www.mohandesyar.com

عنـــــــوان

آشنایی با برنامه نویسی اسمبلی ویندوز

پژوهش و نگارش : وحید نصیری

بهار ۱۳۸۴

قسمت سوم

نگارش ۱

چاپ و یا نشر غیر الکترونیکی این مطالب بدون مجوز کتبی از طرف نویسنده به هر نحوی غیرمجاز است.

انتشار این مطالب بر روی اینترنت و یا استفاده از آن به صورت مستقیم و یا غیر مستقیم در نشریات الکترونیکی بلا مانع است.

مثال ۱۵– طریقه ساخت DLL ها در ویندوز

برنامه نویسان به تجربه در می‌یابند که قسمتی از کدهای نوشته شده توسط آنها در برنامه های مختلف، یکسان و مشترک است. تحت داس ، برنامه نویسان قسمت‌های مشترک کدهای خود را در کتابخانه ها قرار می‌دادند و هنگام نیاز، کتابخانه را به فایل obj برنامه متصل ساخته ، برنامه linker توابع را از کتابخانه استخراج کرده و در فایل اجرایی نهایی قرار می‌داد. به این روش static linking نیز گفته می‌شود. از معایب این روش، می‌توان به اتلاف فضای دیسک سخت با برنامه‌هایی که یک سری توابع یکسان را در خود دارند، اشاره کرد. البته تحت داس ازآنجائیکه عموما تنها یک برنامه فعال در حافظه وجود دارد ، اتلاف حافظه در این حالت حداقل خواهد بود.

تحت ویندوز این موارد با فعال بودن چندین برنامه به صورت همزمان ، بحرانی‌تر گشته و حافظه به سرعت توسط برنامه‌های حجیم مصرف خواهد شد. ویندوز راه حلی را برای مقابله با این مشکل تحت عنوان dynamic link libraries ارائه داده است (DLL) .

یک DLL از مجموعه ای از توابع تشکیل شده است. اگر در یک زمان چندین وهله از برنامه شما در حال اجرا باشند ، ویندوز تنها یکبار DLL های وابسته به آنرا به درون حافظه بارگذاری خواهد نمود.

لازم به ذکر است ، تمام پروسه‌هایی که از یک dll استفاده می‌کنند ، از کپی‌های مخصوص به خود آن ، کمک می‌گیرند. در اینجا به نظر خواهد رسید که کپی‌های بسیاری از یک dll در حافظه قرار گرفته اند. در عمل ویندوز با بکارگیری paging ، کدهای dll یکسانی را به اشتراک خواهد گذاشت. بنابراین در حافظه فیزیکی تنها یک کپی از dll قرار خواهد گرفت ، اما هر پروسه قسمت data منحصربفرد خود را خواهد داشت.

هنگام بکارگیری یک dll ، برنامه تنها در زمان اجرا (برخلاف static library) به آن اتصال برقرار کرده و از کدهای آن استفاده خواهد کرد، به همین جهت به آنها dynamic link library می‌گویند.همچنین امکان unload کردن آنها نیز در زمان عدم نیاز مهیا است.اگر برنامه شما تنها برنامه‌ی استفاده کننده از dll باشد ، هنگام unload ، بلافاصله dll از حافظه خارج می‌گردد. اما اگر سایر پروسه‌ها نیز مشغول استفاده از کدهای dll باشند تا زمان unload شدن dll توسط آخرین برنامه استفاده کننده از آن، در حافظه باقی خواهد ماند.

در اینجا برنامه linker کار دشوارتری را در جهت تولید فایل اجرایی نهایی خواهد داشت ، زیرا امکان استخراج و قرار دادن توابع را در فایل اجرایی ندارد. بنابراین باید اطلاعات دقیقی از فایل dll و همچنین تعاریف توابع قرار گرفته در آنرا در فایل اجرایی قرار داد ، تا هنگام اجرا ، امکان بارگذاری آنها میسر باشد.برای این منظور import library که حاوی اطلاعاتی در مورد dll می‌باشد ، معرفی شده‌است. Linker اطلاعات لازم را از این فایل دریافت کرده و درون فایل اجرایی قرار خواهد داد.

www.mohandesyar.com

هنگامیکه Windows loader ، برنامه را بدرون حافظه بارگذاری می کند ، متوجه خواهد شد که برنامه به یک dll متصل شده و پس از جستجو و یافتن آن ، آنرا بدرون فضای آدرس دهی برنامه نگاشت کرده و همچنین آدرس‌های بکار گرفته شده در برنامه را جهت فراخوانی توابع آن تصحیح می کند.

همچنین امکان بارگذاری dll ها بدون کمک گیری از Windows loader نیز میسر است. این روش مزایا و معایب خاص خودش را دارا است:

- نیازی به استفاده از import libraries برای هر نوع dll ایی وجود نخواهد داشت. هرچند باید اطلاعات دقیقی در مورد نحوه تعریف توابع مورد نظر ، به همراه تعداد و نوع پارامترهای آن داشت.
- در حالت استفاده از windows loader ، اگر dll مورد نظر یافت نشود ، با پیغام "A required .DLL file, xxxxx.dll is missing" برنامه خاتمه خواهد یافت. اما اگر به صورت دستی dll را بارگذاری نماییم ، امکان ادامه برنامه نیز وجود خواهد داشت.
- امکان استفاده از توابعی که در import libraries لحاظ نشده اند نیز وجود دارد.
- اگر از تابع LoadLibrary استفاده نمائیم ، نیاز به فراخوانی تابع GetProcAddress به ازای فراخوانی هر تابعی می‌باشد. تابع GetProcAddress ، آدرس نقطه آغاز یک تابع را در dll ایی خاص بر می گرداند. در این حالت اندازه کدنهایی کمی بیشتر و کندتر خواهد بود (البته نه زیاد).

کدی که در ادامه ارائه می‌شود، قالب نوشتن یک dll به زبان اسمبلی می‌باشد:

```
;------------------------------------------------------------------------------
;                       DLLSkeleton.asm
;------------------------------------------------------------------------------


.386
.model flat,stdcall
option casemap:none
include \masm32\include\windows.inc
include \masm32\include\user32.inc
include \masm32\include\kernel32.inc
includelib \masm32\lib\user32.lib
includelib \masm32\lib\kernel32.lib

.data
.code
DllEntry proc hInstDLL:HINSTANCE, reason:DWORD, reserved1:DWORD
        mov  eax,TRUE
        ret
DllEntry Endp
```


www.mohandesyar.com

```
;-----------------------------------------------------------------------------------------------
;                                    This is a dummy function
; It does nothing. I put it here to show where you can insert functions into
; a DLL.
;-----------------------------------------------------------------------------------------------
TestFunction proc
    ret
TestFunction endp

End DllEntry


;-----------------------------------------------------------------------------
;                          DLLSkeleton.def
;-----------------------------------------------------------------------------
LIBRARY   DLLSkeleton
EXPORTS   TestFunction
```

قالب یک فایل dll را در بالا ملاحظه فرمودید. هر dll باید دارای یک تابع entrypoint باشد. ویندوز، تابع entrypoint را هربار در موارد زیر فراخوانی می کند:

- dll برای اولین بار بارگذاری شود.
- dll از حافظه خارج و unload شود.
- تردی در پروسه‌ای مشابه ایجاد شود.
- تردی در پروسه‌ای مشابه خاتمه یابد.

```
DllEntry  proc  hInstDLL:HINSTANCE, reason:DWORD, reserved1:DWORD
      mov  eax,TRUE
      ret
DllEntry Endp
```

نام تابع entrypoint را هر چیزی می‌توان در نظر گرفت. این تابع سه پارامتر را پذیرفته و دو مورد آن مهم هستند:

hInstDLL : دستگیره ماژول dll بوده و با دستگیره وهله برنامه یکی نیست. بهتر است آنرا برای استفاده‌های بعدی ذخیره نمود.

reason : یکی از چهار مقدار زیر می‌تواند باشد :

DLL_PROCESS_ATTACH : هنگامیکه dll برای اولین بار به درون فضای آدرس دهی پروسه تزریق می‌شود ، این مقدار را دریافت خواهد کرد. از آن برای مقدار دهی اولیه و انجام کارهای اولیه بهتر است استفاده گردد.

www.mohandesyar.com

DLL_PROCESS_DETACH : هنگامیکه dll در حال unload شدن از فضای آدرس دهی پروسه است ، این مقدار دریافت می‌شود.از این موقعیت بهتر است برای آزاد سازی حافظه و منابع تخصیص داده شده استفاده شود.

DLL_THREAD_ATTACH : هنگامیکه پروسه ترد جدیدی را ایجاد نماید ، dll این مقدار را دریافت خواهد کرد.

DLL_THREAD_DETACH : هنگامیکه در پروسه تردی تخریب شود ، dll این مقدار را دریافت خواهد کرد.

اگر نیاز باشد تا dll بارگذاری نشود ، خروجی تابع که مساوی FALSE است در EAX قرار خواهد گرفت. برای مثال اگر نیاز باشد تا dll هنگام بارگذاری مقدار مشخصی حافظه را تخصیص دهد اما موفق به انجام اینکار نگردد ،تابع entrypoint باید FALSE برگرداند (تا مشخص شود که dll قادر به اجرا نیست).

