# 学习大纲

|  |
| --- |
|  |

## SetTextColor函数的用法

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **SetTextColor** 函数将指定设备上下文的文本颜色设置为指定颜色。 语法 C++复制  COLORREF SetTextColor(  [in] HDC hdc,  [in] COLORREF color  ); 参数 [in] hdc  设备上下文的句柄。  [in] color  文本的颜色。 返回值 如果函数成功，则返回值是上一个文本颜色的颜色引用，作为 [COLORREF](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/colorref) 值。  如果函数失败，则返回值为 CLR\_INVALID。 注解 文本颜色用于绘制 [TextOut](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-textouta) 和 [ExtTextOut](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-exttextouta) 函数编写的每个字符的脸。 文本颜色还用于将位图从颜色转换为单色，反之亦然。 示例 有关示例，请参阅 [使用菜单中](https://learn.microsoft.com/zh-cn/windows/desktop/menurc/using-menus)的“设置 Menu-Item 文本字符串的字体”。 要求  |  |  | | --- | --- | | **标头** | wingdi.h (包括 Windows.h) | | **Library** | Gdi32.lib | | **DLL** | Gdi32.dll |  另请参阅 [BitBlt](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-bitblt)  [COLORREF](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/colorref)  [ExtTextOut](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-exttextouta)  [字体和文本函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/font-and-text-functions)  [字体和文本概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/fonts-and-text)  [GetTextColor](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-gettextcolor)  [RGB](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-rgb)  [SetBkColor](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-setbkcolor)  [StretchBlt](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-stretchblt)  [TextOut](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-textouta) |

## SetBKColor函数的用法

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **SetBkColor** 函数将当前背景色设置为指定的颜色值，如果设备无法表示指定的颜色值，则设置为最近的物理颜色。 语法 C++复制  COLORREF SetBkColor(  [in] HDC hdc,  [in] COLORREF color  ); 参数 [in] hdc  设备上下文的句柄。  [in] color  新的背景色。 若要创建 [COLORREF](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/colorref) 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，则返回值将以前的背景色指定为 [COLORREF](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/colorref) 值。  如果函数失败，则返回值为 CLR\_INVALID。 注解 此函数填充使用 [CreatePen](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createpen) 函数创建的笔绘制的带样式的线条之间的间隙;它不会填充使用 [ExtCreatePen](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-extcreatepen) 函数创建的笔绘制的带样式的线条之间的间隙。 **SetBkColor** 函数还设置 [TextOut](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-textouta) 和 [ExtTextOut](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-exttextouta) 的背景色。  如果背景模式为 OPAQUE，则背景色用于填充带样式的线条、画笔中阴影线和字符单元格之间的间隙。 将位图从颜色转换为单色时，也会使用背景色，反之亦然。 示例 有关示例，请参阅 Using Menu 中的“Owner-Drawn [菜单](https://learn.microsoft.com/zh-cn/windows/desktop/menurc/using-menus)项示例”。 要求  |  |  | | --- | --- | | **标头** | wingdi.h (包括 Windows.h) | | **Library** | Gdi32.lib | | **DLL** | Gdi32.dll |  另请参阅 [COLORREF](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/colorref)  [CreatePen](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createpen)  [ExtCreatePen](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-extcreatepen)  [GetBKColor](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getbkcolor)  [GetBkMode](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getbkmode)  [绘制和绘制函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/painting-and-drawing-functions)  [绘画和绘图概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/painting-and-drawing)  [SetBkMode](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-setbkmode) |

