# 学习大纲

|  |
| --- |
|  |

## EnumFonts函数的用法，此函数仅用于与 16 位版本的 Windows 兼容

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 语法 int EnumFontsA(  [in] HDC hdc,  [in] LPCSTR lpLogfont,  [in] FONTENUMPROCA lpProc,  [in] LPARAM lParam  ); 参数 [in] hdc  从中枚举字体的设备上下文的句柄。  [in] lpLogfont  指向以 null 结尾的字符串的指针，该字符串指定所需字体的字体名称。 如果 *lpFaceName* 为 **NULL**， **则 EnumFonts** 会随机选择并枚举每个可用字体的一个字体。  [in] lpProc  指向应用程序 definedcallback 函数的指针。 有关详细信息，请参阅 [EnumFontsProc](https://learn.microsoft.com/zh-cn/previous-versions/dd162623(v=vs.85))。  [in] lParam  指向任何应用程序定义数据的指针。 数据连同字体信息一起传递到回调函数。 返回值 返回值是回调函数返回的最后一个值。 其含义由应用程序定义。 注解 使用 [EnumFontFamiliesEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-enumfontfamiliesexa) 而不是 **EnumFonts**。 **EnumFontFamiliesEx** 函数不同于 **EnumFonts** 函数，因为它检索与 TrueType 字体关联的样式名称。 使用 **EnumFontFamiliesEx**，可以检索有关无法使用 **EnumFonts** 函数枚举的字体样式的信息。  许多东亚语言的字体有两个字体名称：英文名称和本地化名称。 如果系统区域设置与字体语言不匹配，则 [EnumFontFamilies 和 EnumFontFamiliesEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-enumfontfamiliesexa) 将返回英文字体名称。  **备注**  wingdi.h 标头将 EnumFonts 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) | | **Library** | Gdi32.lib | | **DLL** | Gdi32.dll |  另请参阅 [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)  [EnumFontsProc](https://learn.microsoft.com/zh-cn/previous-versions/dd162623(v=vs.85))  [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)  [GetDeviceCaps](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getdevicecaps) |

## EnumFontFamilies函数的用法，提供此函数只是为了与 16 位版本的 Windows 兼容

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **EnumFontFamilies** 函数枚举指定字体系列中在指定设备上可用的字体。  **注意** 提供此函数只是为了与 16 位版本的 Windows 兼容。 应用程序应使用 [**EnumFontFamiliesEx**](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-enumfontfamiliesexa) 函数。   语法 C++复制  int EnumFontFamiliesA(  [in] HDC hdc,  [in] LPCSTR lpLogfont,  [in] FONTENUMPROCA lpProc,  [in] LPARAM lParam  ); 参数 [in] hdc  从中枚举字体的设备上下文的句柄。  [in] lpLogfont  指向以 null 结尾的字符串的指针，该字符串指定所需字体的系列名称。 如果 *lpszFamily* 为 **NULL**， **则 EnumFontFamilies** 选择并枚举每个可用类型系列的一个字体。  [in] lpProc  指向应用程序定义的回调函数的指针。 有关信息，请参阅 [EnumFontFamProc](https://learn.microsoft.com/zh-cn/previous-versions/dd162621(v=vs.85))。  [in] lParam  指向应用程序提供的数据的指针。 数据与字体信息一起传递给回调函数。 返回值 返回值是回调函数返回的最后一个值。 其含义特定于实现。 注解 对于具有 *lpszFamily* 参数指定的字样名称的每个字体， **EnumFontFamilies** 函数将检索有关该字体的信息，并将其传递给 *lpEnumFontFamProc* 参数指向的函数。 应用程序定义的回调函数可以根据需要处理字体信息。 枚举一直持续到没有更多的字体或回调函数返回零。  当使用 SetGraphicsMode 函数将设备上下文中的图形模式设置为 GM\_ADVANCED 并将 DEVICE\_FONTTYPE 标志传递给 FontType 参数时，此函数将在系统上返回类型 1 和 OpenType 字体的列表。 当图形模式未设置为 GM\_ADVANCED 时，此函数将在系统上返回类型 1、OpenType 和 TrueType 字体的列表。  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。 如果系统区域设置与字体的语言不匹配，则 [EnumFontFamilies 和 EnumFontFamiliesEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-enumfontfamiliesexa) 返回英文字样名称。 示例 有关示例，请参阅 [枚举已安装字体](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/enumerating-the-installed-fonts)。  **备注**  wingdi.h 标头将 EnumFontFamilies 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 [EnumFontFamProc](https://learn.microsoft.com/zh-cn/previous-versions/dd162621(v=vs.85))  [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) |