توابع را می‌توان پیش و یا پس از تابع entrypoint تعریف کرد. برای اینکه این توابع در سایر برنامه‌ها قابل استفاده باشد، باید نام آنها را در لیست خروجی یک فایل تعریف ماژول (.def) ، قرار داد. برای مثال:

```
LIBRARY   DLLSkeleton
EXPORTS   TestFunction
```

اولین خط ذکر شده اجباری است. عبارت **LIBRARY** ، تعریف کننده‌ی نام درونی ماژول dll است (باید با نام فایل dll هماهنگ باشد). عبارت **EXPORTS** ، بیانگر تعریف توابعی است که توسط سایر برنامه ها قابل فراخوانی است.

طریقه کامپایل آن نیز به شکل زیر است :

```
Link  /DLL  /SUBSYSTEM:WINDOWS /DEF:DLLSkeleton.def  /LIBPATH:c:\masm32\lib DLLSkeleton.obj
```

حاصل نهایی، یک فایل dll و یک فایل lib می باشد. فایل lib یک import library می‌باشد

در ادامه طریقه بارگذاری و استفاده از dll بررسی می گردد:

```
;-----------------------------------------------------------------------------------
;                                UseDLL.asm
;-----------------------------------------------------------------------------------
.386
.model flat,stdcall
option casemap:none
include \masm32\include\windows.inc
include \masm32\include\user32.inc
include \masm32\include\kernel32.inc
includelib \masm32\lib\kernel32.lib
includelib \masm32\lib\user32.lib
```


www.mohandesyar.com

```
.data
LibName db "DLLSkeleton.dll",0
FunctionName db "TestHello",0
DllNotFound db "Cannot load library",0
AppName db "Load Library",0
FunctionNotFound db "TestHello function not found",0

.data?
hLib dd ?                       ; the handle of the library (DLL)
TestHelloAddr dd ?              ; the address of the TestHello function

.code
start:
    invoke LoadLibrary,addr LibName
;-------------------------------------------------------------------------------------------------------
; Call LoadLibrary with the name of the desired DLL. If the call is successful
; it will return the handle to the library (DLL). If not, it will return NULL
; You can pass the library handle to GetProcAddress or any function that requires
; a library handle as a parameter.
;-------------------------------------------------------------------------------------------------------
    .if eax==NULL
        invoke MessageBox,NULL,addr DllNotFound,addr AppName,MB_OK
    .else
        mov hLib,eax
        invoke GetProcAddress,hLib,addr FunctionName
;-------------------------------------------------------------------------------------------------------
; When you get the library handle, you pass it to GetProcAddress with the address
; of the name of the function in that DLL you want to call. It returns the address
; of the function if successful. Otherwise, it returns NULL
; Addresses of functions don't change unless you unload and reload the library.
; So you can put them in global variables for future use.
;-------------------------------------------------------------------------------------------------------
        .if eax==NULL
            invoke MessageBox,NULL,addr FunctionNotFound,addr AppName,MB_OK
        .else
            mov TestHelloAddr,eax
            call [TestHelloAddr]
;-------------------------------------------------------------------------------------------------------
; Next, you can call the function with a simple call with the variable containing
; the address of the function as the operand.
;-------------------------------------------------------------------------------------------------------
        .endif
        invoke FreeLibrary,hLib
;-------------------------------------------------------------------------------------------------------
; When you don't need the library anymore, unload it with FreeLibrary.
;-------------------------------------------------------------------------------------------------------
    .endif
    invoke ExitProcess,NULL
end start
```


www.mohandesyar.com

مثال ۱۶– Common Controls

تحت ویندوز بهبودهای زیادی در رابط گرافیکی کاربر صورت گرفته است. این بهبودها توسط کنترل‌هایی در اختیار برنامه نویس‌ها می‌باشند. کنترل‌های جدید به شرح زیر هستند:

- Toolbar
- Tooltip
- Status bar
- Property sheet
- Property page
- Tree view
- List view
- Animation
- Drag list
- Header
- Hot-key
- Image list
- Progress bar
- Rich edit
- Tab
- Trackbar
- Up-down

بجز rich edit control (که در richedXX.dll ذخیره شده) ، مابقی کنترل‌های فوق در فایل comctl32.dll تعریف شده‌اند. با فراخوانی تابع InitCommonControls دربرنامه، فایل comctl32.dll در حافظه بارگذاری خواهد شد. این تابع در فایل comctl32.dll قرار داشته و فراخوانی آن سبب می شود تا PE loader ، آنرا بارگذاری نماید. در این حالت تابع entrypoint دی ال ال فوق ، تمام کنترلهای ذکر شده را رجیستر می کند. اکنون با استفاده از یک resource editor و یا ایجاد دستی کنترل‌ها، می‌توان از آنها در دیالوگ باکس‌ها استفاده نمود.

با فراخوانی CreateWindowEx or CreateWindow به همراه نام کلاس کنترل ، امکان ایجاد کنترل‌های فوق وجود دارد. هرچند بعضی از این کنترل‌ها توابع مخصوصی جهت ایجاد نیز دارند، اما لازم به ذکر است که این توابع صرفا محصورکننده‌هایی برای CreateWindowEx بشمار می‌روند. نمونه ای از این توابع به شرح زیر هستند:

- CreateToolbarEx
- CreateStatusWindow
- CreatePropertySheetPage
- PropertySheet
- ImageList_Create

برای ایجاد common controls نیاز است تا با نام کلاس های مرتبط با آنها آشنا گردید:

www.mohandesyar.com

Class Name	Common Control
ToolbarWindow32	Toolbar
tooltips_class32	Tooltip
msctls_statusbar32	Status bar
SysTreeView32	Tree view
SysListView32	List view
SysAnimate32	Animation
SysHeader32	Header
msctls_hotkey32	Hot-key
msctls_progress32	Progress bar
RICHEDIT	Rich edit
msctls_updown32	Up-down
SysTabControl32	Tab

Property sheets ، property pages و image list control ، تابع مخصوص ایجاد خود را دارا می‌باشند. اسامی کلاس‌های ارائه شده در جدول فوق با resource editor مربوط به VC++ کاملا مطابقت داده شده است.

این کنترل‌ها از سبک های عمومی پنجره مانند WS_CHILD و غیره، علاوه بر سبک های اختصاصی مانند TVS_XXXXX برای tree view control ، LVS_xxxx برای list view control ، می‌توان استفاده کرد. برای مشاهده جزئیات دقیق این خواص بهتر است به MSDN و یا راهنمای API ویندوز مراجعه نمود.

این کنترل‌ها برای تبادل اطلاعات با پنجره والد از پیغام‌های WM_NOTIFY استفاده می‌نمایند. همچنین پیغامهای دیگر نیز برای کنترل‌های خاصی طراحی شده‌اند که برای مشاهده جزئیات بیشتر آنها می‌توان به راهنمای API ویندوز مراجعه کرد.

در مثال زیر قصد استفاده از progress bar and status bar را داریم:

```
 .386
.model flat,stdcall
option casemap:none
include \masm32\include\windows.inc
include \masm32\include\user32.inc
include \masm32\include\kernel32.inc
include \masm32\include\comctl32.inc
includelib \masm32\lib\comctl32.lib
includelib \masm32\lib\user32.lib
includelib \masm32\lib\kernel32.lib

WinMain PROTO :DWORD,:DWORD,:DWORD,:DWORD

.const
IDC_PROGRESS equ 1          ; control IDs
IDC_STATUS equ 2
IDC_TIMER  equ 3

.data
ClassName  db "CommonControlWinClass",0
```


www.mohandesyar.com

```
AppName    db "Common Control Demo",0
ProgressClass  db "msctls_progress32",0       ; the class name of the progress bar
Message  db "Finished!",0
TimerID  dd 0

.data?
hInstance  HINSTANCE ?
hwndProgress dd ?
hwndStatus dd ?
CurrentStep dd ?
.code
start:
   invoke GetModuleHandle, NULL
   mov    hInstance,eax
   invoke WinMain, hInstance,NULL,NULL, SW_SHOWDEFAULT
   invoke ExitProcess,eax
   invoke InitCommonControls

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_APPWORKSPACE
   mov   wc.lpszMenuName,NULL
   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_OVERLAPPED+WS_CAPTION+WS_SYSMENU+WS_MINIMIZEBOX+WS_MAXIMIZEBOX+WS_VISI
BLE,CW_USEDEFAULT,\
         CW_USEDEFAULT,CW_USEDEFAULT,CW_USEDEFAULT,NULL,NULL,\
         hInst,NULL
   mov   hwnd,eax
   .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_CREATE
       invoke CreateWindowEx,NULL,ADDR ProgressClass,NULL,\
         WS_CHILD+WS_VISIBLE,100,\
         200,300,20,hWnd,IDC_PROGRESS,\
```


www.mohandesyar.com

```
            hInstance,NULL
        mov hwndProgress,eax
        mov eax,1000            ; the lParam of PBM_SETRANGE message contains the range
        mov CurrentStep,eax
        shl eax,16              ; the high range is in the high word
        invoke SendMessage,hwndProgress,PBM_SETRANGE,0,eax
        invoke SendMessage,hwndProgress,PBM_SETSTEP,10,0
        invoke CreateStatusWindow,WS_CHILD+WS_VISIBLE,NULL,hWnd,IDC_STATUS
        mov hwndStatus,eax
        invoke SetTimer,hWnd,IDC_TIMER,100,NULL      ; create a timer
        mov TimerID,eax
    .elseif uMsg==WM_DESTROY
        invoke PostQuitMessage,NULL
        .if TimerID!=0
            invoke KillTimer,hWnd,TimerID
        .endif
    .elseif uMsg==WM_TIMER      ; when a timer event occurs
        invoke SendMessage,hwndProgress,PBM_STEPIT,0,0    ; step up the progress in the progress bar
        sub CurrentStep,10
        .if CurrentStep==0
            invoke KillTimer,hWnd,TimerID
            mov TimerID,0
            invoke SendMessage,hwndStatus,SB_SETTEXT,0,addr Message
            invoke MessageBox,hWnd,addr Message,addr AppName,MB_OK+MB_ICONINFORMATION
            invoke SendMessage,hwndStatus,SB_SETTEXT,0,0
            invoke SendMessage,hwndProgress,PBM_SETPOS,0,0
        .endif
    .else
        invoke DefWindowProc,hWnd,uMsg,wParam,lParam
        ret
    .endif
    xor eax,eax
    ret
WndProc endp
end start
```

