误"),MB\_OK);  }  //4.显示窗口  ShowWindow(hwnd, nCmdShow);  //5.更新窗口  UpdateWindow(hwnd);  //6.学习循环  while(GetMessage(&msg,NULL,0,0))  {  TranslateMessage(&msg);  DispatchMessage(&msg);  }  return (int) msg.wParam;    }  LRESULT CALLBACK WndProc(HWND hwnd,UINT uMsg,WPARAM wParam,LPARAM lParam)  {  int wmId, wmEvent;  PAINTSTRUCT ps;  HDC hdc;  switch(uMsg)  {  case WM\_COMMAND:  wmId = LOWORD(wParam);  switch(wmId)  {  case IDM\_EXIT:  PostQuitMessage(0);  break;  case IDM\_ABOUT:  MessageBox(NULL,\_T("Lession6-create-win\n version 1.0\n Date Created: 2024/08/29"),\_T("About"),0);  break;  default:  return DefWindowProc(hwnd, uMsg, wParam, lParam);  }  break;  case WM\_PAINT:  hdc = BeginPaint(hwnd,&ps);  TextOut(hdc,0,0,\_T("Hello, Windows"),wcslen(\_T("Hello, Windows")));  EndPaint(hwnd,&ps);  break;  case WM\_DESTROY:  PostQuitMessage(0);  default:  return DefWindowProc(hwnd, uMsg, wParam, lParam);  }  return 0;  } |

### 注意：WndProc非常重要，如果你在写switch语句的时候忘了写defaut里面的代码，会导致窗口创建失败

## 2.使用向导创建项目，有一个示例代码

## 我们把向导创建的代码恢复回来

|  |
| --- |
|  |

### Lession6-create-win.cpp的代码如下

|  |
| --- |
| // Lession6-create-win.cpp : 定义应用程序的入口点。  //  #include "stdafx.h"  #include "Lession6-create-win.h"  #define MAX\_LOADSTRING 100  // 全局变量:  HINSTANCE hInst; // 当前实例  TCHAR szTitle[MAX\_LOADSTRING]; // 标题栏文本  TCHAR szWindowClass[MAX\_LOADSTRING]; // 主窗口类名  // 此代码模块中包含的函数的前向声明:  ATOM MyRegisterClass(HINSTANCE hInstance);  BOOL InitInstance(HINSTANCE, int);  LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);  INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);  int APIENTRY \_tWinMain(HINSTANCE hInstance,  HINSTANCE hPrevInstance,  LPTSTR lpCmdLine,  int nCmdShow)  {  UNREFERENCED\_PARAMETER(hPrevInstance);  UNREFERENCED\_PARAMETER(lpCmdLine);  // TODO: 在此放置代码。  MSG msg;  HACCEL hAccelTable;  // 初始化全局字符串  LoadString(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);  LoadString(hInstance, IDC\_LESSION6CREATEWIN, szWindowClass, MAX\_LOADSTRING);  MyRegisterClass(hInstance);  // 执行应用程序初始化:  if (!InitInstance (hInstance, nCmdShow))  {  return FALSE;  }  hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_LESSION6CREATEWIN));  // 主消息循环:  while (GetMessage(&msg, NULL, 0, 0))  {  if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))  {  TranslateMessage(&msg);  DispatchMessage(&msg);  }  }  return (int) msg.wParam;  }  //  // 函数: MyRegisterClass()  //  // 目的: 注册窗口类。  //  // 注释:  //  // 仅当希望  // 此代码与添加到Windows 95 中的“RegisterClassEx”  // 函数之前的Win32 系统兼容时，才需要此函数及其用法。调用此函数十分重要，  // 这样应用程序就可以获得关联的  // “格式正确的”小图标。  //  ATOM MyRegisterClass(HINSTANCE hInstance)  {  WNDCLASSEX wcex;  wcex.cbSize = sizeof(WNDCLASSEX);  wcex.style = CS\_HREDRAW | CS\_VREDRAW;  wcex.lpfnWndProc = WndProc;  wcex.cbClsExtra = 0;  wcex.cbWndExtra = 0;  wcex.hInstance = hInstance;  wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_LESSION6CREATEWIN));  wcex.hCursor = LoadCursor(NULL, IDC\_ARROW);  wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW+1);  wcex.lpszMenuName = MAKEINTRESOURCE(IDC\_LESSION6CREATEWIN);  wcex.lpszClassName = szWindowClass;  wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));  return RegisterClassEx(&wcex);  }  //  // 函数: InitInstance(HINSTANCE, int)  //  // 目的: 保存实例句柄并创建主窗口  //  // 注释:  //  // 在此函数中，我们在全局变量中保存实例句柄并  // 创建和显示主程序窗口。  //  BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)  {  HWND hWnd;  hInst = hInstance; // 将实例句柄存储在全局变量中  hWnd = CreateWindow(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,  CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, NULL, NULL, hInstance, NULL);  if (!hWnd)  {  return FALSE;  }  ShowWindow(hWnd, nCmdShow);  UpdateWindow(hWnd);  return TRUE;  }  //  // 函数: WndProc(HWND, UINT, WPARAM, LPARAM)  //  // 目的: 处理主窗口的消息。  //  // WM\_COMMAND - 处理应用程序菜单  // WM\_PAINT - 绘制主窗口  // WM\_DESTROY - 发送退出消息并返回  //  //  LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)  {  int wmId, wmEvent;  PAINTSTRUCT ps;  HDC hdc;  switch (message)  {  case WM\_COMMAND:  wmId = LOWORD(wParam);  wmEvent = HIWORD(wParam);  // 分析菜单选择:  switch (wmId)  {  case IDM\_ABOUT:  DialogBox(hInst, MAKEINTRESOURCE(IDD\_ABOUTBOX), hWnd, About);  break;  case IDM\_EXIT:  DestroyWindow(hWnd);  break;  default:  return DefWindowProc(hWnd, message, wParam, lParam);  }  break;  case WM\_PAINT:  hdc = BeginPaint(hWnd, &ps);  // TODO: 在此添加任意绘图代码...  EndPaint(hWnd, &ps);  break;  case WM\_DESTROY:  PostQuitMessage(0);  break;  default:  return DefWindowProc(hWnd, message, wParam, lParam);  }  return 0;  }  // “关于”框的消息处理程序。  INT\_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)  {  UNREFERENCED\_PARAMETER(lParam);  switch (message)  {  case WM\_INITDIALOG:  return (INT\_PTR)TRUE;  case WM\_COMMAND:  if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)  {  EndDialog(hDlg, LOWORD(wParam));  return (INT\_PTR)TRUE;  }  break;  }  return (INT\_PTR)FALSE;  } |