## EnumFontFamiliesEx函数的用法，这是微软推荐的函数

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **EnumFontFamiliesEx** 函数枚举系统中与 [LOGFONT](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-logfonta) 结构指定的字体特征匹配的所有唯一命名字体。 **EnumFontFamiliesEx** 根据字样名称和/或字符集枚举字体。 语法 C++复制  int EnumFontFamiliesExA(  [in] HDC hdc,  [in] LPLOGFONTA lpLogfont,  [in] FONTENUMPROCA lpProc,  [in] LPARAM lParam,  DWORD dwFlags  ); 参数 [in] hdc  从中枚举字体的设备上下文的句柄。  [in] lpLogfont  指向 [LOGFONT](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-logfonta) 结构的指针，该结构包含有关要枚举的字体的信息。 函数检查以下成员。  展开表   |  |  | | --- | --- | | **成员** | **说明** | | **lfCharSet** | 如果设置为 DEFAULT\_CHARSET， 函数将枚举所有字符集中所有唯一命名的字体。 (如果存在两个同名字体，则只枚举一种。) 如果设置为有效的字符集值，则 函数仅枚举指定字符集中的字体。 | | **lfFaceName** | 如果设置为空字符串， 函数将枚举每个可用字样名称中的一个字体。 如果设置为有效的字样名称，则 函数枚举具有指定名称的所有字体。 | | **lfPitchAndFamily** | 对于操作系统的所有语言版本，必须设置为零。 |   [in] lpProc  指向应用程序定义的回调函数的指针。 有关详细信息，请参阅 [EnumFontFamExProc](https://learn.microsoft.com/zh-cn/previous-versions/dd162618(v=vs.85)) 函数。  [in] lParam  应用程序定义的值。 函数将此值与字体信息一起传递给回调函数。  dwFlags  未使用此参数，它必须为零。 返回值 返回值是回调函数返回的最后一个值。 此值取决于哪些字体系列可用于指定设备。 注解 **EnumFontFamiliesEx** 函数不使用带标记的字样名称来标识字符集。 相反，它始终将正确的字样名称和单独的字符集值传递给回调函数。 函数基于 [LOGFONT](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-logfonta) 结构中 **lfCharSet** 和 **lfFaceName** 成员的值枚举字体。  与 [EnumFontFamilies](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-enumfontfamiliesa) 一样， **EnumFontFamiliesEx** 枚举所有字体样式。 并非所有字体样式都涵盖相同的字符集。 例如，Fontorama Bold 可能包含 ANSI、希腊语和西里尔文字符，但 Fontorama Italic 可能仅包含 ANSI 字符。 因此，最好不要假定指定的字体覆盖特定的字符集，即使它是 ANSI 字符集。 下表显示了 **lfCharSet** 和 **lfFaceName** 值的各种组合的结果。  展开表   |  |  | | --- | --- | | **值** | **含义** | | **lfCharSet** = DEFAULT\_CHARSET  **lfFaceName** = '\0' | 枚举所有字符集中所有唯一命名的字体。 如果存在两个同名字体，则仅枚举一种。 | | **lfCharSet** = DEFAULT\_CHARSET  **lfFaceName** = 特定字体 | 枚举特定字体中的所有字符集和样式。 | | **lfCharSet** =特定字符集  **lfFaceName** = '\0' | 枚举特定字符集中所有字体的所有样式。 | | **lfCharSet** =特定字符集  **lfFaceName** = 特定字体 | 枚举特定字符集中字体的所有样式。 |     下面的代码示例演示如何使用这些值。  C++  // To enumerate all styles and charsets of all fonts:  lf.lfFaceName[0] = '\0';  lf.lfCharSet = DEFAULT\_CHARSET;  HRESULT hr;  // To enumerate all styles and character sets of the Arial font:  hr = StringCchCopy( (LPSTR)lf.lfFaceName, LF\_FACESIZE, "Arial" );  if (FAILED(hr))  {  // TODO: write error handler  }  lf.lfCharSet = DEFAULT\_CHARSET;  C++  // To enumerate all styles of all fonts for the ANSI character set  lf.lfFaceName[0] = '\0';  lf.lfCharSet = ANSI\_CHARSET;  // To enumerate all styles of Arial font that cover the ANSI charset  hr = StringCchCopy( (LPSTR)lf.lfFaceName, LF\_FACESIZE, "Arial" );  if (FAILED(hr))  {  // TODO: write error handler  }  lf.lfCharSet = ANSI\_CHARSET;  [EnumFontFamilies](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-enumfontfamiliesa) 和 **EnumFontFamiliesEx** 的回调函数非常相似。 main区别在于 [ENUMLOGFONTEX](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-enumlogfontexa) 结构包含脚本字段。  请注意，根据 **lfCharSet** 和 **lfFaceName** 的值， **EnumFontFamiliesEx** 将枚举相同字体的次数与字体中存在不同字符集的次数一样多。 这可以创建一个广泛的字体列表，这些字体可能会对用户造成负担。 例如，可以针对波罗的海、西文、希腊文、土耳其语和西里尔文字符集显示世纪课本字体。 为避免这种情况，应用程序应筛选字体列表。  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。 如果系统区域设置与字体的语言不匹配，则 **EnumFontFamilies 和 EnumFontFamiliesEx** 返回英文字样名称。  当使用 SetGraphicsMode 函数将设备上下文中的图形模式设置为 GM\_ADVANCED 并将 DEVICE\_FONTTYPE 标志传递给 FontType 参数时，此函数将在系统上返回类型 1 和 OpenType 字体的列表。 当图形模式未设置为 GM\_ADVANCED 时，此函数将在系统上返回类型 1、OpenType 和 TrueType 字体的列表。  **备注**  wingdi.h 标头将 EnumFontFamiliesEx 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 [EnumFontFamExProc](https://learn.microsoft.com/zh-cn/previous-versions/dd162618(v=vs.85))  [EnumFontFamilies](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-enumfontfamiliesa)  [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) |