بررسی کد فوق:

```
    invoke WinMain, hInstance,NULL,NULL, SW_SHOWDEFAULT
    invoke ExitProcess,eax
    invoke InitCommonControls
```

عمدا تابع InitCommonControls پس از ExitProcess فراخوانی شده است تا متذکر شویم که هدف از فراخوانی این تابع در کد ، صرفا ایجاد مرجعی به فایل comctl32.dll است (این تابع هیچ کار خاص دیگری را انجام نمی‌دهد). در ادامه :

```
        .if uMsg==WM_CREATE
            invoke CreateWindowEx,NULL,ADDR ProgressClass,NULL,\
              WS_CHILD+WS_VISIBLE,100,\
              200,300,20,hWnd,IDC_PROGRESS,\
              hInstance,NULL
           mov hwndProgress,eax
```


www.mohandesyar.com

به این صورت یک common control ایجاد می شود. تابع CreateWindowEx ، دستگیره پنجره والد (hWnd) را نیاز دارد. همچنین control ID ، مشخص کننده کنترل است. از آنجائیکه hWnd را در اختیار داریم نیازی به ذکر این ID نیست. سبک تمام child window controls باید مساوی S_CHILD باشد. سپس :

```
mov eax,1000
mov CurrentStep,eax
shl eax,16
invoke SendMessage,hwndProgress,PBM_SETRANGE,0,eax
invoke SendMessage,hwndProgress,PBM_SETSTEP,10,0
```

پس از ایجاد progress bar ، بازه آنرا مشخص می کنیم. بازه پیش فرض آن صفر تا ۱۰۰ است. اگر مایل به تغییر آن بودید می‌توان از پیغام PBM_SETRANGE استفاده کرد. lParam در این پیغام مشخص کننده بازه می‌باشد. high word حد بالایی بازه و low word حدپایینی بازه است. میزان پیشرفت نشانگر آن را در یک مرحله ، می‌توان توسط پیغام PBM_SETSTEP مشخص نمود. در این مثال، این میزان معادل ۱۰ درنظر گرفته شده است. بنابراین با فرستادن پیغام PBM_STEPIT ، به نشانگر ۱۰ واحد اضافه می گردد. همچنین این میزان با پیغام PBM_SETPOS نیز قابل تغییر است. این پیغام امکان کنترل دقیق تری را بر روی progress bar میسر می‌سازد. سپس:

```
invoke CreateStatusWindow,WS_CHILD+WS_VISIBLE,NULL,hWnd,IDC_STATUS
mov hwndStatus,eax
invoke SetTimer,hWnd,IDC_TIMER,100,NULL        ; create a timer
mov TimerID,eax
```

در ادامه با فراخوانی CreateStatusWindow ، یک status bar ایجاد شده است. سپس تایمر ایجاد می گردد. در این مثال ، میزان پیشرفت نشانگر progress bar را هر 100 ms یکبار ، به روز در می‌آوریم. تعریف تابع SetTimer به صورت زیر است:

```
SetTimer PROTO hWnd:DWORD, TimerID:DWORD, TimeInterval:DWORD, lpTimerProc:DWORD
```

hWnd : دستگیره پنجره والد

TimerID :شناسه تایمر (مقداری غیر صفر)

TimerInterval :فواصل زمانی فراخوانی تابع تایمر، یا ارسال پیغام WM_TIMER ، برحسب میلی ثانیه.

lpTimerProc : آدرس رویه تایمر است که در فواصل زمانی مشخصی اجرا می گردد.اگر این مقدار مساوی NULL قرار گیرد، پیغام WM_TIMER به پنجره والد ارسال می گردد.

www.mohandesyar.com

اگر این فراخوانی موفقیت آمیز باشد خروجی تابع TimerID خواهد بود ، در غیراینصورت صفر بر می گرداند. سپس:

```
.elseif uMsg==WM_TIMER
   invoke SendMessage,hwndProgress,PBM_STEPIT,0,0
   sub CurrentStep,10
   .if CurrentStep==0
      invoke KillTimer,hWnd,TimerID
      mov TimerID,0
      invoke SendMessage,hwndStatus,SB_SETTEXT,0,addr Message
      invoke MessageBox,hWnd,addr Message,addr AppName,MB_OK+MB_ICONINFORMATION
      invoke SendMessage,hwndStatus,SB_SETTEXT,0,0
      invoke SendMessage,hwndProgress,PBM_SETPOS,0,0
   .endif
```

در فواصل زمانی به روزرسانی رویه تایمر، پیغام WM_TIMER به پنجره والد ارسال می گردد. بنابراین کد مورد را نظر را در این قسمت می‌توان قرار داد (در این مثال، موقعیت نشانگر progress bar را به روز درخواهیم آورد و همچنین میزان پیشرفت را با توجه به حد بالایی تعریف شده برای آن ، کنترل خواهیم کرد).

www.mohandesyar.com

مثال ۱۶– Tree View Control

در این مثال قصد فراگیری استفاده از tree view control و image list را به همراه آن خواهیم داشت.

یک tree view control ، نوع خاصی از پنجره است برای نمایش سلسله مراتبی اشیاء به همراه ارتباطات بین آنها. با فراخوانی CreateWindowEx به همراه SysTreeView32 بعنوان نام کلاس ، می‌توان آنرا ایجاد نمود. همچنین فراخوانی تابع InitCommonControls را نیز فراموش نکنید.

سبک‌های نمایشی مختلفی برای tree view control تعریف شده اند:

TVS_HASBUTTONS: علائم + و – را در کنار آیتم های والد نمایش می‌دهد.

TVS_HASLINES : از خطوط، جهت نمایش سلسله مراتب اشیاء استفاده می‌شود.

TVS_LINESATROOT: از خطوط، جهت پیوند دادن آیتم ها در ریشه کنترل کمک می گیرد.

این کنترل نیز همانند سایر کنترل‌ها از پیغام‌ها جهت تبادل اطلاعات با پنجره والد استفاده می‌نماید. پیغامی که اینجا ارسال خواهد شد، **WM_NOTIFY** است. در اینجا **wParam** ، ID کنترل را دربر خواهد داشت.از آنجائیکه ضمانتی جهت یکتا بودن این مقدار وجود ندارد ، ما از آن استفاده نخواهیم کرد. بجای آن از اعضای ساختار NMHDR ، اشاره شده توسط lParam کمک می گیریم. این ساختار به صورت زیر تعریف شده است:

```
NMHDR struct DWORD
   hwndFrom    DWORD ?
   idFrom      DWORD ?
   code        DWORD ?
NMHDR ends
```

hwndFrom : دستگیره پنجره کنترلی است که پیغام **WM_NOTIFY** را ارسال می کند.

idFrom : ID کنترلی است که پیغام **WM_NOTIFY** را ارسال می کند.

code : پیغام اصلی می‌باشد که کنترل مایل است تا آنرا به پنجره والد ارسال کند.

www.mohandesyar.com

پیغامهای ارسالی به کنترل tree view با _TVM شروع می‌شوند (برای مثال **TVM_CREATEDRAGIMAGE**). پیغام‌های ارسال شده توسط کنترل tree view ، با _TVN شروع می‌گردند (و در عضو code ساختار NMHDR قرار خواهند گرفت).

اضافه کردن آیتم‌ها به کنترل tree view :

پس از ایجاد کنترل ، می‌توان آیتم ها را بدان افزود. اینکار با ارسال پیغام **TVM_INSERTITEM** به آن ، قابل انجام است.

```
TVM_INSERTITEM
wParam = 0;
lParam = pointer to a TV_INSERTSTRUCT;
```

در اینجا لازم است تا با تعدادی از واژه‌های مرتبط با کنترل tree view و روابط بین آنها آشنا شویم. یک آیتم می‌تواند فرزند ، والد و یا در یک لحظه هر دو مورد باشد.یک والد ، آیتمی است مرتبط با چند subitem . یک والد می‌تواند فرزند آیتم‌های دیگر نیز باشد. به آیتمی بدون والد ، آیتم ریشه نیز گفته می‌شود. در ادامه با ساختار **TV_INSERTSTRUCT** آشنا خواهیم شد:

```
TV_INSERTSTRUCT STRUCT DWORD
  hParent      DWORD     ?
  hInsertAfter  DWORD ?
                ITEMTYPE <>
TV_INSERTSTRUCT ENDS
```

hParent : دستگیره‌ای است به آیتم والد. اگر این عضو با **TVI_ROOT** یا NULL مقدار دهی شود ، آیتم در ریشه کنترل قرار خواهد گرفت.

hInsertAfter : دستگیره ای به آیتمی که اضافه شده است، می‌باشد. همچنین یکی از مقادیر زیر نیز می‌تواند باشد:

TVI_FIRST : آیتم را در ابتدای لیست اضافه می کند.

TVI_LAST : آیتم را در انتهای لیست اضافه می کند.