## 老师的例子有一个新API加载LoadImage，用法如下：

|  |
| --- |
|  |

### 参数说明

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Parameters  hinst  Handle to an instance of the module that contains the image to be loaded. To load an OEM image, set this parameter to zero.  lpszName  Handle to the image to load.  If the hinst parameter is non-NULL and the fuLoad parameter does not include LR\_LOADFROMFILE, lpszName is a pointer to a null-terminated string that contains the name of the image resource in the hinst module.  If hinst is NULL and LR\_LOADFROMFILE is not specified, the low-order word of this parameter must be the identifier of the OEM image to load. The OEM image identifiers are defined in WINUSER.H and have the following prefixes:   |  |  | | --- | --- | | Prefix | Meaning | | OBM\_ | OEM bitmaps | | OIC\_ | OEM icons | | OCR\_ | OEM cursors |   If the fuLoad parameter includes the LR\_LOADFROMFILE value, lpszName is the name of the file that contains the image.  uType  Specifies the type of image to be loaded. This parameter can be one of the following values:   |  |  | | --- | --- | | Value | Meaning | | IMAGE\_BITMAP | Loads a bitmap. | | IMAGE\_CURSOR | Loads a cursor. | | IMAGE\_ICON | Loads an icon. |   cxDesired  Specifies the width, in pixels, of the icon or cursor. If this parameter is zero and the fuLoad parameter is LR\_DEFAULTSIZE, the function uses the SM\_CXICON or SM\_CXCURSOR system metric value to set the width. If this parameter is zero and LR\_DEFAULTSIZE is not used, the function uses the actual resource width.  cyDesired  Specifies the height, in pixels, of the icon or cursor. If this parameter is zero and the fuLoad parameter is LR\_DEFAULTSIZE, the function uses the SM\_CYICON or SM\_CYCURSOR system metric value to set the height. If this parameter is zero and LR\_DEFAULTSIZE is not used, the function uses the actual resource height.  fuLoad  Specifies a combination of the following values:   |  |  | | --- | --- | | Value | Meaning | | LR\_DEFAULTCOLOR | The default flag; it does nothing. All it means is "not LR\_MONOCHROME". | | LR\_CREATEDIBSECTION | When the uType parameter specifies IMAGE\_BITMAP, causes the function to return a DIB section bitmap rather than a compatible bitmap. This flag is useful for loading a bitmap without mapping it to the colors of the display device. | | LR\_DEFAULTSIZE | Uses the width or height specified by the system metric values for cursors or icons, if the cxDesired or cyDesired values are set to zero. If this flag is not specified and cxDesired and cyDesired are set to zero, the function uses the actual resource size. If the resource contains multiple images, the function uses the size of the first image. | | LR\_LOADFROMFILE | Loads the image from the file specified by the lpszName parameter. If this flag is not specified, lpszName is the name of the resource. | | LR\_LOADMAP3DCOLORS | Searches the color table for the image and replaces the following shades of gray with the corresponding 3D color: |  |  |  |  | | --- | --- | --- | |  | Color | Replaced with | |  | Dk Gray,  RGB(128,128,128) | COLOR\_3DSHADOW | |  | Gray,  RGB(192,192,192) | COLOR\_3DFACE | |  | Lt Gray,  RGB(223,223,223) | COLOR\_3DLIGHT |  |  |  | | --- | --- | | LR\_LOADTRANSPARENT | Retrieves the color value of the first pixel in the image and replaces the corresponding entry in the color table with the default window color (COLOR\_WINDOW). All pixels in the image that use that entry become the default window color. This value applies only to images that have corresponding color tables.  If fuLoad includes both the LR\_LOADTRANSPARENT and LR\_LOADMAP3DCOLORS values, LRLOADTRANSPARENT takes precedence. However, the color table entry is replaced with COLOR\_3DFACE rather than COLOR\_WINDOW. | | LR\_MONOCHROME | Loads the image in black and white. | | LR\_SHARED | Shares the image handle if the image is loaded multiple times. If LR\_SHARED is not set, a second call to LoadImage for the same resource will load the image again and return a different handle.  Do not use LR\_SHARED for images that have non-standard sizes, that may change after loading, or that are loaded from a file.  Windows 95 and Windows 98: The function finds the first image with the requested resource name in the cache, regardless of the size requested. | | LR\_VGACOLOR | Uses true VGA colors. |   Return Values  If the function succeeds, the return value is the handle of the newly loaded image.  If the function fails, the return value is NULL. To get extended error information, call [GetLastError](JavaScript:alink_1.Click()).  Remarks  When you are finished using the bitmap, cursor, or icon, you can release its associated memory by calling one of the functions in the following table.   |  |  | | --- | --- | | Resource | Release function | | Bitmap | [DeleteObject](JavaScript:alink_2.Click()) | | Cursor | [DestroyCursor](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/cursors_6sdu.htm) | | Icon | [DestroyIcon](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/icons_5kxa.htm) |   The system automatically deletes these resources when the process that created them terminates, however, calling the appropriate function saves memory and decreases the size of the process's working set. |

### 使用如下：

|  |
| --- |
|  |

## 老师也用到了一个叫做ExitThread的函数，用法如下

|  |
| --- |
|  |

### 参数说明

|  |
| --- |
|  |

### 不过：退出window应用程序，Microsoft建议我们使用PostQuitMessage函数

# 小结

## 1.一个windows应用程序的编写过程：先编写\_tWinMain,再编写WndProc，在主函数中，定义窗口类->注册窗口类->创建窗口->显示窗口->更新窗口->消息循环，窗口过程函数主要是处理消息，需要写一堆switch…case，最后不要忘了在default语句里面返回默认窗口过程的处理结果

