链滴

Win32 汇编学习 (11)：对话框 (2)

作者：Akkuman
原文链接：https://ld246.com/article/1518517508470
来源网站：链滴

我们将进一步学习对话框，探讨如何把对话框当成输入设备。如果您看了前一篇文章，那就会发现这的例子只有少量的改动，就是把我们的对话框窗口附属到主窗口上。另外，我们还要学习通用对话框用法。

<!--more-->

理论：

把对话框当成一个输入设备来用确实是非常地简单，创建完主窗口后，您只要调用函数 DialogBoxParam或 CreateDialogParam 就可以了，前一个函数只要在对话框的过程处理函数中处理相关的消息就以，而后者你必须在消息循环段中插入函数 IsDialogMessage 的调用让它来处理键盘的按键逻辑。为这两个程序段相对来说比较容易，我们就不详解。您可以下载并仔细研究。

下面我们来讨论通用对话框。WINDOWS已经为您准备好了预定义的对话框类，您可以拿来就用，这通用对话框提供给用户以统一的界面。它们包括:打开文件、打印、选择颜色、字体和搜索等。您应该可能地用它们。处理这些对话框的代码在comdlg32.dll中，为了在您的应用程序中使用它们，就必须链接阶段链接库文件 comdlg32.lib。然后调用其中的相关函数即可。对于打开文件通用对话框，该数名为 GetOpenFileName，"保存为..."对话框为 GetSaveFileName，打印通用对话框是 PrintDlg，等等。每一个这样的函数都接收一个指向一个结构体的指针的参数，您可以参考WIN32 API手册得到细的资料，本课中我将讲解创建和使用打开文件对话框。

下面是打开对话框函数 GetOpenFileName 的原型:

```
GetOpenFileName proto lpofn:DWORD
```

您可以看到，该函数只有一个参数，即指向结构体OPENFILENAME的指针。当用户选择了一个文件打开，该函数返回TRUE，否则返回FALSE。接下来我们看看结构体OPENFILENAME的定义:

```
OPENFILENAME  STRUCT
 lStructSize DWORD  ?
 hwndOwner HWND  ?
 hInstance HINSTANCE ?
 lpstrFilter LPCSTR  ?
 lpstrCustomFilter LPSTR  ?
 nMaxCustFilter DWORD  ?
 nFilterIndex DWORD  ?
 lpstrFile LPSTR  ?
 nMaxFile DWORD  ?
 lpstrFileTitle LPSTR  ?
 nMaxFileTitle DWORD  ?
 lpstrInitialDir LPCSTR  ?
 lpstrTitle LPCSTR  ?
 Flags  DWORD  ?
 nFileOffset WORD  ?
 nFileExtension WORD  ?
 lpstrDefExt LPCSTR  ?
 lCustData LPARAM  ?
 lpfnHook DWORD  ?
 lpTemplateName LPCSTR  ?
OPENFILENAME  ENDS
```

好，我们再来看看该结构体中常用的成员的意义:

- lStructSize 结构体OPENFILENAME的大小。

- hwndOwner 拥有打开对话框的窗口的句柄。
- hInstance 拥有该打开文件对话框的应用程序的实例句柄 。
- lpstrFilter 以NULL结尾的一个或多个通配符。通配符是成对出现的，前一部分是描述部分，后一部则是通配符的格式，譬如:

```
FilterString   db "All Files (*.*)",0, "*.*",0
               db "Text Files (*.txt)",0,"*.txt",0,0
```

注意:只有每一对中的第二部分是WINDOWS用来过滤所需选择的文件的，另外您必须在该部分后放一个0，以示字符串的结束。

- nFilterIndex 用来指定打开文件对话框第一次打开时所用的过滤模式串，该索引是从1开始算的，第一个通配符模式的索引是1，第二个是2，譬如上面的例子中，若指定该值为2，则缺省显示的模式就是"*.txt"。
- lpstrFile 需要打开的文件的名称的地址，该名称将会出现在打开文件对话框的编辑控件中，该缓冲不能超过260个字符长，当用户打开文件后，该缓冲区中包含该文件的全路径名，您可以从该缓冲区抽取您所需要的路径或文件名等信息。
- nMaxFile lpstrFile的大小。
- lpstrTitle 指向对话框标题的字符串。
- Flags 该标志决定决定了对话框的风格和特点。
- nFileOffset 在用户打开了一个文件后该值是全路径名称中指向文件名第一个字符的索引。譬如:若路径名为"c:\windows\system\lz32.dll"， 则该值为18。
- nFileExtension 在用户打开了一个文件后该值是全路径名称中指向个文件扩展名第一个字符的索引。

例子:

下例中，我们演示了当用户选择"File->Open"时，将弹出一个打开文件对话框，当用户选择了某个文打开时，会弹出一个对话框，告知要打开的文件的全路径名，文件名和文件扩展名。

```
.386
.model flat,stdcall
option casemap:none
WinMain proto :DWORD,:DWORD,:DWORD,:DWORD
include \masm32\include\windows.inc
include \masm32\include\user32.inc
include \masm32\include\kernel32.inc
include \masm32\include\comdlg32.inc
includelib \masm32\lib\user32.lib
includelib \masm32\lib\kernel32.lib
includelib \masm32\lib\comdlg32.lib

.const
IDM_OPEN equ 1
IDM_EXIT equ 2
MAXSIZE equ 260
OUTPUTSIZE equ 512

.data
ClassName db "SimpleWinClass",0
AppName  db "Our Main Window",0
```

```
MenuName db "FirstMenu",0
ofn   OPENFILENAME <>
FilterString db "All Files",0,"*.*",0
             db "Text Files",0,"*.txt",0,0
buffer db MAXSIZE dup(0)
OurTitle db "-=Our First Open File Dialog Box=-: Choose the file to open",0
FullPathName db "The Full Filename with Path is: ",0
FullName  db "The Filename is: ",0
ExtensionName db "The Extension is: ",0
OutputString db OUTPUTSIZE dup(0)
CrLf db 0Dh,0Ah,0

.data?
hInstance HINSTANCE ?
CommandLine LPSTR ?

.code
start:
    invoke GetModuleHandle, NULL
    mov     hInstance,eax
    invoke GetCommandLine
    mov CommandLine,eax
    invoke WinMain, hInstance,NULL,CommandLine, SW_SHOWDEFAULT
    invoke ExitProcess,eax

WinMain proc hInst:HINSTANCE,hPrevInst:HINSTANCE,CmdLine:LPSTR,CmdShow:DWORD
    LOCAL wc:WNDCLASSEX
    LOCAL msg:MSG
    LOCAL hwnd:HWND
    mov   wc.cbSize,SIZEOF WNDCLASSEX
    mov   wc.style, CS_HREDRAW or CS_VREDRAW
    mov   wc.lpfnWndProc, OFFSET WndProc
    mov   wc.cbClsExtra,NULL
    mov   wc.cbWndExtra,NULL
    push  hInst
    pop   wc.hInstance
    mov   wc.hbrBackground,COLOR_WINDOW+1
    mov   wc.lpszMenuName,OFFSET MenuName
    mov   wc.lpszClassName,OFFSET ClassName
    invoke LoadIcon,NULL,IDI_APPLICATION
    mov   wc.hIcon,eax
    mov   wc.hIconSm,eax
    invoke LoadCursor,NULL,IDC_ARROW
    mov   wc.hCursor,eax
    invoke RegisterClassEx, addr wc
    invoke CreateWindowEx,WS_EX_CLIENTEDGE,ADDR ClassName,ADDR AppName,\
          WS_OVERLAPPEDWINDOW,CW_USEDEFAULT,\
          CW_USEDEFAULT,300,200,NULL,NULL,\
          hInst,NULL
    mov   hwnd,eax
    invoke ShowWindow, hwnd,SW_SHOWNORMAL
    invoke UpdateWindow, hwnd
    .WHILE TRUE
       invoke GetMessage, ADDR msg,NULL,0,0
```

```
        .BREAK .IF (!eax)
        invoke TranslateMessage, ADDR msg
        invoke DispatchMessage, ADDR msg
    .ENDW
    mov     eax,msg.wParam
    ret
WinMain endp

WndProc proc hWnd:HWND, uMsg:UINT, wParam:WPARAM, lParam:LPARAM
    .IF uMsg==WM_DESTROY
        invoke PostQuitMessage,NULL
    .ELSEIF uMsg==WM_COMMAND
        mov eax,wParam
        .if ax==IDM_OPEN
            mov ofn.lStructSize,SIZEOF ofn
            push hWnd
            pop  ofn.hwndOwner
            push hInstance
            pop  ofn.hInstance
            mov  ofn.lpstrFilter, OFFSET FilterString
            mov  ofn.lpstrFile, OFFSET buffer
            mov  ofn.nMaxFile,MAXSIZE
            mov  ofn.Flags, OFN_FILEMUSTEXIST or \
                OFN_PATHMUSTEXIST or OFN_LONGNAMES or\
                OFN_EXPLORER or OFN_HIDEREADONLY
            mov  ofn.lpstrTitle, OFFSET OurTitle
            invoke GetOpenFileName, ADDR ofn
            .if eax==TRUE
                invoke lstrcat,offset OutputString,OFFSET FullPathName
                invoke lstrcat,offset OutputString,ofn.lpstrFile
                invoke lstrcat,offset OutputString,offset CrLf
                invoke lstrcat,offset OutputString,offset FullName
                mov  eax,ofn.lpstrFile
                push ebx
                xor  ebx,ebx
                mov  bx,ofn.nFileOffset
                add  eax,ebx
                pop  ebx
                invoke lstrcat,offset OutputString,eax
                invoke lstrcat,offset OutputString,offset CrLf
                invoke lstrcat,offset OutputString,offset ExtensionName
                mov  eax,ofn.lpstrFile
                push ebx
                xor ebx,ebx
                mov  bx,ofn.nFileExtension
                add eax,ebx
                pop ebx
                invoke lstrcat,offset OutputString,eax
                invoke MessageBox,hWnd,OFFSET OutputString,ADDR AppName,MB_OK
                invoke RtlZeroMemory,offset OutputString,OUTPUTSIZE
            .endif
        .else
            invoke DestroyWindow, hWnd
        .endif
```

```
    .ELSE
        invoke DefWindowProc,hWnd,uMsg,wParam,lParam
        ret
    .ENDIF
    xor     eax,eax
    ret
WndProc endp
end start

;-------------------------------------------------------------------------------
;menu.rc
;-------------------------------------------------------------------------------
#define IDM_OPEN 1
#define IDM_EXIT 2

FirstMenu MENU
{
    MENUITEM "&Open",IDM_OPEN
    MENUITEM "E&xit",IDM_EXIT
}
```