TVI_SORT : آیتم های اضافه شده را بر اساس حروف الفباء اضافه می کند.

www.mohandesyar.com

از ساختارهای زیر جهت ارسال و دریافت اطلاعات پیرامون آیتم های کنترل tree view ، بر اساس پیغام های دریافتی ، استفاده می‌شود.

```
ITEMTYPE UNION
      itemex TVITEMEX <>
      item TVITEM <>
ITEMTYPE ENDS

TV_ITEM STRUCT DWORD
  imask           DWORD     ?
  hItem           DWORD     ?
  state           DWORD     ?
  stateMask       DWORD     ?
  pszText         DWORD     ?
  cchTextMax      DWORD     ?
  iImage          DWORD     ?
  iSelectedImage  DWORD     ?
  cChildren       DWORD     ?
  lParam          DWORD     ?
TV_ITEM ENDS
```

برای مثال توسط پیغام **TVM_INSERTITEM** ، خواص آیتمی را که قرار است به tree view control اضافه شود، می‌توان تعیین کرد و یا توسط پیغام **TVM_GETITEM** ، اطلاعاتی پیرامون آیتم انتخابی دریافت می گردد.

imask : مشخص می کند که کدامیک از اعضای ساختار **TV_ITEM** معتبر هستند. برای مثال اگر مقدار آن مساوی **TVIF_TEXT** باشد، تنها عضو pszText معتبر خواهد بود. اینجا پرچم‌های مختلفی را می‌توان با هم ترکیب کرد.

hItem : دستگیره ای است به یکی از آیتم‌های کنترل. هر آیتم، دستگیره مخصوص به خود را دارد. اگر نیاز باشد تا کاری را بر روی آیتم انجام دهیم ، باید مقدار دستگیره آنرا داشت.

pszText : اشاره گری است به رشته ای مختوم به نال که برچسب آیتمی را مشخص می‌نماید.

cchTextMax : برای دریافت عنوان برچسپ یک آیتم کنترل بکار می رود. این عضو اندازه بافر مورد نیاز برای دریافت این متن را در **pszText** ، مشخص می‌نماید.

iImage and **iSelectedImage** : اندیس تصویر یک آیتم در حالت انتخاب شده و انتخاب نشده را از یک image list ، مشخص می‌نماید.

جهت قرار دادن یک آیتم در tree view control ، باید hParent و hInsertAfter پر شده و همچنین imask and pszText نیز مقدار دهی شوند.

www.mohandesyar.com

اضافه کردن تصویر به کنترل tree view :

برای اضافه کردن تصویر به آیتم‌های tree view کنترل ، باید یک image list را ایجاد (توسط فراخوانی **ImageList_Create**) و سپس آنرا به کنترل منتسب کرد. تعریف این تابع به صورت زیر است:

```
ImageList_Create PROTO cx:DWORD, cy:DWORD, flags:DWORD, \
cInitial:DWORD,  cGrow:DWORD
```

حاصل فراخوانی موفقیت آمیز این تابع ، دستگیره ای خواهد بود به یک image list خالی.

cx : عرض هر تصویر در image list برحسب پیکسل.

cy : ارتفاع هر تصویر در image list برحسب پیکسل. هر تصویر در image list ، باید با سایر تصاویر موجود در آن هم اندازه باشد. اگر تصویر بزرگی را مشخص نمایید ، ویندوز آنرا به قطعات کوچکتری مطابق عرض و ارتفاع ذکر شده، تبدیل خواهد کرد.

flags : رنگی و یا تک رنگ بودن تصاویر را در image list مشخص می کند. برای مطالعه جزئیات بیشتر به MSDN مراجعه نمایید.

cInitial : تعداد تصاویر image list را مشخص می‌نماید. ویندوز از این عدد برای تخصیص حافظه مورد نیاز کمک می گیرد.

cGrow : این پارامتر بیانگر تعداد تصاویر جدیدی است که به image list قرار است اضافه شوند.

پس از ایجاد یک image list ، با فراخوانی تابع **ImageList_Add** ، می‌توان تصاویر را به آن اضافه کرد. تعریف این تابع به صورت زیر است:

```
ImageList_Add PROTO himl:DWORD, hbmImage:DWORD, hbmMask:DWORD
```

با فراخوانی ناموفق این تابع ، خروجی منفی یک ، حاصل خواهد شد.

himl : دستگیره ای به یک image list که از آن برای افزودن تصاویر استفاده می‌شود. این مقدار با فراخوانی موفقیت آمیز تابع **ImageList_Create** بدست می‌آید.

hbmImage : دستگیره ای است به بیت مپی که قرار است به image list اضافه شود. بطور معمول بیت‌مپ ها در فایل های ریسورس ذخیره شده و سپس با تابع **LoadBitmap** فراخوانی می‌شوند.

hbmMask : دستگیره بیت مپی است که حاوی mask می‌باشد. اگر هیچگونه mask ایی در image list استفاده نشود از این پارامتر صرفنظر خواهد شد.

www.mohandesyar.com

بطور معمول تنها از دو تصویر برای نمایش انتخاب شدن و عدم انتخاب یک آیتم در کنترل tree view استفاده می‌شود. هنگامیکه یک image list آماده شد ، با فرستادن پیغام **TVM_SETIMAGELIST** به کنترل tree view ، آنرا به کنترل منتسب خواهیم کرد.

TVM_SETIMAGELIST :

wParam : نوع image list را برای تنظیم مشخص می کند.برای این حالت دو انتخاب وجود دارد:

TVSIL_NORMAL : بیانگر image list متداولی است که شامل تصاویر انتخاب و عدم انتخاب آیتم‌های tree view control است.

TVSIL_STATE : بیانگر image list ایی است که تصاویر قرار گرفته در آن ، در حالت تعریف شده توسط کاربر هستند.

lParam : دستگیره ای به image list است.

دریافت اطلاعات درباره آیتم های کنترل tree view :

با ارسال پیغام **TVM_GETITEM** می‌توان اطلاعاتی را درباه آیتم‌های کنترل tree view ، بدست آورد.

TVM_GETITEM :

wParam = 0

lParam : اشاره گری است به ساختار **TV_ITEM** که قرار است با اطلاعات مورد نیاز پر شود.

قبل از ارسال این پیغام باید توسط عضو imask مشخص نمود که کدامیک از اعضای **TV_ITEM** باید توسط ویندوز مقدار دهی شوند.مهمترین عضوی که باید مقدار دهی شود hItem می باشد که باید با دستگیره ای به آیتمی که قرار است اطلاعات از آن دریافت شود ، مقدار دهی گردد.این مورد مشکل ساز است! به چه صورت باید دستگیره آیتم مورد نظر را بدست آورد؟ آیا باید تمامی این مقادیر را ذخیره نمود؟

با ارسال پیغام **TVM_GETNEXTITEM** به کنترل tree view به همراه مشخصات درخواستی، می‌توان این دستگیره را بدست آورد. برای مثال می توان دستگیره آیتم انتخابی ، اولین آیتم فرزند و غیره را دریافت کرد.

www.mohandesyar.com

TVM_GETNEXTITEM:

wParam : پرچم

lParam : دستگیره ای به یک آیتم کنترل tree view . (تنها برای بعضی مقادیر پرچم‌ها مورد نیاز است)

مقادیر پرچم در این حالت بسیار مهم بوده و تمامی آنها در ادامه ذکر خواهند شد :

TVGN_CARET : آیتم انتخاب شده جاری را دریافت خواهد کرد.

TVGN_CHILD : اولین آیتم فرزند آیتم مشخص شده توسط hitem را باز می گرداند.

TVGN_DROPHILITE : آیتمی را دریافت خواهد کرد که مقصد عملیات drag & drop است.

TVGN_FIRSTVISIBLE : اولین آیتم نمایان را دریافت خواهد کرد.

TVGN_NEXT : آیتم بعدی همخانواده را بر می گرداند.

TVGN_NEXTVISIBLE : آیتم بعدی نمایان را بر می گرداند. جهت مشخص کردن نمایان بودن یا نبودن یک آیتم می‌توان از پیغام TVM_GETITEMRECT استفاده کرد.

TVGN_PARENT : والد آیتم مشخص شده را باز می گرداند.

TVGN_PREVIOUS : آیتم قبلی همخانواده را دریافت می کند.

TVGN_PREVIOUSVISIBLE : آیتم قبلی نمایان را بر می گرداند.

TVGN_ROOT : بالاترین عضو کنترل را بر می گرداند

عملیات drag & drop در کنترل tree view :

برای پیاده سازی عملیات drag & drop در یک کنترل tree view ، به شرح زیر عمل می‌شود:

۱. هنگامیکه کاربری سعی می کند تا آیتمی را drag کند ، کنترل پیغام **TVN_BEGINDRAG** را به پنجره والد ارسال می کند.از این قابلیت می توان برای نمایش تصویری جهت نمایش حالت drag شدن یک آیتم استفاده کرد. می توان با ارسال پیغام **TVM_CREATEDRAGIMAGE** به کنترل ، تصویر پیش فرضی را برای حالت drag شدن یک آیتم از تصویر آیتم انتخابی ایجاد نماید. در این حالت کنترل tree view یک image list را تنها با یک تصویر drag شدن یک آیتم ایجاد کرده و دستگیره آنرا برمی گرداند.

www.mohandesyar.com

۲. پس از ایجاد تصویر drag ، می‌توان hotspot آنرا با فراخوانی تابع **ImageList_BeginDrag** ایجاد نمود. تعریف این تابع به صورت زیر است:

```
ImageList_BeginDrag PROTO himlTrack:DWORD, \
                          iTrack:DWORD , \
                          dxHotspot:DWORD, \
                          dyHotspot:DWORD
```

himlTrack : دستگیره ای است به image list ایی که حاوی تصویر drag است.

iTrack : اندیس تصویر drag در image list مربوطه

dxHotspot : از آنجائیکه در هنگام drag ، تصویر مربوطه بعنوان اشاره گر ماوس عمل خواهد کرد ، نیاز است تا hotspot آن مشخص گردد. این پارامتر ، مکان نسبی افقی hotspot را مشخص می کند.

dyHotspot : مکان نسبی عمودی hotspot را مشخص می کند.