## SetBkMode函数的用法

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SetBkMode** 函数设置指定设备上下文的后台混合模式。 背景混合模式用于文本、阴影画笔和不是实线的笔样式。 语法 int SetBkMode(  [in] HDC hdc,  [in] int mode  ); 参数 [in] hdc  设备上下文的句柄。  [in] mode  背景模式。 此参数的取值可为下列值之一：  展开表   |  |  | | --- | --- | | **值** | **含义** | | **OPAQUE** | 在绘制文本、阴影画笔或触笔之前，使用当前背景色填充背景。 | | **TRANSPARENT** | 背景保持不变。 |  返回值 如果函数成功，则返回值指定以前的后台模式。  如果函数失败，则返回值为零。 注解 **SetBkMode** 函数会影响使用 [CreatePen](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createpen) 函数创建的笔绘制的线条的线条样式。 **SetBkMode** 不会影响使用 [ExtCreatePen](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-extcreatepen) 函数创建的笔绘制的线条。 示例 若要了解如何使阴影画笔的背景透明或不透明，请参阅 [CreateHatchBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createhatchbrush) 主题中显示的示例。  下一个示例绘制一个字符串 36 次，每次逆时针旋转 10 度。 它还会将背景模式设置为透明，以使文本可见。  C++复制  #include "strsafe.h"  LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)  {  int wmId, wmEvent;  PAINTSTRUCT ps;  HDC hdc;  switch (message)  {    case WM\_PAINT:  {  hdc = BeginPaint(hWnd, &ps);  RECT rc;  int angle;  HGDIOBJ hfnt, hfntPrev;  WCHAR lpszRotate[22] = TEXT("String to be rotated.");  HRESULT hr;  size\_t pcch = 22;    // Allocate memory for a LOGFONT structure.    PLOGFONT plf = (PLOGFONT) LocalAlloc(LPTR, sizeof(LOGFONT));      // Specify a font typeface name and weight.    hr = StringCchCopy(plf->lfFaceName, 6, TEXT("Arial"));  if (FAILED(hr))  {  // TODO: write error handler  }  plf->lfWeight = FW\_NORMAL;    // Retrieve the client-rectangle dimensions.    GetClientRect(hWnd, &rc);    // Set the background mode to transparent for the  // text-output operation.    SetBkMode(hdc, TRANSPARENT);    // Draw the string 36 times, rotating 10 degrees  // counter-clockwise each time.    for (angle = 0; angle < 3600; angle += 100)  {  plf->lfEscapement = angle;  hfnt = CreateFontIndirect(plf);  hfntPrev = SelectObject(hdc, hfnt);    //  // The StringCchLength call is fitted to the lpszRotate string  //  hr = StringCchLength(lpszRotate, 22, &pcch);  if (FAILED(hr))  {  // TODO: write error handler  }  TextOut(hdc, rc.right / 2, rc.bottom / 2,  lpszRotate, pcch);  SelectObject(hdc, hfntPrev);  DeleteObject(hfnt);  }    // Reset the background mode to its default.    SetBkMode(hdc, OPAQUE);    // Free the memory allocated for the LOGFONT structure.    LocalFree((LOCALHANDLE) plf);  EndPaint(hWnd, &ps);  break;  }  case WM\_DESTROY:  PostQuitMessage(0);  break;  default:  return DefWindowProc(hWnd, message, wParam, lParam);  }  return 0;  } 要求  |  |  | | --- | --- | | **标头** | wingdi.h (包括 Windows.h) | | **Library** | Gdi32.lib | | **DLL** | Gdi32.dll |  另请参阅 [CreatePen](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createpen)  [ExtCreatePen](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-extcreatepen)  [GetBkMode](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getbkmode)  [绘制和绘制函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/painting-and-drawing-functions)  [绘画和绘图概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/painting-and-drawing) |

## SetTextCharacterExtra函数的用法

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **SetTextCharacterExtra** 函数设置字符间间距。 当系统写入文本行时，会将字符间间距添加到每个字符（包括分隔符）。 语法 int SetTextCharacterExtra(  [in] HDC hdc,  [in] int extra  ); 参数 [in] hdc  设备上下文的句柄。  [in] extra  要添加到每个字符的额外空间量（以逻辑单元为单位）。 如果当前映射模式未MM\_TEXT，则 *nCharExtra* 参数将转换并舍入到最接近的像素。 返回值 如果函数成功，则返回值为前面的字符间间距。  如果函数失败，则返回值0x80000000。 注解 此函数主要是为了与现有应用程序兼容而受支持。 新应用程序通常应避免调用此函数，因为它与需要文本整形的复杂脚本 (脚本不兼容;阿拉伯文脚本就是此) 的一个示例。  建议的方法是，应用程序应调用 [ExtTextOut](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-exttextouta) 并使用其 *lpDx* 参数提供宽度，而不是调用此函数，然后调用 [TextOut](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-textouta)。 要求  |  |  | | --- | --- | | **标头** | wingdi.h (包括 Windows.h) | | **Library** | Gdi32.lib | | **DLL** | Gdi32.dll |  另请参阅 [DrawText](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-drawtext)  [字体和文本函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/font-and-text-functions)  [字体和文本概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/fonts-and-text)  [GetTextCharacterExtra](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-gettextcharacterextra)  [TextOut](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-textouta) |

