# 学习大纲

|  |
| --- |
|  |

## CHOOSEFONT结构体

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 包含 [ChooseFont](https://learn.microsoft.com/zh-cn/previous-versions/windows/desktop/legacy/ms646914(v=vs.85)) 函数用于初始化 **字体** 对话框的信息。 用户关闭对话框后，系统会返回有关此结构中用户选择的信息。 语法 C++复制  typedef struct tagCHOOSEFONTA {  DWORD lStructSize;  HWND hwndOwner;  HDC hDC;  LPLOGFONTA lpLogFont;  INT iPointSize;  DWORD Flags;  COLORREF rgbColors;  LPARAM lCustData;  LPCFHOOKPROC lpfnHook;  LPCSTR lpTemplateName;  HINSTANCE hInstance;  LPSTR lpszStyle;  WORD nFontType;  WORD \_\_\_MISSING\_ALIGNMENT\_\_;  INT nSizeMin;  INT nSizeMax;  } CHOOSEFONTA; 成员 lStructSize  类型：DWORD  结构的长度（以字节为单位）。  hwndOwner  类型：**HWND**  拥有对话框的窗口的句柄。 此成员可以是任何有效的窗口句柄，如果对话框没有所有者，则它可以为 **NULL** 。  hDC  类型： **HDC**  [ChooseFont](https://learn.microsoft.com/zh-cn/previous-versions/windows/desktop/legacy/ms646914(v=vs.85)) 函数会忽略此成员。  **Windows Vista 和 Windows XP/2000：**打印机的设备上下文或信息上下文的句柄，其字体将在对话框中列出。 仅当 **Flags** 成员指定 **CF\_PRINTERFONTS** 或 **CF\_BOTH** 标志时，才使用此成员;否则，将忽略此成员。  lpLogFont  类型： **LPLOGFONT**  指向 [LOGFONT](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-logfonta) 结构的指针。 如果在 **Flags** 成员中设置 **CF\_INITTOLOGFONTSTRUCT** 标志并初始化其他成员，[则 ChooseFont](https://learn.microsoft.com/zh-cn/previous-versions/windows/desktop/legacy/ms646914(v=vs.85)) 函数会使用与 **LOGFONT** 成员匹配的字体初始化对话框。 如果用户单击“ **确定”** 按钮， **ChooseFont** 会根据用户的选择设置 **LOGFONT** 结构的成员。  iPointSize  类型： **INT**  所选字体的大小，以磅的 1/10 为单位。 [ChooseFont](https://learn.microsoft.com/zh-cn/previous-versions/windows/desktop/legacy/ms646914(v=vs.85)) 函数在用户关闭对话框后设置此值。  Flags  类型：DWORD  可用于初始化 **“字体** ”对话框的一组位标志。 当对话框返回时，它将设置这些标志来指示用户输入。 此成员可以是以下一个或多个值。  展开表   |  |  | | --- | --- | | **Value** | **含义** | | **CF\_APPLY**  0x00000200L | 使对话框显示 **“应用** ”按钮。 应提供挂钩过程来处理“**应用**”按钮[WM\_COMMAND](https://learn.microsoft.com/zh-cn/windows/desktop/menurc/wm-command)消息。 挂钩过程可以将 [WM\_CHOOSEFONT\_GETLOGFONT](https://learn.microsoft.com/zh-cn/windows/desktop/dlgbox/wm-choosefont-getlogfont) 消息发送到对话框，以检索包含当前字体选择的结构的地址。 | | **CF\_ANSIONLY**  0x00000400L | 此标志已过时。 若要将字体选择限制为除使用 OEM 或符号字符集的脚本以外的所有脚本，请使用 **CF\_SCRIPTSONLY**。 若要获取原始**CF\_ANSIONLY**行为，请使用 **CF\_SELECTSCRIPT** 并在 **lpLogFont** 指向的 [LOGFONT](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-logfonta) 结构的 **lfCharSet** 成员中指定**ANSI\_CHARSET**。 | | **CF\_BOTH**  0x00000003 | 对于字体枚举，将忽略此标志。  **Windows Vista 和 Windows XP/2000：**使对话框列出可用的打印机和屏幕字体。 **hDC** 成员是设备上下文或与打印机关联的信息上下文的句柄。 此标志是 **CF\_SCREENFONTS** 和 **CF\_PRINTERFONTS** 标志的组合。 | | **CF\_EFFECTS**  0x00000100L | 使对话框显示允许用户指定删除线、下划线和文本颜色选项的控件。 如果设置了此标志，则可以使用 **rgbColors** 成员指定初始文本颜色。 可以使用 **lpLogFont** 指向的结构的 **lfStrikeOut** 和 **lfUnderline** 成员指定删除线的初始设置和下划线检查框。 [ChooseFont](https://learn.microsoft.com/zh-cn/previous-versions/windows/desktop/legacy/ms646914(v=vs.85)) 可以使用这些成员返回用户的选择。 | | **CF\_ENABLEHOOK**  0x00000008L | 启用此结构的 **lpfnHook** 成员中指定的挂钩过程。 | | **CF\_ENABLETEMPLATE**  0x00000010L | 指示 **hInstance** 和 **lpTemplateName** 成员指定要用于替代默认模板的对话框模板。 | | **CF\_ENABLETEMPLATEHANDLE**  0x00000020L | 指示 **hInstance** 成员标识包含预加载对话框模板的数据块。 如果指定了此标志，系统会忽略 **lpTemplateName** 成员。 | | **CF\_FIXEDPITCHONLY**  0x00004000L | [ChooseFont](https://learn.microsoft.com/zh-cn/previous-versions/windows/desktop/legacy/ms646914(v=vs.85)) 应枚举并允许仅选择固定音调字体。 | | **CF\_FORCEFONTEXIST**  0x00010000L | 如果用户尝试选择对话框中未列出的字体或样式，[ChooseFont](https://learn.microsoft.com/zh-cn/previous-versions/windows/desktop/legacy/ms646914(v=vs.85)) 应指示错误条件。 | | **CF\_INACTIVEFONTS**  0x02000000L | [ChooseFont](https://learn.microsoft.com/zh-cn/previous-versions/windows/desktop/legacy/ms646914(v=vs.85)) 还应另外显示设置为“在字体中隐藏”的字体控制面板。  **Windows Vista 和 Windows XP/2000：**此标志在 Windows 7 之前不受支持。 | | **CF\_INITTOLOGFONTSTRUCT**  0x00000040L | [ChooseFont](https://learn.microsoft.com/zh-cn/previous-versions/windows/desktop/legacy/ms646914(v=vs.85)) 应使用 **lpLogFont** 成员指向的结构来初始化对话框控件。 | | **CF\_LIMITSIZE**  0x00002000L | [ChooseFont](https://learn.microsoft.com/zh-cn/previous-versions/windows/desktop/legacy/ms646914(v=vs.85)) 应仅选择 **nSizeMin** 和 **nSizeMax** 成员指定的范围内的字号。 | | **CF\_NOOEMFONTS**  0x00000800L | 与 **CF\_NOVECTORFONTS** 标志相同。 | | **CF\_NOFACESEL**  0x00080000L | 使用 [LOGFONT](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-logfonta) 结构初始化对话框控件时，使用此标志可阻止对话框显示字体名称组合框的初始选择。 当没有单个字体名称应用于文本选择时，这非常有用。 | | **CF\_NOSCRIPTSEL**  0x00800000L | 禁用 **“脚本** ”组合框。 设置此标志后，[当 ChooseFont](https://learn.microsoft.com/zh-cn/previous-versions/windows/desktop/legacy/ms646914(v=vs.85)) 返回时，[LOGFONT](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-logfonta) 结构的 **lfCharSet** 成员将设置为 **DEFAULT\_CHARSET**。 此标志仅用于初始化对话框。 | | **CF\_NOSIMULATIONS**  0x00001000L | [ChooseFont](https://learn.microsoft.com/zh-cn/previous-versions/windows/desktop/legacy/ms646914(v=vs.85)) 不应显示或允许选择字体模拟。 | | **CF\_NOSIZESEL**  0x00200000L | 使用结构初始化对话框控件时，使用此标志可阻止对话框显示“ **字体大小** ”组合框的初始选择。 当没有单一字号适用于文本选择时，这非常有用。 | | **CF\_NOSTYLESEL**  0x00100000L | 使用 [LOGFONT](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-logfonta) 结构初始化对话框控件时，使用此标志可阻止对话框显示“ **字体样式** ”组合框的初始选择。 当没有单一字体样式应用于文本选择时，这非常有用。 | | **CF\_NOVECTORFONTS**  0x00000800L | [ChooseFont](https://learn.microsoft.com/zh-cn/previous-versions/windows/desktop/legacy/ms646914(v=vs.85)) 不应允许选择矢量字体。 | | **CF\_NOVERTFONTS**  0x01000000L | 导致“ **字体** ”对话框仅列出水平方向的字体。 | | **CF\_PRINTERFONTS**  0x00000002 | 对于字体枚举，将忽略此标志。  **Windows Vista 和 Windows XP/2000：**使对话框仅列出打印机支持的字体，这些字体与 **hDC** 成员标识的设备上下文或信息上下文相关联。 它还会导致字体类型说明标签显示在“ **字体** ”对话框的底部。 | | **CF\_SCALABLEONLY**  0x00020000L | 指定 [ChooseFont](https://learn.microsoft.com/zh-cn/previous-versions/windows/desktop/legacy/ms646914(v=vs.85)) 应仅允许选择可缩放字体。 可缩放字体包括矢量字体、可缩放打印机字体、TrueType 字体以及按其他技术缩放的字体。 | | **CF\_SCREENFONTS**  0x00000001 | 对于字体枚举，将忽略此标志。  **Windows Vista 和 Windows XP/2000：**使对话框仅列出系统支持的屏幕字体。 | | **CF\_SCRIPTSONLY**  0x00000400L | [ChooseFont](https://learn.microsoft.com/zh-cn/previous-versions/windows/desktop/legacy/ms646914(v=vs.85)) 应允许为所有非 OEM 字符集和符号字符集以及 ANSI 字符集选择字体。 这将取代 **CF\_ANSIONLY** 值。 | | **CF\_SELECTSCRIPT**  0x00400000L | 在输入时指定时，仅显示 [LOGFONT](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-logfonta) 结构的 **lfCharSet** 成员中标识的字符集的字体。 不允许用户更改 **脚本组合框中** 指定的字符集。 | | **CF\_SHOWHELP**  0x00000004L | 使对话框显示“ **帮助** ”按钮。 **hwndOwner** 成员必须指定用于接收用户单击“帮助**”按钮**时对话框发送的 [HELPMSGSTRING](https://learn.microsoft.com/zh-cn/windows/desktop/dlgbox/helpmsgstring) 注册消息的窗口。 | | **CF\_TTONLY**  0x00040000L | [ChooseFont](https://learn.microsoft.com/zh-cn/previous-versions/windows/desktop/legacy/ms646914(v=vs.85)) 应仅枚举并允许选择 TrueType 字体。 | | **CF\_USESTYLE**  0x00000080L | **lpszStyle** 成员是指向包含样式数据的缓冲区的指针，[ChooseFont](https://learn.microsoft.com/zh-cn/previous-versions/windows/desktop/legacy/ms646914(v=vs.85)) 应使用该数据初始化**字体样式**组合框。 当用户关闭对话框时， **ChooseFont** 会将用户选择的样式数据复制到此缓冲区。  **注意**若要全球化应用程序，应使用 **lpLogFont** 指向的 [**LOGFONT**](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-logfonta) 结构的 **lfWeight** 和 **lfItalic** 成员来指定样式。 样式名称可能会根据系统用户界面语言而更改。 | | **CF\_WYSIWYG**  0x00008000L | 已过时。 [ChooseFont](https://learn.microsoft.com/zh-cn/previous-versions/windows/desktop/legacy/ms646914(v=vs.85)) 忽略此标志。  **Windows Vista 和 Windows XP/2000：**[ChooseFont](https://learn.microsoft.com/zh-cn/previous-versions/windows/desktop/legacy/ms646914(v=vs.85)) 应仅允许选择打印机和显示器上可用的字体。 如果指定了此标志，则还应指定 **CF\_SCREENSHOTS** 和 **CF\_PRINTERFONTS**或 **CF\_BOTH** 标志。 |   rgbColors  类型： [**COLORREF**](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/colorref)  如果设置了 **CF\_EFFECTS** 标志， **rgbColors** 将指定初始文本颜色。 [当 ChooseFont](https://learn.microsoft.com/zh-cn/previous-versions/windows/desktop/legacy/ms646914(v=vs.85)) 成功返回时，此成员包含用户选择的文本颜色的 RGB 值。 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。  lCustData  类型：LPARAM  系统传递给 **由 lpfnHook** 成员标识的挂钩过程的应用程序定义数据。 当系统将 [WM\_INITDIALOG](https://learn.microsoft.com/zh-cn/windows/desktop/dlgbox/wm-initdialog) 消息发送到挂钩过程时，消息的 *lParam* 参数是指向创建对话框时指定的 [CHOOSEFONT](https://learn.microsoft.com/zh-cn/previous-versions/windows/desktop/legacy/ms646914(v=vs.85)) 结构的指针。 挂钩过程可以使用此指针获取 **lCustData** 值。  lpfnHook  类型： **LPCFHOOKPROC**  指向 [CFHookProc](https://learn.microsoft.com/zh-cn/windows/desktop/api/commdlg/nc-commdlg-lpcfhookproc) 挂钩过程的指针，该过程可以处理用于对话框的消息。 除非在 **Flags** 成员中设置了**CF\_ENABLEHOOK**标志，否则将忽略此成员。  lpTemplateName  类型： **LPCTSTR**  **hInstance** 成员标识的模块中对话框模板资源的名称。 此模板将替换为标准对话框模板。 对于编号对话框资源， **lpTemplateName** 可以是 [MAKEINTRESOURCE](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-makeintresourcea) 宏返回的值。 除非在 **Flags** 成员中设置了**CF\_ENABLETEMPLATE**标志，否则将忽略此成员。  hInstance  类型： **HINSTANCE**  如果在 **Flags** 成员中设置了**CF\_ENABLETEMPLATEHANDLE**标志，**则 hInstance** 是包含对话框模板的内存对象的句柄。 如果设置了 **CF\_ENABLETEMPLATE** 标志， **则 hInstance** 是包含由 **lpTemplateName** 成员命名的对话框模板的模块的句柄。 如果未设置 **CF\_ENABLETEMPLATEHANDLE** 或 **CF\_ENABLETEMPLATE** ，则忽略此成员。  lpszStyle  类型： **LPTSTR**  样式数据。 如果指定 **了CF\_USESTYLE** 标志， [ChooseFont](https://learn.microsoft.com/zh-cn/previous-versions/windows/desktop/legacy/ms646914(v=vs.85)) 将使用此缓冲区中的数据来初始化 **“字体样式** ”组合框。 当用户关闭对话框时， **ChooseFont** 会将 **“字体样式** ”组合框中的字符串复制到此缓冲区中。  nFontType  类型： **WORD**  [ChooseFont](https://learn.microsoft.com/zh-cn/previous-versions/windows/desktop/legacy/ms646914(v=vs.85)) 返回时所选字体的类型。 此成员可以是以下一个或多个值。  展开表   |  |  | | --- | --- | | **Value** | **含义** | | **BOLD\_FONTTYPE**  0x0100 | 字体粗细为粗体。 此信息在 [LOGFONT](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-logfonta) 结构的 **lfWeight** 成员中重复，等效于 **FW\_BOLD**。 | | **ITALIC\_FONTTYPE**  0x0200 | 设置了斜体字体属性。 此信息在 [LOGFONT](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-logfonta) 结构的 **lfItalic** 成员中重复。 | | **PRINTER\_FONTTYPE**  0x4000 | 字体是打印机字体。 | | **REGULAR\_FONTTYPE**  0x0400 | 字体粗细正常。 此信息在 [LOGFONT](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-logfonta) 结构的 **lfWeight** 成员中重复，等效于 **FW\_REGULAR**。 | | **SCREEN\_FONTTYPE**  0x2000 | 字体是屏幕字体。 | | **SIMULATED\_FONTTYPE**  0x8000 | 该字体由图形设备界面 (GDI) 模拟。 |   \_\_\_MISSING\_ALIGNMENT\_\_  nSizeMin  类型： **INT**  用户可选择的最小磅值。 仅当指定**了 CF\_LIMITSIZE** 标志时，[ChooseFont](https://learn.microsoft.com/zh-cn/previous-versions/windows/desktop/legacy/ms646914(v=vs.85)) 才会识别此成员。  nSizeMax  类型： **INT**  用户可选择的最大磅值。 仅当指定**了 CF\_LIMITSIZE** 标志时，[ChooseFont](https://learn.microsoft.com/zh-cn/previous-versions/windows/desktop/legacy/ms646914(v=vs.85)) 才会识别此成员。 注解 **备注**  commdlg.h 标头将 CHOOSEFONT 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** | commdlg.h |  另请参阅 [ChooseFont](https://learn.microsoft.com/zh-cn/previous-versions/windows/desktop/legacy/ms646914(v=vs.85))  [通用对话框库](https://learn.microsoft.com/zh-cn/windows/desktop/dlgbox/common-dialog-box-library)  **概念性**  [MAKEINTRESOURCE](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-makeintresourcea)  **引用** |