بطور معمول اگر بخواهیم کنترل tree view ، تصویر مربوط به drag یک آیتم را برای ما ایجاد کند، باید پارامتر iTrack را مساوی صفر قرار داد. همچنین با مساوی صفر قرار دادن dxHotSpot و dyHotSpot ، می‌توان hotspot تصویر را گوشه بالا ، سمت چپ آن تعیین نمود.

۳. هنگامیکه تصویر drag برای نمایش آماده شد، تابع **ImageList_DragEnter** را برای ترسیم تصویر drag شده در پنجره برنامه ، فراخوانی می‌نمائیم. تعریف این تابع به صورت زیر است:

```
ImageList_DragEnter PROTO hwndLock:DWORD, x:DWORD, y:DWORD
```

hwndLock : دستگیره پنجره ای است که مالک تصویر می‌باشد. تصویر drag شده را نمی‌توان به خارج از ناحیه پنجره انتقال داد.

x and y : مختصات نمایش اولیه تصویر عملیات drag هستند. این مختصات بر اساس گوشه سمت چپ بالای پنجره مشخص می گردد.

۴. پس از نمایش تصویر آیتم drag شده ، باید انجام عملیات مربوطه را پشتیبانی و تکمیل کرد. مسیر drag را توسط پیغام **WM_MOUSEMOVE** و مکان drop را توسط پیغام **WM_LBUTTONUP** می‌توان کنترل کرد. اگر تصویر drag بر روی پنجره فرزند قرار داشته باشد ، پنجره والد هیچگاه پیغامی را از اعمال ماوس ، دریافت نخواهد کرد.بنابراین برای دریافت پیغامهای ماوس باید از تابع

www.mohandesyar.com

SetCapture استفاده کرد. در این حالت تمامی پیغامهای ماوس صرفنظر از موقعیت آن به پنجره مشخص شده ارسال خواهد شد.

۵. با پردازش پیغام **WM_MOUSEMOVE** ، مسیر drag را با فراخوانی ImageList_DragMove می‌توان به روز درآورد.این تابع تصویر drag را در حین عملیات مربوطه انتقال خواهد داد. جهت متمایز ساختن یک آیتم هنگامیکه تصویر drag کردن بر روی آن قرار گرفته است ، می توان با ارسال پیغام **TVM_HITTEST** متوجه این امر شد. اگر تصویر بر روی آیتم قرار داشت ، با فرستادن پیغام **TVM_SELECTITEM** به همراه پرچم **TVGN_DROPHILITE** ، می توان آنرا متمایز ساخت. لازم به ذکر است که پیش از ارسال پیغام **TVM_SELECTITEM** ، باید تصویر drag کردن را مخفی ساخت، در غیراینصورت رد جابجایی تصویر بجا خواهد ماند. عملیات مخفی سازی تصویر drag نمودن با فراخوانی ImageList_DragShowNolock انجام می شود. فراخوانی مجدد آن، سبب نمایش تصویر می گردد.

۶. با رها کردن دکمه سمت چپ ماوس ، باید چندین کار را انجام داد. اگر آیتمی را متمایز ساخته اید باید آنرا با ارسال پیغام **TVM_SELECTITEM** به همراه پرچم **TVGN_DROPHILITE** غیرمتمایز ساخت. در این حالت IParam باید صفر باشد. در غیراینصورت اثرات نامطلوبی بر روی صفحه نمایش داده خواهد شد. سپس باید **ImageList_DragLeave** را به همراه **ImageList_EndDrag** فراخوانی نمود. رها سازی ماوس نیز با فراخوانی تابع **ReleaseCapture** انجام می‌پذیرد. اگر در اینجا image list ایی را ایجاد کرده اید باید با فراخوانی **ImageList_Destroy** آنرا تخریب نمایید.

کد مثال ۱۶:

```
.386
.model flat,stdcall
option casemap:none
include \masm32\include\windows.inc
include \masm32\include\user32.inc
include \masm32\include\kernel32.inc
include \masm32\include\comctl32.inc
include \masm32\include\gdi32.inc
includelib \masm32\lib\gdi32.lib
includelib \masm32\lib\comctl32.lib
includelib \masm32\lib\user32.lib
includelib \masm32\lib\kernel32.lib

WinMain PROTO :DWORD,:DWORD,:DWORD,:DWORD
.const
IDB_TREE equ 4006               ; ID of the bitmap resource
.data
ClassName  db "TreeViewWinClass",0
AppName    db "Tree View Demo",0
TreeViewClass  db "SysTreeView32",0
```


www.mohandesyar.com

```
Parent  db "Parent Item",0
Child1  db "child1",0
Child2  db "child2",0
DragMode  dd FALSE              ; a flag to determine if we are in drag mode

.data?
hInstance  HINSTANCE ?
hwndTreeView dd ?           ; handle of the tree view control
hParent  dd ?                 ; handle of the root tree view item
hImageList dd ?               ; handle of the image list used in the tree view control
hDragImageList  dd ?       ; handle of the image list used to store the drag image

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

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_APPWORKSPACE
   mov   wc.lpszMenuName,NULL
   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_OVERLAPPED+WS_CAPTION+WS_SYSMENU+WS_MINIMIZEBOX+WS_MAXIMIZEBOX+WS_VISI
BLE,CW_USEDEFAULT,\
         CW_USEDEFAULT,200,400,NULL,NULL,\
         hInst,NULL
   mov   hwnd,eax
   .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 uses edi hWnd:HWND, uMsg:UINT, wParam:WPARAM, lParam:LPARAM
   LOCAL tvinsert:TV_INSERTSTRUCT
   LOCAL hBitmap:DWORD
   LOCAL tvhit:TV_HITTESTINFO
```


www.mohandesyar.com

```
.if uMsg==WM_CREATE
   invoke CreateWindowEx,NULL,ADDR TreeViewClass,NULL,\
      WS_CHILD+WS_VISIBLE+TVS_HASLINES+TVS_HASBUTTONS+TVS_LINESATROOT,0,\
      0,200,400,hWnd,NULL,\
      hInstance,NULL          ; Create the tree view control
   mov hwndTreeView,eax
   invoke ImageList_Create,16,16,ILC_COLOR16,2,10    ; create the associated image list
   mov hImageList,eax
   invoke LoadBitmap,hInstance,IDB_TREE       ; load the bitmap from the resource
   mov hBitmap,eax
   invoke ImageList_Add,hImageList,hBitmap,NULL    ; Add the bitmap into the image list
   invoke DeleteObject,hBitmap    ; always delete the bitmap resource
   invoke SendMessage,hwndTreeView,TVM_SETIMAGELIST,0,hImageList
   mov tvinsert.hParent,NULL
   mov tvinsert.hInsertAfter,TVI_ROOT
   mov tvinsert.item.imask,TVIF_TEXT+TVIF_IMAGE+TVIF_SELECTEDIMAGE
   mov tvinsert.item.pszText,offset Parent
   mov tvinsert.item.iImage,0
   mov tvinsert.item.iSelectedImage,1
   invoke SendMessage,hwndTreeView,TVM_INSERTITEM,0,addr tvinsert
   mov hParent,eax
   mov tvinsert.hParent,eax
   mov tvinsert.hInsertAfter,TVI_LAST
   mov tvinsert.item.pszText,offset Child1
   invoke SendMessage,hwndTreeView,TVM_INSERTITEM,0,addr tvinsert
   mov tvinsert.item.pszText,offset Child2
   invoke SendMessage,hwndTreeView,TVM_INSERTITEM,0,addr tvinsert
.elseif uMsg==WM_MOUSEMOVE
   .if DragMode==TRUE
      mov eax,lParam
      and eax,0ffffh
      mov ecx,lParam
      shr ecx,16
      mov tvhit.pt.x,eax
      mov tvhit.pt.y,ecx
      invoke ImageList_DragMove,eax,ecx
      invoke ImageList_DragShowNolock,FALSE
      invoke SendMessage,hwndTreeView,TVM_HITTEST,NULL,addr tvhit
      .if eax!=NULL
         invoke SendMessage,hwndTreeView,TVM_SELECTITEM,TVGN_DROPHILITE,eax
      .endif
      invoke ImageList_DragShowNolock,TRUE
   .endif
.elseif uMsg==WM_LBUTTONUP
   .if DragMode==TRUE
      invoke ImageList_DragLeave,hwndTreeView
      invoke ImageList_EndDrag
      invoke ImageList_Destroy,hDragImageList
      invoke SendMessage,hwndTreeView,TVM_GETNEXTITEM,TVGN_DROPHILITE,0
      invoke SendMessage,hwndTreeView,TVM_SELECTITEM,TVGN_CARET,eax
      invoke SendMessage,hwndTreeView,TVM_SELECTITEM,TVGN_DROPHILITE,0
      invoke ReleaseCapture
      mov DragMode,FALSE
   .endif
.elseif uMsg==WM_NOTIFY
   mov edi,lParam
   assume edi:ptr NM_TREEVIEW
   .if [edi].hdr.code==TVN_BEGINDRAG
      invoke SendMessage,hwndTreeView,TVM_CREATEDRAGIMAGE,0,[edi].itemNew.hItem
      mov hDragImageList,eax
      invoke ImageList_BeginDrag,hDragImageList,0,0,0
```


www.mohandesyar.com

```
            invoke ImageList_DragEnter,hwndTreeView,[edi].ptDrag.x,[edi].ptDrag.y
            invoke SetCapture,hWnd
            mov DragMode,TRUE
        .endif
        assume edi:nothing
    .elseif uMsg==WM_DESTROY
        invoke PostQuitMessage,NULL
    .else
        invoke DefWindowProc,hWnd,uMsg,wParam,lParam
        ret
    .endif
    xor eax,eax
    ret
WndProc endp
end start
```