## 2.有两个注册窗口类的函数，分别对应一个窗口类结构体

### 1》RegisterClass，它对应的窗口类结构体是WNDCLASS，数据成员如下

|  |
| --- |
|  |

### 2》. RegisterClassEx，它对应的窗口类结构体是WNDCLASSEX，数据成员如下

|  |
| --- |
|  |

### 下面是一些可以取的值

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **cbSize**  Specifies the size, in bytes, of this structure. Set this member to sizeof(WNDCLASSEX). Be sure to set this member before calling the [**GetClassInfoEx**](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/winclass_2jp4.htm) function.  **style**  Specifies the class style(s). Styles can be combined by using the bitwise OR (|) operator. This member can be any combination of the following values:   |  |  | | --- | --- | | **Value** | **Action** | | CS\_BYTEALIGNCLIENT | Aligns the window's client area on the byte boundary (in the x direction). This style affects the width of the window and its horizontal position on the display. | | CS\_BYTEALIGNWINDOW | Aligns a window on a byte boundary (in the x direction). This style affects the width of the window and its horizontal position on the display. | | CS\_CLASSDC | Allocates one device context to be shared by all windows in the class. Because window classes are process specific, it is possible for multiple threads of an application to create a window of the same class. It is also possible for the threads to attempt to use the device context simultaneously. When this happens, the system allows only one thread to successfully finish its drawing operation. For more information, see [Device Contexts](JavaScript:alink_1.Click()). | | CS\_DBLCLKS | Sends double-click messages to the window procedure when the user double-clicks the mouse while the cursor is within a window belonging to the class. | | CS\_GLOBALCLASS | Allows an application to create a window of the class regardless of the value of the *hInstance* parameter passed to the [**CreateWindowEx**](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/windows_1w6w.htm) function. If you do not specify this style, the *hInstance* parameter passed to the **CreateWindow** (or **CreateWindowEx**) function must be the same as the *hInstance* parameter passed to the [**RegisterClassEx**](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/winclass_0wc8.htm) function. | |  | You can create a global class by creating the window class in a dynamic-link library (DLL) and listing the name of the DLL in the registry under the following keys: | |  | **HKEY\_LOCAL\_MACHINE**\**Software** \**Microsoft**\**Windows NT**\ **CurrentVersion**\**Windows**\**AppInit\_DLLs** | |  | Whenever a process starts, the system loads the specified DLLs in the context of the newly started process before calling the entry-point function in that process. The DLL must register the class during its initialization procedure and must specify the CS\_GLOBALCLASS style. | | CS\_HREDRAW | Redraws the entire window if a movement or size adjustment changes the width of the client area. | | CS\_NOCLOSE | Disables **Close** on the **window** menu. | | CS\_OWNDC | Allocates a unique device context for each window in the class. For more information, see [Device Contexts](JavaScript:alink_2.Click()). | | CS\_PARENTDC | Sets the clipping region of the child window to that of the parent window so that the child can draw on the parent. A window with the CS\_PARENTDC style bit receives a regular device context from the system's cache of device contexts. It does not give the child the parent's device context or device context settings. Specifying CS\_PARENTDC enhances an application's performance. For more information, see [Device Contexts](JavaScript:alink_3.Click()). | | CS\_SAVEBITS | Saves, as a bitmap, the portion of the screen image obscured by a window. The system uses the saved bitmap to re-create the screen image when the window is removed. The system displays the bitmap at its original location and does not send [WM\_PAINT](JavaScript:alink_4.Click()) messages to windows obscured by the window if the memory used by the bitmap has not been discarded and if other screen actions have not invalidated the stored image. This style is useful for small windows (for example, menus or dialog boxes) that are displayed briefly and then removed before other screen activity takes place. This style increases the time required to display the window, because the system must first allocate memory to store the bitmap. | | CS\_VREDRAW | Redraws the entire window if a movement or size adjustment changes the height of the client area. |   **lpfnWndProc**  Pointer to the window procedure. You must use the [**CallWindowProc**](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/winprocs_9dlv.htm) function to call the window procedure. For more information, see [**WindowProc**](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/winprocs_53xf.htm).  **cbClsExtra**  Specifies the number of extra bytes to allocate following the window-class structure. The system initializes the bytes to zero.  **cbWndExtra**  Specifies the number of extra bytes to allocate following the window instance. The system initializes the bytes to zero. If an application uses **WNDCLASSEX** to register a dialog box created by using the **CLASS** directive in the resource file, it must set this member to DLGWINDOWEXTRA.  **hInstance**  Handle to the instance that the window procedure of this class is within.  **hIcon**  Handle to the class icon. This member must be a handle of an icon resource. If this member is NULL, an application must draw an icon whenever the user minimizes the application's window.  **hCursor**  Handle to the class cursor. This member must be a handle of a cursor resource. If this member is NULL, an application must explicitly set the cursor shape whenever the mouse moves into the application's window.  **hbrBackground**  Handle to the class background brush. This member can be a handle to the physical brush to be used for painting the background, or it can be a color value. A color value must be one of the following standard system colors (the value 1 must be added to the chosen color). If a color value is given, you must convert it to one of the following **HBRUSH** types:  COLOR\_ACTIVEBORDER COLOR\_ACTIVECAPTION COLOR\_APPWORKSPACE COLOR\_BACKGROUND COLOR\_BTNFACE COLOR\_BTNSHADOW COLOR\_BTNTEXT COLOR\_CAPTIONTEXT COLOR\_GRAYTEXT COLOR\_HIGHLIGHT COLOR\_HIGHLIGHTTEXT COLOR\_INACTIVEBORDER COLOR\_INACTIVECAPTION COLOR\_MENU COLOR\_MENUTEXT COLOR\_SCROLLBAR COLOR\_WINDOW COLOR\_WINDOWFRAME COLOR\_WINDOWTEXT  The system automatically deletes class background brushes when the class is freed. An application should not delete these brushes, because a class may be used by multiple instances of an application.  When this member is NULL, an application must paint its own background whenever it is requested to paint in its client area. To determine whether the background must be painted, an application can either process the WM\_ERASEBKGND message or test the **fErase** member of the [**PAINTSTRUCT**](JavaScript:alink_5.Click()) structure filled by the [**BeginPaint**](JavaScript:alink_6.Click()) function.  **lpszMenuName**  Pointer to a null-terminated character string that specifies the resource name of the class menu, as the name appears in the resource file. If you use an integer to identify the menu, use the [**MAKEINTRESOURCE**](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/resource_2dd1.htm) macro. If this member is NULL, windows belonging to this class have no default menu.  **lpszClassName**  Pointer to a null-terminated string or is an atom. If this parameter is an atom, it must be a global atom created by a previous call to the **GlobalAddAtom** function. The atom, a 16-bit value, must be in the low-order word of **lpszClassName**; the high-order word must be zero.  If **lpszClassName** is a string, it specifies the window class name.  **hIconSm**  Handle to a small icon that is associated with the window class. If this member is NULL, the system searches the icon resource specified by the **hIcon** member for an icon of the appropriate size to use as the small icon. |