## NEWTEXTMETRICA 结构

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 语法 typedef struct tagNEWTEXTMETRICA {  LONG tmHeight;  LONG tmAscent;  LONG tmDescent;  LONG tmInternalLeading;  LONG tmExternalLeading;  LONG tmAveCharWidth;  LONG tmMaxCharWidth;  LONG tmWeight;  LONG tmOverhang;  LONG tmDigitizedAspectX;  LONG tmDigitizedAspectY;  BYTE tmFirstChar;  BYTE tmLastChar;  BYTE tmDefaultChar;  BYTE tmBreakChar;  BYTE tmItalic;  BYTE tmUnderlined;  BYTE tmStruckOut;  BYTE tmPitchAndFamily;  BYTE tmCharSet;  DWORD ntmFlags;  UINT ntmSizeEM;  UINT ntmCellHeight;  UINT ntmAvgWidth;  } NEWTEXTMETRICA, \*PNEWTEXTMETRICA, \*NPNEWTEXTMETRICA, \*LPNEWTEXTMETRICA; 成员 tmHeight  字符的高度 (上升 + 下降) 。  tmAscent  升 (基线上方的单位) 字符。  tmDescent  字符) 基线以下的下降 (单位。  tmInternalLeading  **tmHeight** 成员设置的边界内) 前导 (空间量。 此区域中可能会出现重音符号和其他音调符号。 设计器可将此成员设置为零。  tmExternalLeading  应用程序在行之间添加的额外前导 (空间量) 。 由于此区域在字体外部，因此它不包含任何标记，并且不会在 OPAQUE 或 TRANSPARENT 模式下被文本输出调用更改。 设计器可将此成员设置为零。  tmAveCharWidth  字体中字符的平均宽度 (通常定义为字母 x) 的宽度。 此值不包括粗体或斜体字符所需的悬垂。  tmMaxCharWidth  该字体中最宽字符的宽度。  tmWeight  该字体的粗细。  tmOverhang  可能添加到一些合成字体中的每个字符串的附加宽度。 合成某些属性（如粗体或斜体）时，图形设备接口 (GDI) 或设备可能需要同时按字符和按字符串添加宽度。 例如，GDI 通过展开每个字符的间距并用偏移值进行过度调整，使字符串加粗;它通过剪切字符串来斜体字体。 在任一情况下，基本字符串都存在悬垂。 对于粗体字符串，悬空是过度打击偏移的距离。 对于斜体字符串，悬垂是字体顶部从字体底部剪切的量。  **tmOverhang** 成员使应用程序能够确定 [GetTextExtentPoint32](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-gettextextentpoint32a) 函数调用对单个字符返回的字符宽度是多少是实际字符宽度，以及每字符串多余宽度多少。 实际宽度是范围减去悬垂。  tmDigitizedAspectX  该字体的目标设备的水平比例。  tmDigitizedAspectY  该字体的目标设备的垂直比例。 **tmDigitizedAspectX** 和 **tmDigitizedAspectY** 成员的比率是为其设计字体的设备纵横比。  tmFirstChar  字体中定义的第一个字符的值。  tmLastChar  字体中定义的最后一个字符的值。  tmDefaultChar  要替换不在字体中的字符的字符的值。  tmBreakChar  用于定义文本对齐的分词符的字符的值。  tmItalic  如果为非零，则为斜体字体。  tmUnderlined  带下划线的字体（如果为非零）。  tmStruckOut  如果为非零，则为删除线字体。  tmPitchAndFamily  所选字体的间距和系列。 低位 (位 0) 指定字体的间距。 如果为 1，则字体为可变间距 (或成比例) 。 如果为 0，则字体为固定间距 (或单空格) 。 位 1 和 2 指定字体类型。 如果两位均为 0，则字体为光栅字体;如果位 1 为 1，位 2 为 0，则字体为向量字体;如果位 1 为 0 且设置位 2，或者如果两位均为 1，则字体为其他类型。 如果字体是设备字体，则位 3 为 1;否则为 0。  四个高序位指定字体系列。 可以使用按位 AND 运算符将 **tmPitchAndFamily** 成员与十六进制值0xF0组合，然后可以与字体系列名称进行比较，以获取相同的匹配项。 有关字体系列的详细信息，请参阅 [LOGFONT](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-logfonta)。  tmCharSet  字体的字符集。  ntmFlags  指定字体是否为斜体、下划线、轮廓、加粗等。 可以是以下值的任意合理组合。  展开表   |  |  |  | | --- | --- | --- | | **bit** | **名称** | **含义** | | 0 | NTM\_ITALIC | 斜体 | | 5 | NTM\_BOLD | 粗体 | | 8 | NTM\_REGULAR | 普通 | | 16 | NTM\_NONNEGATIVE\_AC | 任何字形中任何大小的字体都没有负 A 或 C 空间。 | | 17 | NTM\_PS\_OPENTYPE | PostScript OpenType 字体 | | 18 | NTM\_TT\_OPENTYPE | TrueType OpenType 字体 | | 19 | NTM\_MULTIPLEMASTER | 多母版字体 | | 20 | NTM\_TYPE1 | 键入 1 字体 | | 21 | NTM\_DSIG | 具有数字签名的字体。 这允许可追溯性，并确保字体已经过测试且未损坏 |   ntmSizeEM  字体的 em 正方形的大小。 此值以名义单位表示， (即字体设计) 单位。  ntmCellHeight  字体的高度（以名义单位表示）。 此值应与 **ntmSizeEM** 成员的值进行比较。  ntmAvgWidth  字体中字符的平均宽度（以名义单位表示）。 此值应与 **ntmSizeEM** 成员的值进行比较。 备注 **NEWTEXTMETRIC** 结构的最后四个成员不包括在 [TEXTMETRIC](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-textmetrica) 结构中;在所有其他方面，结构都是相同的。  **NEWTEXTMETRIC** 结构中的大小通常以逻辑单元指定;也就是说，它们依赖于显示上下文的当前映射模式。  **备注**  wingdi.h 标头将 NEWTEXTMETRIC 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) |  另请参阅 [EnumFontFamilies](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-enumfontfamiliesa)  [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)  [GetTextExtentPoint32](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-gettextextentpoint32a)  [GetTextMetrics](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-gettextmetrics)  [LOGFONT](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-logfonta) |