بررسی کد فوق:

با پردازش پیغام WM_CREATE ، کنترل tree view را ایجاد خواهیم کرد:

```
invoke CreateWindowEx,NULL,ADDR TreeViewClass,NULL,\
        WS_CHILD+WS_VISIBLE+TVS_HASLINES+TVS_HASBUTTONS+TVS_LINESATROOT,0,\
        0,200,400,hWnd,NULL,\
        hInstance,NULL
```

در ادامه :

```
    invoke ImageList_Create,16,16,ILC_COLOR16,2,10
    mov hImageList,eax
    invoke LoadBitmap,hInstance,IDB_TREE
    mov hBitmap,eax
    invoke ImageList_Add,hImageList,hBitmap,NULL
    invoke DeleteObject,hBitmap
    invoke SendMessage,hwndTreeView,TVM_SETIMAGELIST,0,hImageList
```

در اینجا image list ایی که تصاویری را با اندازه 16x16 pixels ، با عمق رنگ ۱۶ بیت می‌پذیرید، ایجاد خواهیم کرد.این image list در ابتدا حاوی ۲ تصویر خواهد بود اما تا ۱۰ تصویر قابلیت انبساط است.

سپس تصاویر از فایل ریسورس بارگذاری شده و به image list اضافه می‌شود. در ادامه چون دستگیره بیت مپ را نیاز نخواهیم داشت، آنرا حذف می‌نمائیم. پس از ایجاد image list ، با فرستادن پیغام TVM_SETIMAGELIST به tree view کنترل ، آنرا به کنترل منتسب خواهیم کرد. در ادامه :

```
    mov tvinsert.hParent,NULL
    mov tvinsert.hInsertAfter,TVI_ROOT
    mov tvinsert.u.item.imask,TVIF_TEXT+TVIF_IMAGE+TVIF_SELECTEDIMAGE
    mov tvinsert.u.item.pszText,offset Parent
    mov tvinsert.u.item.iImage,0
    mov tvinsert.u.item.iSelectedImage,1
    invoke SendMessage,hwndTreeView,TVM_INSERTITEM,0,addr tvinsert
```


www.mohandesyar.com

با اضافه کردن آیتم ریشه به کنترل، شروع خواهیم کرد. بنابراین در این حالت hParent مساوی نال بوده و hInsertAfter معادل TVI_ROOT خواهد بود. توسط مقدار imask مشخص می گردد که اعضای pszText, iImage and iSelectedImage ساختار TV_ITEM معتبر بوده و با مقادیر مناسبی پر خواهند شد. pszText ، عنوان برچسب آیتم ریشه است. iImage ، اندیس تصویر حالت انتخاب نشده‌ی آیتم ریشه در image list است. iSelectedImage ، اندیس تصویر حالت انتخاب شده‌ی آیتم ریشه در image list است. پس از مقدار دهی های ذکر شده ، پیغام TVM_INSERTITEM به tree view کنترل، جهت اضافه کردن آیتم ریشه ارسال خواهد شد. سپس :

```
mov hParent,eax
mov tvinsert.hParent,eax
mov tvinsert.hInsertAfter,TVI_LAST
mov tvinsert.u.item.pszText,offset Child1
invoke SendMessage,hwndTreeView,TVM_INSERTITEM,0,addr tvinsert
mov tvinsert.u.item.pszText,offset Child2
invoke SendMessage,hwndTreeView,TVM_INSERTITEM,0,addr tvinsert
```

پس از اضافه شدن آیتم ریشه ، نوبت به اضافه کردن آیتم‌های فرزندان این والد فرا می‌رسد. در این حالت، hParent با دستگیره آیتم والد مقدار دهی خواهد شد. از آنجائیکه برای این آیتم‌ها نیز از تصاویر یکسانی همانند آیتم ریشه استفاده می گردد ، مقادیر iImage and iSelectedImage را تغییر نخواهیم داد. در ادامه :

```
.elseif uMsg==WM_NOTIFY
    mov edi,lParam
    assume edi:ptr NM_TREEVIEW
    .if [edi].hdr.code==TVN_BEGINDRAG
        invoke SendMessage,hwndTreeView,TVM_CREATEDRAGIMAGE,0,[edi].itemNew.hItem
        mov hDragImageList,eax
        invoke ImageList_BeginDrag,hDragImageList,0,0,0
        invoke ImageList_DragEnter,hwndTreeView,[edi].ptDrag.x,[edi].ptDrag.y
        invoke SetCapture,hWnd
        mov DragMode,TRUE
    .endif
    assume edi:nothing
```

اکنون اگر کاربر سعی کند تا آیتمی را drag کند ، tree view کنترل ، پیغام WM_NOTIFY را به همراه TVN_BEGINDRAG بعنوان code ارسال می‌نماید.در اینجا lParam اشاره‌گری به ساختار NM_TREEVIEW می‌باشد. سپس مقدار آنرا در edi قرار داده و از edi بعنوان اشاره‌گری به ساختار NM_TREEVIEW استفاده خواهیم کرد. دستور assume edi:ptr NM_TREEVIEW روشی است برای اینکه MASM از edi بعنوان اشاره گری به ساختار NM_TREEVIEW استفاده نماید.

سپس با ارسال پیغام TVM_CREATEDRAGIMAGE به tree view کنترل ، تصویر drag ایجاد خواهد شد. پس از ایجاد موفقیت آمیز آن ، دستگیره ای به این تصویر بازگشت داده خواهد شد. برای تعیین hotspot در تصویر drag ، تابع

www.mohandesyar.com

ImageList_BeginDrag را فراخوانی خواهیم کرد. سپس عملیات drag را با فراخوانی ImageList_DragEnter شروع خواهیم کرد. این تابع تصویر drag را در مکانی مشخص در پنجره ای معلوم ترسیم می‌نماید. از ساختار ptDrag که عضوی از ساختار NM_TREEVIEW می‌باشد، جهت تعیین محل نمایش اولیه تصویر drag ، استفاده می‌شود. سپس ورودی‌های ماوس را ضبط نموده و پرچمی را جهت مشخص نمودن ورود به حالت drag تنظیم می‌نمائیم. در ادامه:

```
.elseif uMsg==WM_MOUSEMOVE
    .if DragMode==TRUE
        mov eax,lParam
        and eax,0ffffh
        mov ecx,lParam
        shr ecx,16
        mov tvhit.pt.x,eax
        mov tvhit.pt.y,ecx
        invoke ImageList_DragMove,eax,ecx
        invoke ImageList_DragShowNolock,FALSE
        invoke SendMessage,hwndTreeView,TVM_HITTEST,NULL,addr tvhit
        .if eax!=NULL
            invoke SendMessage,hwndTreeView,TVM_SELECTITEM,TVGN_DROPHILITE,eax
        .endif
        invoke ImageList_DragShowNolock,TRUE
    .endif
```

سپس بر روی WM_MOUSEMOVE متمرکز خواهیم شد. هنگامیکه کاربر تصویر drag را drag می نماید ، پنجره والد، پیغام WM_MOUSEMOVE را دریافت خواهد نمود. در جهت پردازش این پیغام از تابع ImageList_DragMove جهت حرکت تصویر drag استفاده می کنیم. سپس بررسی خواهیم کرد که آیا تصویر drag بر روی آیتمی قرار دارد یا خیر؟ اینکار را با فرستادن پیغام TVM_HITTEST به همراه نقطه ای برای بررسی به کنترل tree view ، انجام خواهیم داد. اگر تصویر drag بر روی آیتمی قرار داشته باشد ، آیتم را با فرستادن پیغام TVM_SELECTITEM با پرچم TVGN_DROPHILITE ، متمایز خواهیم ساخت. در حین نمایش یک آیتم به صورت متمایز ، تصویر drag را مخفی خواهیم ساخت تا اثرات نامطلوبی را از خود بجا نگذارد. سپس :

```
.elseif uMsg==WM_LBUTTONUP
    .if DragMode==TRUE
        invoke ImageList_DragLeave,hwndTreeView
        invoke ImageList_EndDrag
        invoke ImageList_Destroy,hDragImageList
        invoke SendMessage,hwndTreeView,TVM_GETNEXTITEM,TVGN_DROPHILITE,0
        invoke SendMessage,hwndTreeView,TVM_SELECTITEM,TVGN_CARET,eax
        invoke SendMessage,hwndTreeView,TVM_SELECTITEM,TVGN_DROPHILITE,0
        invoke ReleaseCapture
        mov DragMode,FALSE
    .endif
```


www.mohandesyar.com

هنگامیکه کاربر دکمه سمت چپ ماوس را رها سازد ، عملیات drag باید خاتمه یابد. اینکار با فراخوانی تابع ImageList_DragLeave به همراه ImageList_EndDrag and ImageList_Destroy صورت می‌پذیرد.