## 3.这个应用程序最复杂的API就是创建窗口函数

|  |
| --- |
|  |

### 参数说明

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Parameters *lpClassName*  Pointer to a null-terminated string or is an integer atom. If this parameter is an atom, it must be a global atom created by a previous call to the [**GlobalAddAtom**](JavaScript:alink_1.Click()) function. The atom, a 16-bit value less than 0xC000, must be in the low-order word of *lpClassName*; the high-order word must be zero.  If *lpClassName* is a string, it specifies the window class name. The class name can be any name registered with the [**RegisterClassEx**](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/winclass_0wc8.htm) function or any of the predefined control-class names. For a complete list, see the Remarks section.  *lpWindowName*  Pointer to a null-terminated string that specifies the window name.  If the window style specifies a title bar, the window title pointed to by *lpWindowName* is displayed in the title bar. When using **CreateWindow** to create controls, such as buttons, check boxes, and static controls, use *lpWindowName* to specify the text of the control.  *dwStyle*  Specifies the style of the window being created. This parameter can be a combination of the following window styles, plus the control styles indicated in the Remarks section.   |  |  | | --- | --- | | **Style** | **Meaning** | | WS\_BORDER | Creates a window that has a thin-line border. | | WS\_CAPTION | Creates a window that has a title bar (includes the WS\_BORDER style). | | WS\_CHILD | Creates a child window. This style cannot be used with the WS\_POPUP style. | | WS\_CHILDWINDOW | Same as the WS\_CHILD style. | | WS\_CLIPCHILDREN | Excludes the area occupied by child windows when drawing occurs within the parent window. This style is used when creating the parent window. | | WS\_CLIPSIBLINGS | Clips child windows relative to each other; that is, when a particular child window receives a [WM\_PAINT](JavaScript:alink_2.Click()) message, the WS\_CLIPSIBLINGS style clips all other overlapping child windows out of the region of the child window to be updated. If WS\_CLIPSIBLINGS is not specified and child windows overlap, it is possible, when drawing within the client area of a child window, to draw within the client area of a neighboring child window. | | WS\_DISABLED | Creates a window that is initially disabled. A disabled window cannot receive input from the user. | | WS\_DLGFRAME | Creates a window that has a border of a style typically used with dialog boxes. A window with this style cannot have a title bar. | | WS\_GROUP | Specifies the first control of a group of controls. The group consists of this first control and all controls defined after it, up to the next control with the WS\_GROUP style. The first control in each group usually has the WS\_TABSTOP style so that the user can move from group to group. The user can subsequently change the keyboard focus from one control in the group to the next control in the group by using the direction keys. | | WS\_HSCROLL | Creates a window that has a horizontal scroll bar. | | WS\_ICONIC | Creates a window that is initially minimized. Same as the WS\_MINIMIZE style. | | WS\_MAXIMIZE | Creates a window that is initially maximized. | | WS\_MAXIMIZEBOX | Creates a window that has a Maximize button. Cannot be combined with the WS\_EX\_CONTEXTHELP style. The WS\_SYSMENU style must also be specified. | | WS\_MINIMIZE | Creates a window that is initially minimized. Same as the WS\_ICONIC style. | | WS\_MINIMIZEBOX | Creates a window that has a Minimize button. Cannot be combined with the WS\_EX\_CONTEXTHELP style. The WS\_SYSMENU style must also be specified. | | WS\_OVERLAPPED | Creates an overlapped window. An overlapped window has a title bar and a border. Same as the WS\_TILED style. | | WS\_OVERLAPPEDWINDOW | Creates an overlapped window with the WS\_OVERLAPPED, WS\_CAPTION, WS\_SYSMENU, WS\_THICKFRAME, WS\_MINIMIZEBOX, and WS\_MAXIMIZEBOX styles. Same as the WS\_TILEDWINDOW style. | | WS\_POPUP | Creates a pop-up window. This style cannot be used with the WS\_CHILD style. | | WS\_POPUPWINDOW | Creates a pop-up window with WS\_BORDER, WS\_POPUP, and WS\_SYSMENU styles. The WS\_CAPTION and WS\_POPUPWINDOW styles must be combined to make the window menu visible. | | WS\_SIZEBOX | Creates a window that has a sizing border. Same as the WS\_THICKFRAME style. | | WS\_SYSMENU | Creates a window that has a window-menu on its title bar. The WS\_CAPTION style must also be specified. | | WS\_TABSTOP | Specifies a control that can receive the keyboard focus when the user presses the tab key. Pressing the tab key changes the keyboard focus to the next control with the WS\_TABSTOP style. | | WS\_THICKFRAME | Creates a window that has a sizing border. Same as the WS\_SIZEBOX style. | | WS\_TILED | Creates an overlapped window. An overlapped window has a title bar and a border. Same as the WS\_OVERLAPPED style. | | WS\_TILEDWINDOW | Creates an overlapped window with the WS\_OVERLAPPED, WS\_CAPTION, WS\_SYSMENU, WS\_THICKFRAME, WS\_MINIMIZEBOX, and WS\_MAXIMIZEBOX styles. Same as the WS\_OVERLAPPEDWINDOW style. | | WS\_VISIBLE | Creates a window that is initially visible. | | WS\_VSCROLL | Creates a window that has a vertical scroll bar. |   *x*  Specifies the initial horizontal position of the window. For an overlapped or pop-up window, the *x* parameter is the initial x-coordinate of the window's upper-left corner, in screen coordinates. For a child window, *x* is the x-coordinate of the upper-left corner of the window relative to the upper-left corner of the parent window's client area.  If this parameter is set to CW\_USEDEFAULT, the system selects the default position for the window's upper-left corner and ignores the *y* parameter. CW\_USEDEFAULT is valid only for overlapped windows; if it is specified for a pop-up or child window, the *x* and *y* parameters are set to zero.  *y*  Specifies the initial vertical position of the window. For an overlapped or pop-up window, the *y* parameter is the initial y-coordinate of the window's upper-left corner, in screen coordinates. For a child window, *y* is the initial y-coordinate of the upper-left corner of the child window relative to the upper-left corner of the parent window's client area. For a list box, *y* is the initial y-coordinate of the upper-left corner of the list box's client area relative to the upper-left corner of the parent window's client area.  If an overlapped window is created with the WS\_VISIBLE style bit set and the *x* parameter is set to CW\_USEDEFAULT, the system ignores the *y* parameter.  *nWidth*  Specifies the width, in device units, of the window. For overlapped windows, *nWidth* is either the window's width, in screen coordinates, or CW\_USEDEFAULT. If *nWidth* is CW\_USEDEFAULT, the system selects a default width and height for the window; the default width extends from the initial x-coordinate to the right edge of the screen, and the default height extends from the initial y-coordinate to the top of the icon area. CW\_USEDEFAULT is valid only for overlapped windows; if CW\_USEDEFAULT is specified for a pop-up or child window, *nWidth* and *nHeight* are set to zero.  *nHeight*  Specifies the height, in device units, of the window. For overlapped windows, *nHeight* is the window's height, in screen coordinates. If *nWidth* is set to CW\_USEDEFAULT, the system ignores *nHeight*.  *hWndParent*  Handle to the parent or owner window of the window being created. To create a child window or an owned window, supply a valid window handle. This parameter is optional for pop-up windows.  **Windows NT 5.0 and later:** To create a message-only window, supply HWND\_MESSAGE or a handle to an existing message-only window.  *hMenu*  Handle to a menu, or specifies a child-window identifier depending on the window style. For an overlapped or pop-up window, *hMenu* identifies the menu to be used with the window; it can be NULL if the class menu is to be used. For a child window, *hMenu* specifies the child-window identifier, an integer value used by a dialog box control to notify its parent about events. The application determines the child-window identifier; it must be unique for all child windows with the same parent window.  *hInstance*  Handle to the instance of the module to be associated with the window.  *lpParam*  A pointer to a value to be passed to the window through the [**CREATESTRUCT**](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/windows_06lu.htm) structure passed in the *lParam* parameter the [WM\_CREATE](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/windows_41d1.htm) message. If an application calls **CreateWindow** to create a multiple document interface (MDI) client window, *lpParam* must point to a [**CLIENTCREATESTRUCT**](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/windows_5bhu.htm) structure. Return Values If the function succeeds, the return value is a handle to the new window.  