### CHOOSEFONT结构体需要先初始化，再使用

## LOGFONT结构体

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **LOGFONT** 结构定义字体的属性。 语法 C++复制  typedef struct tagLOGFONTA {  LONG lfHeight;  LONG lfWidth;  LONG lfEscapement;  LONG lfOrientation;  LONG lfWeight;  BYTE lfItalic;  BYTE lfUnderline;  BYTE lfStrikeOut;  BYTE lfCharSet;  BYTE lfOutPrecision;  BYTE lfClipPrecision;  BYTE lfQuality;  BYTE lfPitchAndFamily;  CHAR lfFaceName[LF\_FACESIZE];  } LOGFONTA, \*PLOGFONTA, \*NPLOGFONTA, \*LPLOGFONTA; 成员 lfHeight  字体的字符单元格或字符的高度（以逻辑单位为单位）。 字符高度值 (也称为 em height) 是字符单元格高度值减去内部前导值。 字体映射器按以下方式解释 **lfHeight** 中指定的值。  展开表   |  |  | | --- | --- | | **值** | **含义** | | > 0 | 字体映射器将此值转换为设备单位，并将其与可用字体的单元格高度匹配。 | | 0 | 字体映射器在搜索匹配项时使用默认高度值。 | | < 0 | 字体映射器将此值转换为设备单位，并将其绝对值与可用字体的字符高度匹配。 |     对于所有高度比较，字体映射器会查找不超过请求大小的最大字体。  首次使用该字体时，会发生此映射。  对于MM\_TEXT映射模式，可以使用以下公式指定具有指定点大小的字体的高度：  C++复制  lfHeight = -MulDiv(PointSize, GetDeviceCaps(hDC, LOGPIXELSY), 72);  lfWidth  字体中字符的平均宽度（以逻辑单位为单位）。 如果 **lfWidth** 为零，则设备的纵横比将与可用字体的数字化纵横比进行匹配，以查找由差值的绝对值确定的最接近匹配项。  lfEscapement  转义向量与设备的 x 轴之间的角度（以十分之一度为单位）。 转义向量与文本行的基线平行。  当图形模式设置为“GM\_ADVANCED”时，可以独立于字符串字符的方向角度指定字符串的转义角度。  当图形模式设置为GM\_COMPATIBLE时， **lfEscapement** 同时指定转义和方向。 应将 **lfEscapement** 和 **lfOrientation** 设置为相同的值。  lfOrientation  每个字符的基线和设备 x 轴之间的角度（以十分之一度为单位）。  lfWeight  字体的粗细，范围为 0 到 1000。 例如，400 为正常值，700 为粗体。 如果此值为零，则使用默认权重。  为方便起见，定义了以下值。  展开表   |  |  | | --- | --- | | **值** | **重量** | | FW\_DONTCARE | 0 | | FW\_THIN | 100 | | FW\_EXTRALIGHT | 200 | | FW\_ULTRALIGHT | 200 | | FW\_LIGHT | 300 | | FW\_NORMAL | 400 | | FW\_REGULAR | 400 | | FW\_MEDIUM | 500 | | FW\_SEMIBOLD | 600 | | FW\_DEMIBOLD | 600 | | FW\_BOLD | 700 | | FW\_EXTRABOLD | 800 | | FW\_ULTRABOLD | 800 | | FW\_HEAVY | 900 | | FW\_BLACK | 900 |   lfItalic  如果设置为 **TRUE**，则为斜体字体。  lfUnderline  如果设置为 **TRUE**，则为带下划线的字体。  lfStrikeOut  如果设置为 **TRUE**，则为删除线字体。  lfCharSet  字符集。 以下值是预定义的：   * ANSI\_CHARSET * BALTIC\_CHARSET * CHINESEBIG5\_CHARSET * DEFAULT\_CHARSET * EASTEUROPE\_CHARSET * GB2312\_CHARSET * GREEK\_CHARSET * HANGUL\_CHARSET * MAC\_CHARSET * OEM\_CHARSET * RUSSIAN\_CHARSET * SHIFTJIS\_CHARSET * SYMBOL\_CHARSET * TURKISH\_CHARSET * VIETNAMESE\_CHARSET   Windows 的朝鲜语版本：   * JOHAB\_CHARSET   Windows 中东语言版本：   * ARABIC\_CHARSET * HEBREW\_CHARSET   Windows 的泰语版本：   * THAI\_CHARSET   OEM\_CHARSET值指定依赖于操作系统的字符集。  