همچنین جهت نمایش ظاهری بهتر tree view کنترل ، آخرین آیتم متمایز شده را بررسی و انتخاب خواهیم کرد. همچنین باید آنرا از حالت متمایز نیز خارج نمائیم تا سایر آیتم‌ها را نیز بتوان متمایز و انتخاب نمود. در آخر حالت ضبط ماوس را رها خواهیم کرد.

www.mohandesyar.com

مثال ۱۷– Window Subclassing

در اکثر مواقع ویندوز خواص تقریبا لازمی را برای کار با پنجره ها در اختیار برنامه نویس قرار می‌دهد. اما این خواص گاهی از اوقات تمام نیاز یک برنامه نویس را فراهم نمی‌کنند. برای مثال فرض کنید که نیاز به تکست باکسی دارید که فقط حروف هگز را بپذیرد. برای حل این مساله به صورتی مناسب ، window subclassing ابداع شده است.

در حالت معمول، کنترلی بر روی یک تکست باکس استاندارد نمی توان داشت و تنها عکس العملی را که می توان نشان داد ، رد کردن کل رشته وارد شده توسط کاربر است. روشی حرفه ای تر برای فیلتر کردن حروف ناخواسته ، کنترل ورودی کاربر درست در لحظه ای می‌باشد که او چیزی را تایپ کرده است.

هنگامیکه کاربر حرفی را در یک تکست باکس تایپ می‌کند ، ویندوز پیغام WM_CHAR را به روال پنجره تکست باکس ارسال می‌نماید. این روال متعلق به ویندوز بوده و قابل تغییر نیست. اما می‌توان پیش از رسیدن پیغام به این روال استاندارد ، آنرا به روال پنجره تعریف شده توسط خود ، هدایت نمائیم. در این حالت می‌توان ورودی کاربر را پردازش کرد و یا در صورت عدم تمایل به پردازش آن، می توان آنرا به روال پنجره استاندارد انتقال داد. در این حالت روال پنجره ما ، خود را بین ویندوز و کنترل تکست باکس قرار داده است.

پیش از subclassing

ویندوز ⟸ روال پنجره کنترل تکست باکس

پس از subclassing

ویندوز ⟸ روال پنجره ما ⟸ روال پنجره کنترل تکست باکس

در ادامه روش انجام window subclassing را بررسی خواهیم کرد. لازم به ذکر است که این روش منحصر به کنترل‌ها نبوده و برای هر نوع پنجره ای قابل اجرا است.

ویندوز با کمک عضو lpfnWndProc ساختار WNDCLASSEX ، محل قرار گیری روال پنجره کنترل تکست باکس را می‌داند. اگر این عضو به آدرس قرارگیری روال پنجره تعریف شده توسط ما در حافظه اشاره کند ، ویندوز پیغامهای لازم را به این روال ارسال خواهد کرد. اینکار را می‌توان با فراخوانی تابع SetWindowLong انجام داد. تعریف این روال به صورت زیر است:

```
SetWindowLong  PROTO  hWnd:DWORD, nIndex:DWORD, dwNewLong:DWORD
```


www.mohandesyar.com

hWnd : دستگیره پنجره ای است که مقداری در WNDCLASSEX آنرا می‌خواهیم تغییر دهیم.

nIndex : مقداری است برای تغییر و یکی از موارد زیر می تواند باشد:

GWL_EXSTYLE : سبک جدید توسعه یافته پنجره را مشخص می کند.

GWL_STYLE : سبک پنجره را مشخص می‌نماید.

GWL_WNDPROC : آدرس جدید رویه پنجره را مشخص می‌نماید.

GWL_HINSTANCE : دستگیره وهله برنامه را تنظیم می کند.

GWL_ID : ID جدید پنجره را تنظیم می‌نماید.

GWL_USERDATA : مقدار ۳۲ بیتی وابسته به پنجره را مشخص می کند.هر پنجره دارای مقدار ۳۲ بیتی متناظری است و توسط برنامه ای که آنرا ایجاد کرده است، استفاده می‌شود.

dwNewLong : مقدار جایگزینی

بنابراین رویه پنجره خود را که پیغام‌های رسیده از تکست باکس را پردازش می کند ایجاد کرده و سپس تابع SetWindowLong را با پرچم GWL_WNDPROC ، به همراه آدرس رویه پنجره خود، بعنوان سومین پارامتر ، فراخوانی می کنیم. اگر فراخوانی تابع موفقیت آمیز باشد ، مقدار بازگشتی ، مقدار ۳۲ بیتی مشخص شده یعنی آدرس رویه اصلی پنجره ، خواهد بود. برای استفاده از این مقدار در رویه پنجره خود ، نیاز است تا آنرا ذخیره نمائیم. بدیهی است که قصد پردازش تعدادی از پیغام های رسیده را نخواهیم داشت. آنها را با فراخوانی CallWindowProc به رویه اصلی پنجره انتقال خواهیم داد. تعریف این تابع به شکل زیر است:

```
CallWindowProc PROTO lpPrevWndFunc:DWORD, \
                     hWnd:DWORD,\
                     Msg:DWORD,\
                     wParam:DWORD,\
                     lParam:DWORD
```

در این تابع پارامتر lpPrevWndFunc ، آدرس رویه پنجره اصلی است. سایر پارامترهای آن از رویه پنجره ما دریافت خواهند شد.

www.mohandesyar.com

کد مثال ۱۷ :

```
.386
.model flat,stdcall
option casemap:none
include \masm32\include\windows.inc
include \masm32\include\user32.inc
include \masm32\include\kernel32.inc
include \masm32\include\comctl32.inc
includelib \masm32\lib\comctl32.lib
includelib \masm32\lib\user32.lib
includelib \masm32\lib\kernel32.lib

WinMain PROTO :DWORD,:DWORD,:DWORD,:DWORD
EditWndProc PROTO :DWORD,:DWORD,:DWORD,:DWORD

.data
ClassName  db "SubclassWinClass",0
AppName    db "Subclassing Demo",0
EditClass  db "EDIT",0
Message  db "You pressed Enter in the text box!",0

.data?
hInstance  HINSTANCE ?
hwndEdit dd ?
OldWndProc dd ?

.code
start:
   invoke GetModuleHandle, NULL
   mov    hInstance,eax
   invoke WinMain, hInstance,NULL,NULL, 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_APPWORKSPACE
   mov   wc.lpszMenuName,NULL
   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_OVERLAPPED+WS_CAPTION+WS_SYSMENU+WS_MINIMIZEBOX+WS_MAXIMIZEBOX+WS_VISI
BLE,CW_USEDEFAULT,\
         CW_USEDEFAULT,350,200,NULL,NULL,\
         hInst,NULL
   mov   hwnd,eax
```


www.mohandesyar.com

```
    .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_CREATE
        invoke CreateWindowEx,WS_EX_CLIENTEDGE,ADDR EditClass,NULL,\
            WS_CHILD+WS_VISIBLE+WS_BORDER,20,\
            20,300,25,hWnd,NULL,\
            hInstance,NULL
        mov hwndEdit,eax
        invoke SetFocus,eax
        ;--------------------------------------
        ; Subclass it!
        ;--------------------------------------
        invoke SetWindowLong,hwndEdit,GWL_WNDPROC,addr EditWndProc
        mov OldWndProc,eax
    .elseif uMsg==WM_DESTROY
        invoke PostQuitMessage,NULL
    .else
        invoke DefWindowProc,hWnd,uMsg,wParam,lParam
        ret
    .endif
    xor eax,eax
    ret
WndProc endp

EditWndProc PROC hEdit:DWORD,uMsg:DWORD,wParam:DWORD,lParam:DWORD
    .if uMsg==WM_CHAR
        mov eax,wParam
        .if (al>="0" && al<="9") || (al>="A" && al<="F") || (al>="a" && al<="f") || al==VK_BACK
            .if al>="a" && al<="f"
                sub al,20h
            .endif
            invoke CallWindowProc,OldWndProc,hEdit,uMsg,eax,lParam
            ret
        .endif
    .elseif uMsg==WM_KEYDOWN
        mov eax,wParam
        .if al==VK_RETURN
            invoke MessageBox,hEdit,addr Message,addr AppName,MB_OK+MB_ICONINFORMATION
            invoke SetFocus,hEdit
        .else
            invoke CallWindowProc,OldWndProc,hEdit,uMsg,wParam,lParam
            ret
        .endif
    .else
        invoke CallWindowProc,OldWndProc,hEdit,uMsg,wParam,lParam
        ret
    .endif
    xor eax,eax
    ret
EditWndProc endp
end start
```


www.mohandesyar.com

بررسی کد فوق:

```
invoke SetWindowLong,hwndEdit,GWL_WNDPROC,addr EditWndProc
mov OldWndProc,eax
```

پس از ایجاد کنترل تکست باکس ، با فراخوانی تابع SetWindowLong ، آنرا subclass خواهیم کرد. با اینکار آدرس رویه اصلی پنجره را با آدرس رویه پنجره خود جایگزین خواهیم کرد. همچنین در ادامه آدرس رویه پنجره اصلی را نیز ذخیره کرده ایم. از آن در ادامه در تابع CallWindowProc استفاده خواهیم کرد. لازم به ذکر است که رویه EditWndProc ، مطابق قالب یک رویه متداول پنجره می‌باشد. سپس :

```
.if uMsg==WM_CHAR
    mov eax,wParam
    .if (al>="0" && al<="9") || (al>="A" && al<="F") || (al>="a" && al<="f") || al==VK_BACK
       .if al>="a" && al<="f"
          sub al,20h
       .endif
       invoke CallWindowProc,OldWndProc,hEdit,uMsg,eax,lParam
       ret
    .endif
```