分析:

```
        mov ofn.lStructSize,SIZEOF ofn
        push hWnd
        pop  ofn.hwndOwner
        push hInstance
        pop  ofn.hInstance
```

我们在此填充结构体OPENFILENAME变量ofn的有关成员。

```
        mov  ofn.lpstrFilter, OFFSET FilterString
```

这里FilterString 是文件过滤模式的字符串地址，我们指定的过滤模式字符串如下:

```
FilterString db "All Files",0,"*.*",0
             db "Text Files",0,"*.txt",0,0
```

注意:所有的模式串都是配对的，前一个是描述，后一个才是真正的模式，此处*.*和*.txt是WIONDOS用来寻找匹配的欲打开的文件的。我们当能可以指定任何模式，但是不要忘记在结尾处加0以代表字串已结束，否则您的对话框在操作时可能不稳定。

```
        mov  ofn.lpstrFile, OFFSET buffer
        mov  ofn.nMaxFile,MAXSIZE
```

这里是把缓冲区的地址放到结构体中，同时必须设定大小。以后我们可以随意编辑在该缓冲区中返回信息。

```
        mov  ofn.Flags, OFN_FILEMUSTEXIST or \
           OFN_PATHMUSTEXIST or OFN_LONGNAMES or\
           OFN_EXPLORER or OFN_HIDEREADONLY
```

Flags 中放入的是对话框的风格和特性值。

其中OFN_FILEMUSTEXIST和 OFN_PATHMUSTEXIST要求用户在打开对话框的编辑控件中输入的文

名或路径名必须存在。

OFN_LONGNAMES 告诉对话框显示长文件名。

OFN_EXPLORER 告诉WINDOWS对话框的外观必须类似资源管理器。

OFN_HIDEREADONLY 指定不要显示只读文件(既使它的扩展名符合过滤模式)。

除此之外，还有许多其它的标志位，您可以参考有关WIN32 API手册。

```
mov  ofn.lpstrTitle, OFFSET OurTitle
```

指定打开文件对话框的标题名。

```
invoke GetOpenFileName, ADDR ofn
```

调用GetOpenFileName函数，并传入指向结构体ofn的指针。

这时候，打开文件对话框就显示出来了，GetOpenFileName函数要一直等到用户选择了一个文件后会返回，或者当用户按下了CANCEL键或关闭对话框时。

当用户选择了打开一个文件时，该函数返回TRUE， 否则返回FALSE。

```
.if eax==TRUE
   invoke lstrcat,offset OutputString,OFFSET FullPathName
   invoke lstrcat,offset OutputString,ofn.lpstrFile
   invoke lstrcat,offset OutputString,offset CrLf
   invoke lstrcat,offset OutputString,offset FullName
```

当用户选择打开一个文件时，我们就在一个对话框中显示一个字符串，我们先给OutputString变量分内存，然后调用PAI 函数lstrcat，把所有的字符串连到一起，为了让这些字符串分行显示，我们必须每个字符串后面加一个换行符。

```
   mov  eax,ofn.lpstrFile
   push ebx
   xor  ebx,ebx
   mov  bx,ofn.nFileOffset
   add  eax,ebx
   pop  ebx
   invoke lstrcat,offset OutputString,eax
```

上面这几行可能需要一些解释。nFileOffset的值等于被打开文件的全路径名中的文件名的第一个字符索引，由于nFileOffset是一个WORD型变量，而lpstrFile是一个DWORD型的指针，所以我们就要作转换把nFileOffset存入ebx寄存器的底字节，然后再加到eax寄存器中得到DWORD型的指针。

```
   invoke MessageBox,hWnd,OFFSET OutputString,ADDR AppName,MB_OK
```

我们在对话框中显示该字符串。

```
   invoke RtlZeroIMemory,offset OutputString,OUTPUTSIZE
```

为了下一次能正确地显示，必须清除缓冲区，我们调用函数RtlZeroIMemory来做这件事。