# 枚举已安装字体

## EnumFamCallBack回调函数用法

|  |
| --- |
| 在某些情况下，应用程序必须能够枚举可用字体并选择最适合特定操作的字体。 应用程序可以通过调用 [EnumFonts](https://learn.microsoft.com/zh-cn/windows/desktop/api/Wingdi/nf-wingdi-enumfontsa) 或 [EnumFontFamilies](https://learn.microsoft.com/zh-cn/windows/desktop/api/Wingdi/nf-wingdi-enumfontfamiliesa) 函数来枚举可用字体。 这些函数将有关可用字体的信息发送到应用程序提供的回调函数。 回调函数接收 [LOGFONT](https://learn.microsoft.com/zh-cn/windows/win32/api/wingdi/ns-wingdi-logfonta) 和 [NEWTEXTMETRIC](https://learn.microsoft.com/zh-cn/windows/win32/api/wingdi/ns-wingdi-newtextmetrica) 结构中的信息。 ([**NEWTEXTMETRIC**](https://learn.microsoft.com/zh-cn/windows/win32/api/wingdi/ns-wingdi-newtextmetrica) 结构包含有关 TrueType 字体的信息。当回调函数接收有关非 TrueType 字体的信息时，该信息包含在 [TEXTMETRIC](https://learn.microsoft.com/zh-cn/windows/win32/api/wingdi/ns-wingdi-textmetrica) 结构中。) 通过使用此信息，应用程序可以将用户的选择限制为仅那些可用的字体。  [**EnumFontFamilies**](https://learn.microsoft.com/zh-cn/windows/win32/api/wingdi/nf-wingdi-enumfontfamiliesa) 函数类似于 [**EnumFonts**](https://learn.microsoft.com/zh-cn/windows/win32/api/wingdi/nf-wingdi-enumfontsa) 函数，但包含一些额外的功能。 **EnumFontFamilies** 允许应用程序利用 TrueType 字体可用的样式。 新的和升级的应用程序应使用 **EnumFontFamilies** 而不是 **EnumFonts**。  TrueType 字体围绕字体名称组织， (例如 Courier New) 和样式名称 (斜体、粗体和超粗) 。 [**EnumFontFamilies**](https://learn.microsoft.com/zh-cn/windows/win32/api/wingdi/nf-wingdi-enumfontfamiliesa) 函数枚举与指定系列名称关联的所有样式，而不仅仅是粗体和斜体属性。 例如，当系统包含名为 Courier New Extra-Bold 的 TrueType 字体时， **EnumFontFamilies** 会将其与其他 Courier New 字体一起列出。 **EnumFontFamilies** 的功能对于具有多种或不同寻常样式的字体以及跨越国际边界的字体非常有用。  如果应用程序不提供字样名称， [则 EnumFonts](https://learn.microsoft.com/zh-cn/windows/desktop/api/Wingdi/nf-wingdi-enumfontsa) 和 [EnumFontFamilies](https://learn.microsoft.com/zh-cn/windows/desktop/api/Wingdi/nf-wingdi-enumfontfamiliesa) 函数会提供有关每个可用系列中一种字体的信息。 若要枚举设备上下文中的所有字体，应用程序可以为字样名称指定 **NULL** ，编译可用字样的列表，然后枚举每个字样中的每个字体。  以下示例使用 [**EnumFontFamilies**](https://learn.microsoft.com/zh-cn/windows/win32/api/wingdi/nf-wingdi-enumfontfamiliesa) 函数检索可用光栅、矢量和 TrueType 字体系列的数量。  