If the function fails, the return value is NULL. To get extended error information, call [**GetLastError**](JavaScript:alink_3.Click()). Remarks Before returning, **CreateWindow** sends a [WM\_CREATE](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/windows_41d1.htm) message to the window procedure. For overlapped, pop-up, and child windows, **CreateWindow** sends [WM\_CREATE](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/windows_41d1.htm), [WM\_GETMINMAXINFO](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/windows_2c8f.htm), and [WM\_NCCREATE](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/windows_8fol.htm) messages to the window. The *lParam* parameter of the [WM\_CREATE](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/windows_41d1.htm) message contains a pointer to a [**CREATESTRUCT**](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/windows_06lu.htm) structure. If the WS\_VISIBLE style is specified, **CreateWindow** sends the window all the messages required to activate and show the window.  For information on controlling whether the Taskbar displays a button for the created window, see [Visibility of Taskbar Buttons](JavaScript:alink_4.Click()).  The following predefined control classes can be specified in the *lpClassName* parameter. Note the corresponding control styles you can use in the *dwStyle* parameter.   |  |  | | --- | --- | | **Class** | **Meaning** | | BUTTON | Designates a small rectangular child window that represents a button the user can click to turn it on or off. Button controls can be used alone or in groups, and they can either be labeled or appear without text. Button controls typically change appearance when the user clicks them. For more information, see [Buttons](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/buttons_7zhv.htm). | |  | For a table of the button styles you can specify in the *dwStyle* parameter, see [Button Styles](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/buttons_34c3.htm). | | COMBOBOX | Designates a control consisting of a list box and a selection field similar to an edit control. When using this style, an application should either display the list box at all times or enable a drop-down list box. If the list box is visible, typing characters into the selection field highlights the first list box entry that matches the characters typed. Conversely, selecting an item in the list box displays the selected text in the selection field. For more information, see [Combo Boxes](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/combobox_77eb.htm). | |  | For a table of the combo box styles you can specify in the *dwStyle* parameter, see [Combo Box Styles](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/combobox_8mcz.htm). | | EDIT | Designates a rectangular child window into which the user can type text from the keyboard. The user selects the control and gives it the keyboard focus by clicking it or moving to it by pressing the tab key. The user can type text when the edit control displays a flashing caret; use the mouse to move the cursor, select characters to be replaced, or position the cursor for inserting characters; or use the backspace key to delete characters. For more information, see [Edit Controls](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/editcon_4uwj.htm). | |  | For a table of the edit control styles you can specify in the *dwStyle* parameter, see [Edit Control Styles](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/editcon_7rqr.htm). | | LISTBOX | Designates a list of character strings. Specify this control whenever an application must present a list of names, such as filenames, from which the user can choose. The user can select a string by clicking it. A selected string is highlighted, and a notification message is passed to the parent window. For more information, see [List Boxes](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/listbox_9s4z.htm). | |  | For a table of the list box styles you can specify in the *dwStyle* parameter, see [List Box Styles](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/listbox_5cfn.htm). | | MDICLIENT | Designates an MDI client window. This window receives messages that control the MDI application's child windows. The recommended style bits are WS\_CLIPCHILDREN and WS\_CHILD. Specify the WS\_HSCROLL and WS\_VSCROLL styles to create an MDI client window that allows the user to scroll MDI child windows into view. For more information, see [Multiple Document Interface](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/mdocint_6crp.htm). | | RichEdit | Designates a Rich Edit version 1.0 control. This window lets the user view and edit text with character and paragraph formatting, and can include embedded COM objects. For more information, see [Rich Edit Controls](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/richedit_5a7n.htm). | |  | For a table of the rich edit control styles you can specify in the *dwStyle* parameter, see [Rich Edit Control Styles](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/richedit_5lpv.htm). | | RICHEDIT\_CLASS | Designates a Rich Edit version 2.0 control. This controls let the user view and edit text with character and paragraph formatting, and can include embedded COM objects. For more information, see [Rich Edit Controls](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/richedit_5a7n.htm). | |  | For a table of the rich edit control styles you can specify in the *dwStyle* parameter, see [Rich Edit Control Styles](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/richedit_5lpv.htm). | | SCROLLBAR | Designates a rectangle that contains a scroll box and has direction arrows at both ends. The scroll bar sends a notification message to its parent window whenever the user clicks the control. The parent window is responsible for updating the position of the scroll box, if necessary. For more information, see [Scroll Bars](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/scrolbar_3v8z.htm). | |  | For a table of the scroll bar control styles you can specify in the *dwStyle* parameter, see [Scroll Bar Control Styles](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/scrolbar_8s37.htm). | | STATIC | Designates a simple text field, box, or rectangle used to label, box, or separate other controls. Static controls take no input and provide no output. For more information, see [Static Controls](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/statcon_1qib.htm). | |  | For a table of the static control styles you can specify in the *dwStyle* parameter, see [Static Control Styles](mk:@MSITStore:C:\Program%20Files%20(x86)\Microsoft%20Visual%20Studio\MSDN98\98VS\2052\winui.chm::/devdoc/live/pdui/statcon_360j.htm). |   **Windows 95:** The system can support a maximum of 16,364 window handles.  **Note**  If you specify Windows version 4.x when linking your application, its windows cannot have caption buttons unless they also have window menus. This is not a requirement if you specify Windows version 3.x when linking your application.  **Windows CE:** **CreateWindow** is implemented as a macro. It is defined as **CreateWindowEx**, but with the *dwExStyle* parameter set to 0L.  Menu bars are not supported. The *hMenu* parameter must be NULL, unless it is used as a child-window identifier.  The MDICLIENT window class is not supported.  The *dwStyle* parameter can be a combination of the window styles and control styles documented in:  Dialog Boxes  Windows  Controls  The following *dwStyle* flags are not supported for windows:   |  |  | | --- | --- | | WS\_CHILDWINDOW | WS\_ICONIC | | WS\_MAXIMIZE | WS\_MAXIMIZEBOX | | WS\_MINIMIZE | WS\_MINIMIZEBOX | | WS\_OVERLAPPEDWINDOW | WS\_POPUPWINDOW | | WS\_SIZEBOX | WS\_THICKFRAME | | WS\_TILED | WS\_TILEDWINDOW |   The following *dwStyle* flags are not supported for controls and dialog boxes:   |  |  | | --- | --- | | **Unsupported button styles** | **Unsupported static control styles** | | BS\_LEFTTEXT | SS\_BLACKFRAME | | BS\_MULTILINE | SS\_GRAYFRAME | | BS\_TEXT | SS\_METAPICT | | BS\_USERBUTTON | SS\_SIMPLE | | **Unsupported combo box styles** | SS\_WHITERECT | | CBS\_OWNERDRAWFIXED | SS\_BLACKRECT | | CBS\_OWNERDRAWVARIABLE | SS\_GRAYRECT | | CBS\_SIMPLE | SS\_RIGHTIMAGE | | **Unsupported list box control styles** | SS\_WHITEFRAME | | LBS\_NODATA | **Unsupported dialog box styles** | | LBS\_OWNERDRAWFIXED | DS\_ABSALIGN | | LBS\_OWNERDRAWVARIABLE | DS\_CENTERMOUSE | | LBS\_STANDARD | DS\_CONTEXTHELP | | **Unsupported scroll bar styles** | DS\_FIXEDSYS | | SBS\_BOTTOMALIGN | DS\_NOFAILCREATE | | SBS\_RIGHTALIGN | DS\_NOIDLEMSG | | SBS\_SIZEBOXBOTTOMRIGHTALIGN | DS\_SYSMODAL | | SBS\_SIZEGRIP |  |   You can use the BS\_OWNERDRAW style as a substitute for the BS\_USERBUTTON style.  You can use the SS\_LEFT or SS\_LEFTNOWORDWRAP style instead of the SS\_SIMPLE style for static controls..  The MDICLIENT window class is not supported.  All windows implicitly have the WS\_CLIPSIBLINGS and WS\_CLIPCHILDREN styles.  Windows CE 1.0 does not support owned windows, except for dialog boxes. If the *hwndParent* parameter is not NULL, the window is implicitly given the WS\_CHILD style.  Windows CE version 1.0 does not support menu bars. |

