# 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;  } |

# 这一节的学习到此为止