C++：  UINT uAlignPrev;  int aFontCount[] = { 0, 0, 0 };  char szCount[8];  HRESULT hr;  size\_t \* pcch;    EnumFontFamilies(hdc, (LPCTSTR) NULL,  (FONTENUMPROC) EnumFamCallBack, (LPARAM) aFontCount);    uAlignPrev = SetTextAlign(hdc, TA\_UPDATECP);    MoveToEx(hdc, 10, 50, (LPPOINT)NULL);  TextOut(hdc, 0, 0, "Number of raster fonts: ", 24);  itoa(aFontCount[0], szCount, 10);    hr = StringCchLength(szCount, 9, pcch);  if (FAILED(hr))  {  // TODO: write error handler  }  TextOut(hdc, 0, 0, szCount, \*pcch);    MoveToEx(hdc, 10, 75, (LPPOINT)NULL);  TextOut(hdc, 0, 0, "Number of vector fonts: ", 24);  itoa(aFontCount[1], szCount, 10);  hr = StringCchLength(szCount, 9, pcch);  if (FAILED(hr))  {  // TODO: write error handler  }  TextOut(hdc, 0, 0, szCount, \*pcch);    MoveToEx(hdc, 10, 100, (LPPOINT)NULL);  TextOut(hdc, 0, 0, "Number of TrueType fonts: ", 26);  itoa(aFontCount[2], szCount, 10);  hr = StringCchLength(szCount, 9, pcch);  if (FAILED(hr))  {  // TODO: write error handler  }  TextOut(hdc, 0, 0, szCount, \*pcch);    SetTextAlign(hdc, uAlignPrev);    BOOL CALLBACK EnumFamCallBack(LPLOGFONT lplf, LPNEWTEXTMETRIC lpntm, DWORD FontType, LPVOID aFontCount)  {  int far \* aiFontCount = (int far \*) aFontCount;    // Record the number of raster, TrueType, and vector  // fonts in the font-count array.    if (FontType & RASTER\_FONTTYPE)  aiFontCount[0]++;  else if (FontType & TRUETYPE\_FONTTYPE)  aiFontCount[2]++;  else  aiFontCount[1]++;    if (aiFontCount[0] || aiFontCount[1] || aiFontCount[2])  return TRUE;  else  return FALSE;    UNREFERENCED\_PARAMETER( lplf );  UNREFERENCED\_PARAMETER( lpntm );  }  此示例使用两个掩码（RASTER\_FONTTYPE和TRUETYPE\_FONTTYPE）来确定要枚举的字体类型。 如果设置了RASTER\_FONTTYPE位，则字体为光栅字体。 如果设置了TRUETYPE\_FONTTYPE位，则字体为 TrueType 字体。 如果未设置这两个位，则字体为矢量字体。 第三个掩码，DEVICE\_FONTTYPE，是在设备 (时设置的，例如，激光打印机) 支持下载 TrueType 字体;如果设备是显示适配器、点阵打印机或其他光栅设备，则为零。 应用程序还可以使用DEVICE\_FONTTYPE掩码来区分 GDI 提供的光栅字体和设备提供的字体。 系统可以模拟 GDI 提供的光栅字体的粗体、斜体、下划线和删除线属性，但不适用于设备提供的字体。  应用程序还可以在 [NEWTEXTMETRIC](https://learn.microsoft.com/zh-cn/windows/win32/api/wingdi/ns-wingdi-newtextmetrica) 结构的 **tmPitchAndFamily** 成员中检查位 1 和 2，以标识 TrueType 字体。 如果位 1 为 0，位 2 为 1，则字体为 TrueType 字体。  矢量字体分类为OEM\_CHARSET而不是ANSI\_CHARSET。 某些应用程序通过使用此信息来标识矢量字体，并检查 [**NEWTEXTMETRIC**](https://learn.microsoft.com/zh-cn/windows/win32/api/wingdi/ns-wingdi-newtextmetrica) 结构的 **tmCharSet** 成员。 这种分类通常会阻止字体映射器选择矢量字体，除非有具体要求。 (大多数应用程序不再使用矢量字体，因为它们的笔划是单行，绘制时间比 TrueType 字体长，后者提供了许多需要矢量字体的相同缩放和旋转功能。) |