DEFAULT\_CHARSET根据当前系统区域设置设置为值。 例如，当系统区域设置为英语 (美国) 时，它设置为 ANSI\_CHARSET。  操作系统中可能存在具有其他字符集的字体。 如果应用程序使用字符集未知的字体，则不应尝试翻译或解释使用该字体呈现的字符串。  此参数在字体映射过程中非常重要。 为确保创建字体时的结果一致，请勿指定OEM\_CHARSET或DEFAULT\_CHARSET。 如果在 **lfFaceName** 成员中指定字样名称，请确保 **lfCharSet** 值与 **lfFaceName** 中指定的字样字符集匹配。  lfOutPrecision  输出精度。 输出精度定义输出与所请求字体的高度、宽度、字符方向、转义、间距和字体类型的匹配程度。 可以是下列值之一。  展开表   |  |  | | --- | --- | | **值** | **含义** | | OUT\_CHARACTER\_PRECIS | 未使用。 | | OUT\_DEFAULT\_PRECIS | 指定默认字体映射器行为。 | | OUT\_DEVICE\_PRECIS | 当系统包含多个同名字体时，指示字体映射器选择设备字体。 | | OUT\_OUTLINE\_PRECIS | 此值指示字体映射器从 TrueType 和其他基于大纲的字体中进行选择。 | | OUT\_PS\_ONLY\_PRECIS | 指示字体映射器仅从 PostScript 字体中进行选择。 如果系统中没有安装 PostScript 字体，字体映射器将返回到默认行为。 | | OUT\_RASTER\_PRECIS | 当系统包含多个同名字体时，指示字体映射器选择光栅字体。 | | OUT\_STRING\_PRECIS | 字体映射器不使用此值，但在枚举光栅字体时会返回此值。 | | OUT\_STROKE\_PRECIS | 字体映射器不使用此值，但在枚举 TrueType、其他基于轮廓的字体和矢量字体时返回此值。 | | OUT\_TT\_ONLY\_PRECIS | 指示字体映射器仅从 TrueType 字体中进行选择。 如果系统中没有安装 TrueType 字体，字体映射器将返回到默认行为。 | | OUT\_TT\_PRECIS | 当系统包含多个同名字体时，指示字体映射器选择 TrueType 字体。 |     当操作系统包含多个具有指定名称的字体时，应用程序可以使用OUT\_DEVICE\_PRECIS、OUT\_RASTER\_PRECIS、OUT\_TT\_PRECIS和OUT\_PS\_ONLY\_PRECIS值来控制字体映射器选择字体的方式。 例如，如果操作系统在光栅和 TrueType 窗体中包含名为 Symbol 的字体，则指定OUT\_TT\_PRECIS将强制字体映射器选择 TrueType 版本。 指定OUT\_TT\_ONLY\_PRECIS会强制字体映射器选择 TrueType 字体，即使它必须替换另一个名称的 TrueType 字体。  lfClipPrecision  剪裁精度。 剪裁精度定义如何剪裁部分超出剪裁区域的字符。 它可以是以下一个或多个值。  有关坐标系的方向的详细信息，请参阅 *nOrientation* 参数的说明。  展开表   |  |  | | --- | --- | | **值** | **含义** | | CLIP\_CHARACTER\_PRECIS | 未使用。 | | CLIP\_DEFAULT\_PRECIS | 指定默认剪辑行为。 | | CLIP\_DFA\_DISABLE | **Windows XP SP1：** 关闭字体的字体关联。 请注意，此标志不保证在 Windows Server 2003 之后对任何平台产生任何影响。 | | CLIP\_EMBEDDED | 必须指定此标志才能使用嵌入的只读字体。 | | CLIP\_LH\_ANGLES | 使用此值时，所有字体的旋转取决于坐标系的方向是左手还是右手。如果未使用，设备字体始终逆时针旋转，但其他字体的旋转取决于坐标系的方向。 | | CLIP\_MASK | 未使用。 | | CLIP\_DFA\_OVERRIDE | 关闭字体的字体关联。 这与CLIP\_DFA\_DISABLE相同，但在某些情况下可能会有问题：建议使用的标志是CLIP\_DFA\_DISABLE。 | | CLIP\_STROKE\_PRECIS | 字体映射器不使用，但在枚举光栅、矢量或 TrueType 字体时返回。 为了兼容，枚举字体时始终返回此值。 | | CLIP\_TT\_ALWAYS | 未使用。 |   lfQuality  输出质量。 输出质量定义图形设备接口 (GDI) 必须尝试将逻辑字体属性与实际物理字体属性匹配的谨慎程度。 可以是下列值之一。  展开表   |  |  | | --- | --- | | **值** | **含义** | | ANTIALIASED\_QUALITY | 如果字体支持该字体，并且字体大小不是太小或太大，则字体始终为抗锯齿。 | | CLEARTYPE\_QUALITY | 如果设置，则尽可能使用 ClearType 抗锯齿方法) 呈现文本 (。 有关更多信息，请参见备注。 | | DEFAULT\_QUALITY | 字体的外观并不重要。 | | DRAFT\_QUALITY | 字体的外观不如使用PROOF\_QUALITY时重要。 对于 GDI 光栅字体，会启用缩放，这意味着可以使用更多字号，但质量可能较低。 如有必要，将合成粗体、斜体、下划线和删除线字体。 | | NONANTIALIASED\_QUALITY | 字体永远不会抗锯齿。 | | PROOF\_QUALITY | 字体的字符质量比逻辑字体属性的精确匹配更重要。 对于 GDI 光栅字体，将禁用缩放，并选择大小最接近的字体。 虽然在使用PROOF\_QUALITY时可能无法精确映射所选字号，但字体质量较高，外观不会失真。 如有必要，将合成粗体、斜体、下划线和删除线字体。 |     如果ANTIALIASED\_QUALITY和NONANTIALIASED\_QUALITY均未选中，则仅当用户在控制面板中选择平滑屏幕字体时，字体才会抗锯齿。  lfPitchAndFamily  字体的间距和系列。 两个低位指定字体的间距，可以是以下值之一。   * DEFAULT\_PITCH * FIXED\_PITCH * VARIABLE\_PITCH   成员的位 4 到 7 指定字体系列，可以是以下值之一。   * FF\_DECORATIVE * FF\_DONTCARE * FF\_MODERN * FF\_ROMAN * FF\_SCRIPT * FF\_SWISS   通过使用布尔 OR 运算符将一个音调常量与一个系列常量相联接，可以获得适当的值。  字体系列以一般方式描述字体的外观。 它们用于在所需的确切字样不可用时指定字体。 字体系列的值如下所示。  展开表   |  |  | | --- | --- | | **值** | **含义** | | FF\_DECORATIVE | 新奇字体。 例如 Old English。 | | FF\_DONTCARE | 使用默认字体。 | | FF\_MODERN | 具有固定笔划宽度的字体 (带衬线或无衬线) 的空白字体。 Monospace 字体通常是新式字体。 例如，Pica、Elite 和 CourierNew。 | | FF\_ROMAN | 笔划宽度可变的字体 (比例) 和衬线。 例如 MS Serif。 | | FF\_SCRIPT | 设计为类似于手写的字体。 例如 Script 和 Cursive。 | | FF\_SWISS | 笔划宽度可变的字体 (成比例) 且不带衬线。 例如 MS Sans Serif。 |   lfFaceName[LF\_FACESIZE]  以 NULL 结尾的字符串指定字体的字体名称。 此字符串的长度不能超过 32 **个 TCHAR** 值，包括终止 **NULL**。 [EnumFontFamiliesEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-enumfontfamiliesexa) 函数可用于枚举所有当前可用字体的字体名称。 如果 **lfFaceName** 是空字符串，则 GDI 使用与其他指定属性匹配的第一个字体。 注解 以下情况不支持 ClearType 抗锯齿：   * 文本在打印机上呈现。 * 显示设置为 256 种或更少颜色。 * 文本呈现到终端服务器客户端。 * 该字体不是 TrueType 字体或具有 TrueType 轮廓的 OpenType 字体。 例如，以下不支持 ClearType 抗锯齿：键入 1 字体、不带 TrueType 轮廓的 Postscript OpenType 字体、位图字体、矢量字体和设备字体。 * 该字体已针对包含嵌入位图的任何字号调整嵌入位图。 例如，这通常发生在东亚字体中。   **备注**  wingdi.h 标头将 LOGFONT 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** | wingdi.h (包括 Windows.h) |  请参阅 [CreateFont](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createfonta)  [CreateFontIndirect](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createfontindirecta)  [EnumFontFamiliesEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-enumfontfamiliesexa)  [字体和文本结构](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/font-and-text-structures)  [字体和文本概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/fonts-and-text) |