# 扩展，我们可以学习一些在窗口显示位图的函数BitBlt，用法如下

|  |
| --- |
|  |

## 参数说明

**HDC** *hdcDest，*//这个用是需要显示位图的窗口句柄获取的设备上下文句柄

**int** *nXDest***,** // 目标位置的起始x坐标

**int** *nYDest***,** // 目标位置的起始x坐标

**int** *nWidth***,** // 需要显示的宽度

**int** *nHeight***,** // 需要显示的高度

**HDC** *hdcSrc***,** // 源设备上下文句柄，也就是有图片的设备上下文句柄

**int** *nXSrc***,** // 源图片的起始位置x坐标

**int** *nYSrc***,** // 源图片的起始位置y坐标

**DWORD** *dwRop* //操作方式，有下面的选项

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *dwRop*  Specifies a raster-operation code. These codes define how the color data for the source rectangle is to be combined with the color data for the destination rectangle to achieve the final color.  The following list shows some common raster operation codes:   |  |  | | --- | --- | | **Value** | **Description** | | BLACKNESS | Fills the destination rectangle using the color associated with index 0 in the physical palette. (This color is black for the default physical palette.) | | DSTINVERT | Inverts the destination rectangle. | | MERGECOPY | Merges the colors of the source rectangle with the specified pattern by using the Boolean AND operator. | | MERGEPAINT | Merges the colors of the inverted source rectangle with the colors of the destination rectangle by using the Boolean OR operator. | | NOTSRCCOPY | Copies the inverted source rectangle to the destination. | | NOTSRCERASE | Combines the colors of the source and destination rectangles by using the Boolean OR operator and then inverts the resultant color. | | PATCOPY | Copies the specified pattern into the destination bitmap. | | PATINVERT | Combines the colors of the specified pattern with the colors of the destination rectangle by using the Boolean XOR operator. | | PATPAINT | Combines the colors of the pattern with the colors of the inverted source rectangle by using the Boolean OR operator. The result of this operation is combined with the colors of the destination rectangle by using the Boolean OR operator. | | SRCAND | Combines the colors of the source and destination rectangles by using the Boolean AND operator. | | SRCCOPY | Copies the source rectangle directly to the destination rectangle. | | SRCERASE | Combines the inverted colors of the destination rectangle with the colors of the source rectangle by using the Boolean AND operator. | | SRCINVERT | Combines the colors of the source and destination rectangles by using the Boolean XOR operator. | | SRCPAINT | Combines the colors of the source and destination rectangles by using the Boolean OR operator. | | WHITENESS | Fills the destination rectangle using the color associated with index 1 in the physical palette. (This color is white for the default physical palette.) |  Return Values If the function succeeds, the return value is nonzero.  If the function fails, the return value is zero. |