## windows使用的字体一般有3种类型：

### 1）光栅型：Raster Type

### 2）向量型: Vector Type

### 3) True Type 类型

# 演练

## 1.新建一个c++常规空项目，取名：Lesson42-enum-intalled-fonts，然后新建一个cpp文件，取名enum-intalled-fonts.cpp

|  |
| --- |
|  |

## 2.我们自定义一个函数，来显示所有系统安装了的字体，这个函数里面调用了EnumFontFamillies函数

|  |
| --- |
|  |

## 3. EnumFontFamillies函数需要一个回调函数作为参数，我们来编写这个回调函数

|  |
| --- |
|  |

### 你也可以在找到一种字体后，判断一下它是不是斜体字，代码如下

|  |
| --- |
|  |

## 4.然后在面函数里面调用这个函数

|  |
| --- |
|  |

# 这一节的学习到此为止，完整代码如下

## enum-intalled-fonts.cpp

|  |
| --- |
| #include<Windows.h>  #include<cstdio>  BOOL CALLBACK FontEnumProc(LPLOGFONT lplf,LPNEWTEXTMETRIC lpntm,DWORD fontType,LPVOID aFontCount)  {    UNREFERENCED\_PARAMETER( lpntm );  int far\* fCount = (int far\*)aFontCount; //类型转换  if(fontType & RASTER\_FONTTYPE)  {  fCount[0]++;  }  else if(fontType & TRUETYPE\_FONTTYPE)  {  fCount[2]++;  }  else  {  fCount[1]++;  }  //printf("%s\t italic = %d\n",lplf->lfFaceName,lplf->lfItalic);//可以每一种字体输出是否是斜体  if(fCount[0]||fCount[1]||fCount[2])  {  return TRUE;  }    return FALSE;  }  //自定义枚举系统安装字体的函数  DWORD ListInstalledFonts(HWND hwnd)  {  HDC hdc;  hdc = GetDC(hwnd);  UINT aAlignPrev;//原来的对齐方式  int aFontCount[] = {0,0,0}; //一个数组，用来计算每一种类型的字体有多少个  /\*char szCount[8];  HRESULT hr;  size\_t \* pcch;\*/  EnumFontFamilies(hdc,(LPCTSTR)NULL,(FONTENUMPROC)FontEnumProc,(LPARAM)aFontCount);  printf("Raster Fonts Count:%d\n",aFontCount[0]);  printf("Vector Fonts Count:%d\n",aFontCount[1]);  printf("True Type Fonts Count:%d\n",aFontCount[2]);    ReleaseDC(hwnd,hdc);  return 0;  }  int main()  {  ListInstalledFonts(NULL);//因使用这里传递为命令行程序没有窗口，  system("pause");  return 0;  } |