### LOGFONT结构体不需要初始化就可以使用

## ChooseFont函数用法

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ChooseFontA Choosefonta;  BOOL Choosefonta(  LPCHOOSEFONTA lpFont;  )  {...} 参数 lpFont 指向CHOOSEFONTA结构体的指针 要求 展开表   | **要求** | **值** | | --- | --- | | **Header** | commdlg.h |  另请参阅 [ChooseFont](https://docs.microsoft.com/previous-versions/windows/desktop/legacy/ms646914(v=vs.85)) |

## CeateFontIndirect函数的用法

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **CreateFontIndirect** 函数创建具有指定特征的逻辑字体。 随后可以选择该字体作为任何设备上下文的当前字体。 语法 HFONT CreateFontIndirectA(  [in] const LOGFONTA \*lplf  ); 参数 [in] lplf  指向定义逻辑字体特征的 [LOGFONT](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-logfonta) 结构的指针。 返回值 如果函数成功，则返回值是逻辑字体的句柄。  如果函数失败，则返回值为 NULL。 注解 **CreateFontIndirect** 函数使用 [LOGFONT](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-logfonta) 结构中指定的特征创建逻辑字体。 使用 [SelectObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-selectobject) 函数选择此字体时，GDI 的字体映射器会尝试将逻辑字体与现有物理字体匹配。 如果找不到完全匹配项，它将提供一个替代项，其特征与尽可能多的请求的特征匹配。  若要在不同语言版本的操作系统上获取适当的字体，请使用 [LOGFONT](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-logfonta) 结构中所需的字体特征调用 [EnumFontFamiliesEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-enumfontfamiliesexa)，检索适当的字样名称，并使用 [CreateFont](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createfonta) 或 **CreateFontIndirect** 创建字体。  如果不再需要该字体，请调用 [DeleteObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-deleteobject) 函数将其删除。  许多东亚语言的字体有两个字样名称：英文名称和本地化名称。 [CreateFont](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createfonta) 和 **CreateFontIndirect** 仅在与语言匹配的系统区域设置上采用本地化的字样名称，而在所有其他系统区域设置上采用英文字样名称。 最好的方法是尝试一个名称，如果失败，请尝试另一个名称。 请注意，如果系统区域设置与字体的语言不匹配，[EnumFontFamilies 和 EnumFontFamiliesEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-enumfontfamiliesexa) 将返回英文字样名称。  [CreateFont](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createfonta)、**CreateFontIndirect** 和 [CreateFontIndirectEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createfontindirectexa) 的字体映射器可识别英语和本地化的字样名称，而不考虑区域设置。 示例 有关示例，请参阅 [创建逻辑字体](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/creating-a-logical-font)。  **备注**  wingdi.h 标头将 CreateFontIndirect 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** | wingdi.h (包括 Windows.h) | | **Library** | Gdi32.lib | | **DLL** | Gdi32.dll |  另请参阅 [CreateFont](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createfonta)  [CreateFontIndirectEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createfontindirectexa)  [DeleteObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-deleteobject)  [EnumFontFamilies](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-enumfontfamiliesa)  [EnumFontFamiliesEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-enumfontfamiliesexa)  [EnumFonts](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-enumfontsa)  [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)  [LOGFONT](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-logfonta)  [SelectObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-selectobject) |