## 扩展：GetSysColor函数的用法

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 检索指定显示元素的当前颜色。 显示元素是窗口的组成部分，也是显示在系统显示屏幕上的显示器。 语法 C++复制  DWORD GetSysColor(  [in] int nIndex  ); 参数 [in] nIndex  类型： **int**  要检索其颜色的显示元素。 此参数的取值可为下列值之一：  展开表   |  |  | | --- | --- | | **值** | **含义** | | **COLOR\_3DDKSHADOW**  21 | 三维显示元素的深色阴影。  **Windows 10或更高版本：**不支持此值。 | | **COLOR\_3DFACE**  15 | 三维显示元素和对话框背景的人脸颜色。 | | **COLOR\_3DHIGHLIGHT**  20 | 三维显示元素的突出显示颜色 (面向光源的边缘。)  **Windows 10或更高版本：**不支持此值。 | | **COLOR\_3DHILIGHT**  20 | 三维显示元素的突出显示颜色 (面向光源的边缘。)  **Windows 10或更高版本：**不支持此值。 | | **COLOR\_3DLIGHT**  22 | 三维显示元素的浅色 (面向光源的边缘。)  **Windows 10或更高版本：**不支持此值。 | | **COLOR\_3DSHADOW**  16 | 三维显示元素的阴影颜色 (面向远离光源的边缘) 。  **Windows 10或更高版本：**不支持此值。 | | **COLOR\_ACTIVEBORDER**  10 | 活动窗口边框。  **Windows 10或更高版本：**不支持此值。 | | **COLOR\_ACTIVECAPTION**  2 | 活动窗口标题栏。  关联的前景色为 **COLOR\_CAPTIONTEXT**。  如果启用了渐变效果，则指定活动窗口标题栏的颜色渐变中的左侧颜色。  **Windows 10或更高版本：**不支持此值。 | | **COLOR\_APPWORKSPACE**  12 | 多个文档界面 (MDI) 应用程序的背景色。  **Windows 10或更高版本：**不支持此值。 | | **COLOR\_BACKGROUND**  1 | 桌面设备。  **Windows 10或更高版本：**不支持此值。 | | **COLOR\_BTNFACE**  15 | 三维显示元素和对话框背景的人脸颜色。 关联的前景色 **COLOR\_BTNTEXT**。  **Windows 10或更高版本：**不支持此值。 | | **COLOR\_BTNHIGHLIGHT**  20 | 三维显示元素的突出显示颜色 (面向光源的边缘。)  **Windows 10或更高版本：**不支持此值。 | | **COLOR\_BTNHILIGHT**  20 | 三维显示元素的突出显示颜色 (面向光源的边缘。)  **Windows 10或更高版本：**不支持此值。 | | **COLOR\_BTNSHADOW**  16 | 三维显示元素的阴影颜色 (面向远离光源的边缘) 。  **Windows 10或更高版本：**不支持此值。 | | **COLOR\_BTNTEXT**  18 | 按下按钮上的文本。 关联的背景色为COLOR\_BTNFACE。 | | **COLOR\_CAPTIONTEXT**  9 | 描述文字、大小框和滚动条箭头框中的文本。 关联的背景色COLOR\_ACTIVECAPTION。  **Windows 10或更高版本：**不支持此值。 | | **COLOR\_DESKTOP**  1 | 桌面设备。  **Windows 10或更高版本：**不支持此值。 | | **COLOR\_GRADIENTACTIVECAPTION**  27 | 活动窗口标题栏颜色渐变中的右侧颜色。 COLOR\_ACTIVECAPTION指定左侧颜色。 将 SPI\_GETGRADIENTCAPTIONS 与 [SystemParametersInfo](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-systemparametersinfoa) 函数配合使用，以确定是否启用了渐变效果。  **Windows 10或更高版本：**不支持此值。 | | **COLOR\_GRADIENTINACTIVECAPTION**  28 | 非活动窗口标题栏颜色渐变中的右侧颜色。 COLOR\_INACTIVECAPTION指定左侧颜色。  **Windows 10或更高版本：**不支持此值。 | | **COLOR\_GRAYTEXT**  17 | 已禁用) 文本的灰色 (。 如果当前显示驱动程序不支持纯灰色，则此颜色设置为 0。 | | **COLOR\_HIGHLIGHT**  13 | 在 控件中选择的项 () 。 关联的前景色COLOR\_HIGHLIGHTTEXT。 | | **COLOR\_HIGHLIGHTTEXT**  14 | 在 控件中选择的项 () 的文本。 关联的背景色为COLOR\_HIGHLIGHT。 | | **COLOR\_HOTLIGHT**  26 | 超链接或热跟踪项的颜色。 