### 效果

|  |
| --- |
|  |

# 扩展,可以使用MFC对话框程序来枚举系统字体

## 1.新建一个MFC对话框应用程序,取名:EnumFontsDemo,删除对话框的默认控件,把对话框布局如下:

|  |
| --- |
|  |

## 2.控件的ID如下

|  |
| --- |
|  |

## 3.EnumFontsDemoDlg.cpp的源码如下

|  |
| --- |
| // EnumFontsDemoDlg.cpp: 实现文件  //  #include "pch.h"  #include "framework.h"  #include "EnumFontsDemo.h"  #include "EnumFontsDemoDlg.h"  #include "afxdialogex.h"  #ifdef \_DEBUG  #define new DEBUG\_NEW  #endif  CString g\_fontStr;  // 用于应用程序“关于”菜单项的 CAboutDlg 对话框  class CAboutDlg : public CDialogEx  {  public:  CAboutDlg();  // 对话框数据  #ifdef AFX\_DESIGN\_TIME  enum { IDD = IDD\_ABOUTBOX };  #endif  protected:  virtual void DoDataExchange(CDataExchange\* pDX); // DDX/DDV 支持  // 实现  protected:  DECLARE\_MESSAGE\_MAP()  };  CAboutDlg::CAboutDlg() : CDialogEx(IDD\_ABOUTBOX)  {  }  void CAboutDlg::DoDataExchange(CDataExchange\* pDX)  {  CDialogEx::DoDataExchange(pDX);  }  BEGIN\_MESSAGE\_MAP(CAboutDlg, CDialogEx)  END\_MESSAGE\_MAP()  // CEnumFontsDemoDlg 对话框  CEnumFontsDemoDlg::CEnumFontsDemoDlg(CWnd\* pParent /\*=nullptr\*/)  : CDialogEx(IDD\_ENUMFONTSDEMO\_DIALOG, pParent)  {  m\_hIcon = AfxGetApp()->LoadIcon(IDR\_MAINFRAME);  }  void CEnumFontsDemoDlg::DoDataExchange(CDataExchange\* pDX)  {  CDialogEx::DoDataExchange(pDX);  }  BEGIN\_MESSAGE\_MAP(CEnumFontsDemoDlg, CDialogEx)  ON\_WM\_SYSCOMMAND()  ON\_WM\_PAINT()  ON\_WM\_QUERYDRAGICON()  ON\_BN\_CLICKED(IDC\_BN\_ENUM, &CEnumFontsDemoDlg::OnBnClickedBnEnum)  END\_MESSAGE\_MAP()  // CEnumFontsDemoDlg 消息处理程序  BOOL CEnumFontsDemoDlg::OnInitDialog()  {  CDialogEx::OnInitDialog();  // 将“关于...”菜单项添加到系统菜单中。  // IDM\_ABOUTBOX 必须在系统命令范围内。  ASSERT((IDM\_ABOUTBOX & 0xFFF0) == IDM\_ABOUTBOX);  ASSERT(IDM\_ABOUTBOX < 0xF000);  CMenu\* pSysMenu = GetSystemMenu(FALSE);  if (pSysMenu != nullptr)  {  BOOL bNameValid;  CString strAboutMenu;  bNameValid = strAboutMenu.LoadString(IDS\_ABOUTBOX);  ASSERT(bNameValid);  if (!strAboutMenu.IsEmpty())  {  pSysMenu->AppendMenu(MF\_SEPARATOR);  pSysMenu->AppendMenu(MF\_STRING, IDM\_ABOUTBOX, strAboutMenu);  }  }  // 设置此对话框的图标。 