# 演练

## 1.新建一个win32项目，取名：Lesson40-choose-font

|  |
| --- |
|  |

## 2.首先我们把窗口的样式修改一下，添加一个CS\_DBLCLKS样式

|  |
| --- |
|  |

## 3.我们想双击窗口客户区，就弹出一个字体选择对话框，然后我们可以选择一个字体。我们需要在窗口过程函数里面添加对鼠标双击的消息处理代码，先用消息框测试一下

|  |
| --- |
|  |

### 效果

|  |
| --- |
|  |

### 说明是能够响应鼠标左键双击事件的，然后我们就可以把消息框注释了。

## 3.下面我们来实现双击弹出选择字体对话框的功能，我们把代码封装到一个叫做PickFont的函数中，它返回一个HFONT对象，注意：默认stdafx.h里面有一句话：#define WIN32\_LEAN\_AND\_MEAN // 从 Windows 头文件中排除极少使用的信息我们需要把他注释了，否则那些字体对话框api和CHOOSEFONT结构体就好变为没有定义的类型

|  |
| --- |
|  |

## 4.PickFont函数的代码如下

|  |
| --- |
|  |

## 5.我们先在双击事件里面调用一下这个函数，看看能否弹出字体对话框

|  |
| --- |
|  |

### 效果：是能够弹出字体选择对话框的

|  |
| --- |
|  |

## 6.然后我们需要定义一个变量来接受这个字体，同时需要一个变量接受原来的字体

|  |
| --- |
|  |
|  |

### 效果

|  |
| --- |
|  |
|  |

## 7.我们可以使用随机数代替TextOut中的固定的x和y，这样子，每一次都会在不同的位置显示文本，需要设置一些本量

|  |
| --- |
|  |

## 8.然后还需要处理WM\_SIZE消息，在这里回去客户区窗口大小

|  |
| --- |
|  |

## 9.然后我们就可以把TextOut的位置替换为x和y

|  |
| --- |
|  |

### 效果

|  |
| --- |
|  |

# 这个项目的代码如下

## Lesson40-choose-font.cpp

|  |
| --- |
| // Lesson40-choose-font.cpp : 定义应用程序的入口点。  //  #include "stdafx.h"  #include "Lesson40-choose-font.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\_LESSON40CHOOSEFONT, 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\_LESSON40CHOOSEFONT));  // 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|CS\_DBLCLKS;//添加接受双击标记  wcex.lpfnWndProc = WndProc;  wcex.cbClsExtra = 0;  wcex.cbWndExtra = 0;  wcex.hInstance = hInstance;  wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_LESSON40CHOOSEFONT));  wcex.hCursor = LoadCursor(NULL, IDC\_ARROW);  wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW+1);  wcex.lpszMenuName = MAKEINTRESOURCE(IDC\_LESSON40CHOOSEFONT);  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;  }  HFONT PickFont()  {  HFONT hfont;  LOGFONT lf;//这个结构体不需要初始化  CHOOSEFONT cf;  //使用这个结构体的变量之前，需要把他初始化,这个初始化也是有点技巧的，可以先把他所有的成员设置为0  memset(&cf,0,sizeof(cf));  //然后我们再来设置不是为0的成员,一般只有4个  cf.lStructSize = sizeof(CHOOSEFONT);  cf.lpLogFont = &lf;  cf.Flags = CF\_SCREENFONTS;  cf.nFontType = SCREEN\_FONTTYPE;  ChooseFont(&cf);  hfont = CreateFontIndirect(cf.lpLogFont);  return hfont;  }  //  // 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;  HFONT font,oldFont;  int x,y;//文本的显示位置  static int cxClient,cyClient;//窗口客户区大小  switch (message)  {  case WM\_SIZE:  cxClient = LOWORD(lParam);  cyClient = HIWORD(lParam);  break;  //处理鼠标双击  case WM\_LBUTTONDBLCLK:  //测试  //MessageBox(hWnd,\_T("mouse double clicked"),\_T(""),0);  //弹出选择字体对话框  HDC hdc;  hdc = GetDC(hWnd);  font = PickFont();  oldFont = (HFONT)SelectObject(hdc,font);  //设置随机数种子  srand(GetTickCount());  //产生随机数  x = rand()%cxClient;  y = rand()%cyClient;  //在随机的位置输出文本  TextOut(hdc,x,y,\_T("Hello,Font"),10);  SelectObject(hdc,oldFont);  DeleteObject(font);  ReleaseDC(hWnd,hdc);  break;  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;  } |

# 扩展，其实这个例子写的不太好，因为当窗口发生变化的时候，文本会消失。

## 我们需要把双击事件的代码移到WM\_PAINT消息处理代码中，想想看有什么解决办法