关联的背景色COLOR\_WINDOW。 | | **COLOR\_INACTIVEBORDER**  11 | 非活动窗口边框。  **Windows 10或更高版本：**不支持此值。 | | **COLOR\_INACTIVECAPTION**  3 | 非活动窗口描述文字。  关联的前景色COLOR\_INACTIVECAPTIONTEXT。  如果启用了渐变效果，则指定非活动窗口标题栏的颜色渐变中的左侧颜色。  **Windows 10或更高版本：**不支持此值。 | | **COLOR\_INACTIVECAPTIONTEXT**  19 | 非活动描述文字中的文本颜色。 关联的背景色COLOR\_INACTIVECAPTION。  **Windows 10或更高版本：**不支持此值。 | | **COLOR\_INFOBK**  24 | 工具提示控件的背景色。 关联的前景色COLOR\_INFOTEXT。  **Windows 10或更高版本：**不支持此值。 | | **COLOR\_INFOTEXT**  23 | 工具提示控件的文本颜色。 关联的背景色COLOR\_INFOBK。  **Windows 10或更高版本：**不支持此值。 | | **COLOR\_MENU**  4 | 菜单背景。 关联的前景色COLOR\_MENUTEXT。  **Windows 10或更高版本：**不支持此值。 | | **COLOR\_MENUHILIGHT**  29 | 当菜单显示为平面菜单时用于突出显示菜单项的颜色 (请参阅 [SystemParametersInfo](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-systemparametersinfoa)) 。 突出显示的菜单项带有COLOR\_HIGHLIGHT。  **Windows 2000，Windows 10或更高版本：**不支持此值。 | | **COLOR\_MENUBAR**  30 | 菜单显示为平面菜单时菜单栏的背景色 (请参阅 [SystemParametersInfo](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-systemparametersinfoa)) 。 但是，COLOR\_MENU继续指定菜单弹出窗口的背景色。  **Windows 2000，Windows 10或更高版本：**不支持此值。 | | **COLOR\_MENUTEXT**  7 | 菜单中的文本。 关联的背景色COLOR\_MENU。  **Windows 10或更高版本：**不支持此值。 | | **COLOR\_SCROLLBAR**  0 | 滚动条灰色区域。  **Windows 10或更高版本：**不支持此值。 | | **COLOR\_WINDOW**  5 | 窗口背景。 关联的前景色COLOR\_WINDOWTEXT和COLOR\_HOTLITE。 | | **COLOR\_WINDOWFRAME**  6 | 窗口框架。  **Windows 10或更高版本：**不支持此值。 | | **COLOR\_WINDOWTEXT**  8 | 窗口中的文本。 关联的背景色COLOR\_WINDOW。 |  返回值 类型：DWORD  函数返回给定元素的红色、绿色、蓝色 (RGB) 颜色值。  如果 *nIndex* 参数范围外，则返回值为零。 由于零也是有效的 RGB 值，因此无法使用 **GetSysColor** 来确定当前平台是否支持系统颜色。 请改用 [GetSysColorBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-getsyscolorbrush) 函数，如果该颜色不受支持，则返回 **NULL** 。 注解 若要显示 RGB 值的分量，请使用 [GetRValue](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getrvalue)、 [GetGValue](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getgvalue) 和 [GetBValue](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getbvalue) 宏。  单色显示器的系统颜色通常解释为灰色阴影。  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。  颜色是大多数用户界面中重要的视觉元素。 有关在应用程序中使用颜色的指南，请参阅 [Color - Win32](https://learn.microsoft.com/zh-cn/windows/win32/uxguide/vis-color) 和 [Color in Windows 11](https://learn.microsoft.com/zh-cn/windows/apps/design/signature-experiences/color)。 Windows 10/11 系统颜色 下表显示了Windows 10和Windows 11中支持的值，以及Windows 11水生对比度主题中的颜色值。  