当应用程序主窗口不是对话框时，框架将自动  // 执行此操作  SetIcon(m\_hIcon, TRUE); // 设置大图标  SetIcon(m\_hIcon, FALSE); // 设置小图标  // TODO: 在此添加额外的初始化代码  return TRUE; // 除非将焦点设置到控件，否则返回 TRUE  }  void CEnumFontsDemoDlg::OnSysCommand(UINT nID, LPARAM lParam)  {  if ((nID & 0xFFF0) == IDM\_ABOUTBOX)  {  CAboutDlg dlgAbout;  dlgAbout.DoModal();  }  else  {  CDialogEx::OnSysCommand(nID, lParam);  }  }  // 如果向对话框添加最小化按钮，则需要下面的代码  // 来绘制该图标。 对于使用文档/视图模型的 MFC 应用程序，  // 这将由框架自动完成。  void CEnumFontsDemoDlg::OnPaint()  {  if (IsIconic())  {  CPaintDC dc(this); // 用于绘制的设备上下文  SendMessage(WM\_ICONERASEBKGND, reinterpret\_cast<WPARAM>(dc.GetSafeHdc()), 0);  // 使图标在工作区矩形中居中  int cxIcon = GetSystemMetrics(SM\_CXICON);  int cyIcon = GetSystemMetrics(SM\_CYICON);  CRect rect;  GetClientRect(&rect);  int x = (rect.Width() - cxIcon + 1) / 2;  int y = (rect.Height() - cyIcon + 1) / 2;  // 绘制图标  dc.DrawIcon(x, y, m\_hIcon);  }  else  {  CDialogEx::OnPaint();  }  }  //当用户拖动最小化窗口时系统调用此函数取得光标  //显示。  HCURSOR CEnumFontsDemoDlg::OnQueryDragIcon()  {  return static\_cast<HCURSOR>(m\_hIcon);  }  //枚举字体的回调函数,不是成员函数.  BOOL CALLBACK EnumFontProc(LPLOGFONT lpft,LPNEWTEXTMETRIC lpnm,DWORD fontType,LPVOID fontcounts)  {  int far\* pCount = (int far\*)fontcounts;  if (fontType & RASTER\_FONTTYPE)  {  pCount[0]++;  }  else if (fontType & TRUETYPE\_FONTTYPE)  {  pCount[2]++;  }  else  {  pCount[1]++;  }  CString fontstr;  fontstr.Format(L"%s\r\n",lpft->lfFaceName);  g\_fontStr += fontstr;  if (pCount[0]++ || pCount[1]++ || pCount[2]++)  {  return TRUE;  }  return FALSE;  }  void CEnumFontsDemoDlg::OnBnClickedBnEnum()  {  // TODO: 在此添加控件通知处理程序代码  HDC hdc;  DWORD fontCounts[] = { 0,0,0 };//注意:这里数组的类型必须是DWORD如果不是,可能会抛异常  hdc = ::GetDC(m\_hWnd);  EnumFontFamilies(hdc, (LPCTSTR)NULL, (FONTENUMPROC)EnumFontProc, (LPARAM)fontCounts);  SetDlgItemText(IDC\_EDIT1, g\_fontStr);  SetDlgItemInt(IDC\_RASTOR, fontCounts[0]);  SetDlgItemInt(IDC\_VECTOR, fontCounts[1]);  SetDlgItemInt(IDC\_TRUETYPE, fontCounts[2]);  } |

### 功能:这个小程序能够枚举系统字体名称显示在大编辑框里面,项目会显示每一种字体类型有多少种字体

|  |
| --- |
|  |