### 常用的值是SRCCOPY

## Lession6-create-win.cpp完整代码如下、

|  |
| --- |
| // Lession6-create-win.cpp : 定义应用程序的入口点。  #include "stdafx.h"  #include "Lession6-create-win.h"  #define MAX\_LOADSTRING 100  // 自定义全局变量:  HDC hdc;  HDC memDC = NULL;  HBITMAP bitmap;  BITMAP bm;  // 全局变量:  HINSTANCE hInst; // 当前实例  TCHAR szTitle[MAX\_LOADSTRING]; // 标题栏文本  TCHAR szWindowClass[MAX\_LOADSTRING]; // 主窗口类名  // 此代码模块中包含的函数的前向声明:  ATOM MyRegisterClass(HINSTANCE hInstance);  BOOL InitInstance(HINSTANCE, int);  LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);  INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);  int APIENTRY \_tWinMain(HINSTANCE hInstance,  HINSTANCE hPrevInstance,  LPTSTR lpCmdLine,  int nCmdShow)  {  UNREFERENCED\_PARAMETER(hPrevInstance);  UNREFERENCED\_PARAMETER(lpCmdLine);  // TODO: 在此放置代码。  MSG msg;  HACCEL hAccelTable;  // 初始化全局字符串  LoadString(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);  LoadString(hInstance, IDC\_LESSION6CREATEWIN, szWindowClass, MAX\_LOADSTRING);  MyRegisterClass(hInstance);  // 执行应用程序初始化:  if (!InitInstance (hInstance, nCmdShow))  {  return FALSE;  }  hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_LESSION6CREATEWIN));  // 主消息循环:  while (GetMessage(&msg, NULL, 0, 0))  {  if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))  {  TranslateMessage(&msg);  DispatchMessage(&msg);  }  }  return (int) msg.wParam;  }  // 函数: MyRegisterClass()  // 目的: 注册窗口类。  // 注释:  // 仅当希望  // 此代码与添加到Windows 95 中的“RegisterClassEx”  // 函数之前的Win32 系统兼容时，才需要此函数及其用法。调用此函数十分重要，  // 这样应用程序就可以获得关联的  // “格式正确的”小图标。  ATOM MyRegisterClass(HINSTANCE hInstance)  {  WNDCLASSEX wcex;  wcex.cbSize = sizeof(WNDCLASSEX);  wcex.style = CS\_HREDRAW | CS\_VREDRAW;  wcex.lpfnWndProc = WndProc;  wcex.cbClsExtra = 0;  wcex.cbWndExtra = 0;  wcex.hInstance = hInstance;  wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_LESSION6CREATEWIN));  wcex.hCursor = LoadCursor(NULL, IDC\_ARROW);  wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW+1);  wcex.lpszMenuName = MAKEINTRESOURCE(IDC\_LESSION6CREATEWIN);  wcex.lpszClassName = szWindowClass;  wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));  return RegisterClassEx(&wcex);  }  // 函数: InitInstance(HINSTANCE, int)  // 目的: 保存实例句柄并创建主窗口  // 注释:  // 在此函数中，我们在全局变量中保存实例句柄并  // 创建和显示主程序窗口。  BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)  {  HWND hWnd;  hInst = hInstance; // 将实例句柄存储在全局变量中  hWnd = CreateWindow(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,  CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, NULL, NULL, hInstance, NULL);  if (!hWnd)  {  return FALSE;  }  ShowWindow(hWnd, nCmdShow);  UpdateWindow(hWnd);  return TRUE;  }  // 函数: WndProc(HWND, UINT, WPARAM, LPARAM)  // 目的: 处理主窗口的消息。  // WM\_COMMAND - 处理应用程序菜单  // WM\_PAINT - 绘制主窗口  // WM\_DESTROY - 发送退出消息并返回  LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)  {  int wmId, wmEvent;  PAINTSTRUCT ps;    switch (message)  {  case WM\_COMMAND:  wmId = LOWORD(wParam);  wmEvent = HIWORD(wParam);  // 分析菜单选择:  switch (wmId)  {  case IDM\_ABOUT:  DialogBox(hInst, MAKEINTRESOURCE(IDD\_ABOUTBOX), hWnd, About);  break;  case IDM\_EXIT:  DestroyWindow(hWnd);  break;  default:  return DefWindowProc(hWnd, message, wParam, lParam);  }  break;  case WM\_PAINT:  hdc = BeginPaint(hWnd, &ps);  // TODO: 在此添加任意绘图代码...  bitmap = LoadBitmap(hInst,MAKEINTRESOURCE(IDB\_BITMAP1));  memDC = CreateCompatibleDC(hdc);  SelectObject(memDC,bitmap);  GetObject(bitmap,sizeof(bm),&bm);  BitBlt(hdc,0,0,bm.bmWidth,bm.bmHeight,memDC,0,0,SRCCOPY);  DeleteDC(memDC);  EndPaint(hWnd, &ps);  break;  case WM\_DESTROY:  PostQuitMessage(0);  break;  default:  return DefWindowProc(hWnd, message, wParam, lParam);  }  return 0;  }  // “关于”框的消息处理程序。  INT\_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)  {  UNREFERENCED\_PARAMETER(lParam);  switch (message)  {  case WM\_INITDIALOG:  return (INT\_PTR)TRUE;  case WM\_COMMAND:  if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)  {  EndDialog(hDlg, LOWORD(wParam));  return (INT\_PTR)TRUE;  }  break;  }  return (INT\_PTR)FALSE;  } |