展开表   | **颜色样本** | **说明** | | --- | --- | | https://learn.microsoft.com/zh-cn/windows/win32/api/winuser/images/sys-color/aquatic-color-window.png | **COLOR\_WINDOW** 页面、窗格、弹出窗口和窗口的背景。  与 COLOR\_WINDOWTEXT 配对 | | https://learn.microsoft.com/zh-cn/windows/win32/api/winuser/images/sys-color/aquatic-color-windowtext.png | COLOR\_WINDOWTEXT标题、正文副本、列表、占位符文本、应用和窗口边框，以及无法与之交互的任何 UI。  与COLOR\_WINDOW配对 | | https://learn.microsoft.com/zh-cn/windows/win32/api/winuser/images/sys-color/aquatic-color-hotlight.png | **COLOR\_HOTLIGHT** 链接。  与COLOR\_WINDOW配对 | | https://learn.microsoft.com/zh-cn/windows/win32/api/winuser/images/sys-color/aquatic-color-graytext.png | COLOR\_GRAYTEXT非活动 (禁用) UI。  与COLOR\_WINDOW配对 | | https://learn.microsoft.com/zh-cn/windows/win32/api/winuser/images/sys-color/aquatic-color-highlighttext.png | COLOR\_HIGHLIGHTTEXT处于选定状态、与悬停 (交互的文本或 UI 的前景色、按下) 或正在进行中。  与COLOR\_HIGHLIGHT配对 | | https://learn.microsoft.com/zh-cn/windows/win32/api/winuser/images/sys-color/aquatic-color-highlight.png | **COLOR\_HIGHLIGHT** 处于选定状态、与 (悬停交互、按下) 或正在进行的 UI 的背景色或主题色。  与COLOR\_HIGHLIGHTTEXT配对 | | https://learn.microsoft.com/zh-cn/windows/win32/api/winuser/images/sys-color/aquatic-color-btntext.png | **COLOR\_BTNTEXT** 按钮和可以与之交互的任何 UI 的前景色。  与COLOR\_3DFACE配对 | | https://learn.microsoft.com/zh-cn/windows/win32/api/winuser/images/sys-color/aquatic-color-3dface.png | **COLOR\_3DFACE** 按钮和可以与之交互的任何 UI 的背景色。  与COLOR\_BTNTEXT配对 |   这些图像显示在设置为 **COLOR\_WINDOW**的背景上使用时颜色的显示方式。  展开表   | **示例** | **值** | | --- | --- | | 使用窗口文本颜色显示文本的窗口。 | **COLOR\_WINDOWTEXT** | | 具有使用热光颜色的超链接文本的窗口。 | **COLOR\_HOTLIGHT** | | 使用灰色文本颜色显示非活动文本的窗口。 | **COLOR\_GRAYTEXT** | | 一个窗口，其中包含使用突出显示颜色上的突出显示文本颜色的文本。 | **COLOR\_HIGHLIGHTTEXT + HIGHLIGHT** | | 一个窗口，其中包含使用 3d 人脸颜色的按钮文本和按钮文本的按钮文本。 | **COLOR\_BTNTEXT + COLOR\_3DFACE** |  示例 有关示例，请参阅 [SetSysColors](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-setsyscolors)。 要求  |  |  | | --- | --- | | **标头** | winuser.h (包括 Windows.h) | | **Library** | User32.lib | | **DLL** | User32.dll |  另请参阅 [CreateSolidBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createsolidbrush)  [GetSysColorBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-getsyscolorbrush)  [SetSysColors](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-setsyscolors)  [SystemParametersInfo](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-systemparametersinfoa) |