# 扩展,创建一个项目作为复习用,是一个win32项目

|  |
| --- |
|  |

## win-app.cpp的源码如下

|  |
| --- |
| // win-app.cpp : 定义应用程序的入口点。  //  #include "framework.h"  #include "win-app.h"  #define MAX\_LOADSTRING 100  // 全局变量:  HINSTANCE hInst; // 当前实例  WCHAR szTitle[MAX\_LOADSTRING]; // 标题栏文本  WCHAR szWindowClass[MAX\_LOADSTRING]; // 主窗口类名  // 此代码模块中包含的函数的前向声明:  ATOM MyRegisterClass(HINSTANCE hInstance);  BOOL InitInstance(HINSTANCE, int);  LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);  INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);  int APIENTRY wWinMain(\_In\_ HINSTANCE hInstance,  \_In\_opt\_ HINSTANCE hPrevInstance,  \_In\_ LPWSTR lpCmdLine,  \_In\_ int nCmdShow)  {  UNREFERENCED\_PARAMETER(hPrevInstance);  UNREFERENCED\_PARAMETER(lpCmdLine);  // TODO: 在此处放置代码。  // 初始化全局字符串  LoadStringW(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);  LoadStringW(hInstance, IDC\_WINAPP, szWindowClass, MAX\_LOADSTRING);  MyRegisterClass(hInstance);  // 执行应用程序初始化:  if (!InitInstance (hInstance, nCmdShow))  {  return FALSE;  }  HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_WINAPP));  MSG msg;  // 主消息循环:  while (GetMessage(&msg, nullptr, 0, 0))  {  if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))  {  TranslateMessage(&msg);  DispatchMessage(&msg);  }  }  return (int) msg.wParam;  }  //  // 函数: MyRegisterClass()  //  // 目标: 注册窗口类。  //  ATOM MyRegisterClass(HINSTANCE hInstance)  {  WNDCLASSEXW wcex;  wcex.cbSize = sizeof(WNDCLASSEX);  wcex.style = CS\_HREDRAW | CS\_VREDRAW;  wcex.lpfnWndProc = WndProc;  wcex.cbClsExtra = 0;  wcex.cbWndExtra = 0;  wcex.hInstance = hInstance;  wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_WINAPP));  wcex.hCursor = LoadCursor(nullptr, IDC\_ARROW);  wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW+1);  wcex.lpszMenuName = MAKEINTRESOURCEW(IDC\_WINAPP);  wcex.lpszClassName = szWindowClass;  wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));  return RegisterClassExW(&wcex);  }  //  // 函数: InitInstance(HINSTANCE, int)  //  // 目标: 保存实例句柄并创建主窗口  //  // 注释:  //  // 在此函数中，我们在全局变量中保存实例句柄并  // 创建和显示主程序窗口。  //  BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)  {  hInst = hInstance; // 将实例句柄存储在全局变量中  HWND hWnd = CreateWindowW(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,  CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, nullptr, nullptr, hInstance, nullptr);  if (!hWnd)  {  return FALSE;  }  ShowWindow(hWnd, nCmdShow);  UpdateWindow(hWnd);  return TRUE;  }  //  // 函数: WndProc(HWND, UINT, WPARAM, LPARAM)  //  // 目标: 处理主窗口的消息。  //  // WM\_COMMAND - 处理应用程序菜单  // WM\_PAINT - 绘制主窗口  // WM\_DESTROY - 发送退出消息并返回  //  //  LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)  {  HDC hdc;  HBITMAP hbm;  HBRUSH brush;  switch (message)  {  case WM\_LBUTTONDOWN:  hdc = GetDC(hWnd);  hbm = LoadBitmap(hInst, MAKEINTRESOURCE(IDB\_BITMAP2));  brush = CreatePatternBrush(hbm);  RECT rc;  GetClientRect(hWnd, &rc);  FillRect(hdc, &rc, brush);  ReleaseDC(hWnd, hdc);  break;  case WM\_COMMAND:  {  int wmId = LOWORD(wParam);  // 分析菜单选择:  switch (wmId)  {  case IDM\_ABOUT:  DialogBox(hInst, MAKEINTRESOURCE(IDD\_ABOUTBOX), hWnd, About);  break;  case IDM\_EXIT:  DestroyWindow(hWnd);  break;  default:  return DefWindowProc(hWnd, message, wParam, lParam);  }  }  break;  case WM\_PAINT:  {  PAINTSTRUCT ps;  HDC hdc = BeginPaint(hWnd, &ps);  // TODO: 在此处添加使用 hdc 的任何绘图代码...  HBITMAP hbm;  //BITMAP bm;  HBRUSH brush;  RECT rc;  GetClientRect(hWnd, &rc);  hbm = LoadBitmap(hInst, MAKEINTRESOURCE(IDB\_BITMAP1));  //GetObject(hbm, sizeof(BITMAP), &bm);  //HDC hMemDc = CreateCompatibleDC(hdc);  //SelectObject(hMemDc, hbm);  ////BitBlt(hdc, 0, 0, bm.bmWidth, bm.bmHeight, hMemDc, 0, 0, SRCCOPY);  //StretchBlt(hdc, 0, 0, rc.right - rc.left, rc.bottom - rc.top, hMemDc, 0, 0, bm.bmWidth, bm.bmHeight, SRCCOPY);  brush = CreatePatternBrush(hbm);  FillRect(hdc, &rc, brush);  EndPaint(hWnd, &ps);  //ReleaseDC(hWnd,hMemDc);  }  break;  case WM\_DESTROY:  PostQuitMessage(0);  break;  default:  return DefWindowProc(hWnd, message, wParam, lParam);  }  return 0;  }  // “关于”框的消息处理程序。  INT\_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)  {  UNREFERENCED\_PARAMETER(lParam);  switch (message)  {  case WM\_INITDIALOG:  return (INT\_PTR)TRUE;  case WM\_COMMAND:  if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)  {  EndDialog(hDlg, LOWORD(wParam));  return (INT\_PTR)TRUE;  }  break;  }  return (INT\_PTR)FALSE;  } |

### 效果

|  |
| --- |
|  |
|  |

# 可以给程序添加对鼠标左键按下的消息响应代码,当我们按下鼠标左键,会用位图画刷来填充窗口,也可以中WM\_PAINT的信息响应代码里面会用位图画刷来填充窗口或者贴雅虎位图到窗口客户区

# 这一节的学习到此为止