# 演练

## 1.新建一个win32的项目，取名：Lesson39-text-context-demo

|  |
| --- |
|  |

## 2.在窗口过程函数里面找到WM\_PAINT消息的处理代码，默认窗口背景色是白色，文字颜色是黑色，我们先来修改一下文本颜色，然后输出一行文本

|  |
| --- |
|  |
|  |

### 效果

|  |
| --- |
|  |

## 3.我们现在设置一下背景颜色和一个新的文本颜色，然后再输出一次文本

|  |
| --- |
|  |

### 效果

|  |
| --- |
|  |

## 4.然后我们可以设置新的背景色和新的文本颜色，然后白背景模式设置为透明

|  |
| --- |
|  |

### 效果

|  |
| --- |
|  |

## 5.我们在下面把背景模式设置为不透明，然后再输出一次文本

|  |
| --- |
|  |

### 效果

|  |
| --- |
|  |

## 6.然后我们设置一下字符间距，然后再输出一次文本

|  |
| --- |
|  |

### 效果

|  |
| --- |
|  |

## 7.我们也可以设置窗口的背景颜色，这个需要在设置窗口类结构体的时候设置，我们可以在COLOR\_WINDOW上面大家右键-》转到定义，可以看到有很多预先定义好的颜色

|  |
| --- |
|  |

## 8.我们这里选择一个叫做COLOR\_BTNFACE的颜色

|  |
| --- |
|  |

### 效果如下

|  |
| --- |
|  |

## 9.我们也可以自己创建一个画刷，然后设置到窗口中，先定义一个背景画刷的全局变量

|  |
| --- |
|  |

### 然后在注册窗口类的函数里面创建画刷并且把他设置为窗口类的背景

|  |
| --- |
|  |

### 效果

|  |
| --- |
|  |

### 不过，虽然那个设置，微软建议我们还是使用微软预定义好的颜色，我们把背景色改回来

|  |
| --- |
|  |

## 10.微软也建议我们使用标准的文本颜色，如果你不知道是什么颜色，可以使用GetSysColor(COLOR\_WINDOWTEXT)来获取

|  |
| --- |
|  |

## 11.也可以把背景颜色也设置为标准颜色

|  |
| --- |
|  |

## 12.另外，调用SetTextCharacterExtra时把第二个参数设置为0，windows就会使用默认的字符间距来设置

|  |
| --- |
|  |

## 13.如果想当用户修改了windows的颜色方案时，你的应用程序的颜色也会改变，你需要处理一个叫做WM\_SYSCOLORCHANGE消息

|  |
| --- |
|  |

# 这一节的学习到此为止，完整代码如下

## Lesson39-text-context-demo.cpp

|  |
| --- |
| // Lesson39-text-context-demo.cpp : 定义应用程序的入口点。  //  #include "stdafx.h"  #include "Lesson39-text-context-demo.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  HBRUSH hBkBrush; //自定义背景画刷  // 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\_LESSON39TEXTCONTEXTDEMO, 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\_LESSON39TEXTCONTEXTDEMO));  // 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;    //创建窗口背景画刷  hBkBrush = CreateSolidBrush(RGB(255,100,200));  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\_LESSON39TEXTCONTEXTDEMO));  wcex.hCursor = LoadCursor(NULL, IDC\_ARROW);  wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW+1);  //wcex.hbrBackground = (HBRUSH)(COLOR\_BTNFACE+1);//修改窗口的背景颜色  //wcex.hbrBackground = hBkBrush;//使用自己的背景画刷  wcex.lpszMenuName = MAKEINTRESOURCE(IDC\_LESSON39TEXTCONTEXTDEMO);  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;  TCHAR ptMsg[] = \_T("Hello Win32 Application!!!");  COLORREF color,oldColor;  COLORREF bkColor,oldBkColor;    RECT rc;  int OldExtra = 0;  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:  GetClientRect(hWnd,&rc);  hdc = BeginPaint(hWnd, &ps);  // TODO: 在此添加任意绘图代码...  //设置文本颜色  color = RGB(255,155,255);  oldColor = SetTextColor(hdc,color);  TextOut(hdc,50,100,ptMsg,lstrlen(ptMsg));  SetTextColor(hdc,oldColor);//输出文本后需要设置回原来的颜色  //设置背景颜色  bkColor = RGB(125,125,125);  oldBkColor = SetBkColor(hdc,bkColor);  //设置新的字符颜色  color = RGB(125,255,125);  oldColor = SetTextColor(hdc,color);  TextOut(hdc,50,130,ptMsg,lstrlen(ptMsg));  SetTextColor(hdc,oldColor);//输出文本后需要设置回原来的颜色  SetBkColor(hdc,oldBkColor);//输出文本后需要把背景颜色设置回原来的颜色  //设置背景模式为透明背景  SetBkMode(hdc,TRANSPARENT);  //设置背景颜色  bkColor = RGB(200,200,200);  oldBkColor = SetBkColor(hdc,bkColor);  //设置新的字符颜色  color = RGB(80,255,255);  oldColor = SetTextColor(hdc,color);  TextOut(hdc,50,160,ptMsg,lstrlen(ptMsg));  //把背景设置为不透明  SetBkMode(hdc,OPAQUE);  TextOut(hdc,50,190,ptMsg,lstrlen(ptMsg));  //设置字符间距，注意在这个版本的vs中，变量需要先声明，再使用。  OldExtra = SetTextCharacterExtra(hdc,15);  TextOut(hdc,50,220,ptMsg,lstrlen(ptMsg));  SetTextColor(hdc,oldColor);//输出文本后需要设置回原来的颜色  SetBkColor(hdc,oldBkColor);//输出文本后需要把背景颜色设置回原来的颜色  SetTextCharacterExtra(hdc,OldExtra);//把压力的字符间距设置回来  EndPaint(hWnd, &ps);  break;  case WM\_SYSCOLORCHANGE:  InvalidateRect(hWnd,NULL,TRUE);  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;  } |