درون رویه EditWndProc ، پیغامهای WM_CHAR را پردازش می کنیم. اگر حرف دریافتی بین a-f or 0-9 باشد،آنرا پذیرفته و به رویه اصلی پنجره انتقال خواهیم داد. اگر حرف وارد شده با حروف کوچک باشد ، با افزودن 20h به آن، آنرا به حروف بزرگ تبدیل خواهیم کرد. اگر حرف، مورد قبول نباشد صرفا آنرا رد کرده و به رویه اصلی پنجره انتقال نخواهیم داد. بنابراین اینگونه حروف نمایش داده نخواهند شد.

```
.elseif uMsg==WM_KEYDOWN
    mov eax,wParam
    .if al==VK_RETURN
       invoke MessageBox,hEdit,addr Message,addr AppName,MB_OK+MB_ICONINFORMATION
       invoke SetFocus,hEdit
    .else
       invoke CallWindowProc,OldWndProc,hEdit,uMsg,wParam,lParam
       ret
    .end
```

همچنین برای نمایش سایر توانایی‌های subclassing ، پیغامهای WM_KEYDOWN را نیز جهت عکس العمل نشان دادن به فشرده شدن کلید enter ، پردازش کرده ایم.

www.mohandesyar.com

مثال ۱۸ – Pipe

Pipe مجرای ارتباطی دو طرفه است. از pipe برای تبادل اطلاعات بین دو پروسه و یا درون یک پروسه می‌توان استفاده کرد. دو نوع pipe وجود دارند : anonymous and named pipes . از Anonymous pipe ، بدون دانستن نام آن می‌توان استفاده کرد.named pipe ، برخلاف آن بوده و برای استفاده از آن دانستن نام آن الزامی است.

همچنین pipes را بر اساس خواص آنها که شامل یک طرفه و دو طرفه می‌شود ، می‌توان طبقه بندی کرد. anonymous pipe همواره یک طرفه بوده و named pipe (مجرای نامدار)، دو طرفه و یا یک طرفه می‌باشد. از named pipe عموما در محیط شبکه برای اتصال سرور به چندین کلاینت استفاده می گردد.

در این مثال قصد بررسی یک anonymous pipe (مجرای بی‌نام) را داریم. هدف اصلی از ایجاد مجرای بی نام ، ایجاد پلی ارتباطی بین پروسه والد و پروسه‌های فرزند و یا بین پروسه‌های فرزند است.

مجرای بی‌نام، هنگام کار با برنامه‌های حالت console بسیار مفید است. این نوع برنامه‌ها، در پنجره‌ای شبیه به داس در ویندوز اجرا شده و ۳۲ بیتی هستند.

یک برنامه console از سه handle برای دریافت و ارائه اطلاعات استفاده می کند. به آنها دستگیره‌های استاندارد نیز گفته می‌شود (standard handles) . این سه دستگیره عبارتند از :

standard input, standard output and standard error

(اگر برنامه نویس C باشید با این اصطلاحات به خوبی آشنایی خواهید داشت)

یک برنامه console با فراخوانی تابع GetStdHandle ، این دستگیره‌ها را می‌تواند بدست آورد. برنامه‌های معمول دارای GUI بوده، console نداشته و امکان فراخوانی تابع فوق را ندارند. اگر واقعا نیاز به console باشد با فراخوانی تابع AllocConsole می‌توان یک console را به برنامه اختصاص داد و پس از اتمام کار، با فراخوانی FreeConsole منابع تخصیص داده شده را به سیستم عامل بازگشت داد.

مجرای بی‌نام غالبا در برنامه‌های console برای تغییر جهت ورودی و یا خروجی برنامه‌های console فرزند بکار می‌رود. برنامه والد می‌تواند یک برنامه حالت GUI و یا console بوده اما برنامه فرزند باید حتما حالت console باشد. برای این تغییر جهت می‌توان یک سر دستگیره استاندارد را به یک سر مجرا اتصال داد. در این حالت برنامه کنسول اطلاعی در این مورد پیدا نخواهد کرد و از آن بعنوان یک دستگیره استاندارد استفاده می‌نماید. این در حقیقت نوعی polymorphism در واژگان برنامه نویسی شیء گرا محسوب می‌شود.این روش از آنجائیکه نیاز به ایجاد تغییر در پروسه فرزند را نخواهد داشت ، روشی قدرتمند محسوب می گردد.

مطلب دیگری را که باید در مورد برنامه console دانست این است که برنامه این دستگیره‌های استاندارد را از کجا بدست خواهد آورد. زمانیکه یک برنامه console ایجاد می‌شود ، پروسه والد دو انتخاب را خواهد داشت: می تواند کنسول جدیدی را برای فرزند ایجاد نماید و یا از کنسول فعلی، نمونه ای را به ارث ببرد. برای حالت دوم ، برنامه والد یا

www.mohandesyar.com

باید خود نیز حالت کنسول باشد و یا اگر دارای GUI است باید تابع AllocConsole را جهت تخصیص یک کنسول ، فراخوانی نماید.

برای ایجاد یک مجرای بدون نام باید تابع CreatePipe را فراخوانی نمود. این تابع به شکل زیر تعریف می‌شود:

```
CreatePipe proto pReadHandle:DWORD, \
    pWriteHandle:DWORD,\
    pPipeAttributes:DWORD,\
    nBufferSize:DWORD
```

pReadHandle : اشاره‌گری است به متغیری از نوع dword که دستگیره‌ای به انتهای خواندنی مجرا را دریافت خواهد کرد.

pWriteHandle : اشاره‌گری است به متغیری از نوع dword که دستگیره‌ای به انتهای نوشتنی مجرا را دریافت خواهد کرد.

pPipeAttributes : اشاره‌گری است به ساختار SECURITY_ATTRIBUTES و مشخص می‌کند که دستگیره های خواندن و نوشتن توسط پروسه فرزند قابل به ارث رسیدن هستند.

nBufferSize : اندازه بافر پیشنهادی است که مجرا برای استفاده تخصیص خواهد داد. اگر مساوی Null قرار گیرد از اندازه پیش فرض استفاده خواهد شد.

اگر فراخوانی این تابع موفقیت آمیز باشد ، خروجی آن غیرصفر خواهد بود. در این حالت دو دستگیره به انتهای خواندنی و نوشتنی مجرا بدست خواهد آمد.

در ادامه مراحل مورد نیاز برای هدایت خروجی استاندارد برنامه فرزند حالت console به پروسه‌ی خود را بررسی خواهیم کرد:

۱. ایجاد مجرایی بی‌نام توسط فراخوانی تابع CreatePipe . دراینجا باید عضو bInheritable مربوط به SECURITY_ATTRIBUTES ، به TRUE تنظیم شود (زیرا دستگیره باید قابل ارث‌بری گردد).

۲. در ادامه باید پارامترهای تابع CreateProcess آماده شوند، زیرا از آنها برای ایجاد برنامه حالت کنسول استفاده خواهیم کرد.ساختار مهم در اینجا، STARTUPINFO می‌باشد. این ساختار خواص ظاهری پنجره اصلی پروسه فرزند را در اولین بار نمایش آن، تعیین می‌کند.این ساختار برای برنامه ما حیاتی می‌باشد.به این وسیله می‌توان پنجره اصلی را مخفی نمود و سپس دستگیره مجرا را به پروسه حالت کنسول فرزند ، انتقال داد. در ادامه اعضایی از آن که باید مقدار دهی شوند مشخص گردیده است:

cb : اندازه ساختار STARTUPINFO را مشخص می‌کند.

www.mohandesyar.com

dwFlags : پرچمی است که مشخص می کند کدامیک از اعضای ساختار معتبر هستند همچنین حالت مخفی و نمایان بودن پنجره اصلی را نیز مشخص می کند. برای مقصود ما باید ترکیبی از STARTF_USESHOWWINDOW و STARTF_USESTDHANDLES را بکار برد.

hStdOutput and hStdError : دستگیره‌هایی که قرار است پروسه فرزند از آنها بعنوان standard output/error استفاده نماید. برای برنامه این مثال، از دستگیره نوشتی مجرا بدین منظور استفاده خواهد شد. بنابراین در این مواقع اطلاعات از طریق مجرا به برنامه والد منتقل خواهد شد.

wShowWindow : مخفی و یا نمایان بودن پنجره اصلی را مشخص می کند.از آنجائیکه می‌خواهیم تنها برنامه فرزند نمایش یابد این عضو را با SW_HIDE مقدار دهی خواهیم کرد.

۳. فراخوانی تابع CreateProcess جهت بارگذاری برنامه فرزند. پس از فراخوانی موفقیت آمیز این تابع ، برنامه فرزند به حافظه منتقل شده اما نمایش داده نمی‌شود.
۴. بستن دستگیره نوشتنی مجرا. انجام اینکار الزامی است. زیرا پروسه والد نیازی به دستگیره نوشتنی مجرا نداشته و مجرا در صورت وجود داشتن بیشتر از یک انتها کار نخواهد کرد. بستن آن باید پیش از خواندن داده‌ها از مجرا صورت گیرد. لازم به ذکر است که اگر پیش از فراخوانی CreateProcess ، این دستگیره بسته شود ، مجرا شکسته خواهد شد.
۵. اکنون با فراخوانی ReadFile ، از انتهای خواندنی مجرا می‌توان داده‌ها را دریافت کرد. با فراخوانی این تابع ، پروسه فرزند به حالت اجرایی درخواهد آمد. پس از اجرا و هنگامیکه داده ها را به دستگیره استاندارد خروجی ارسال می کند (که در حقیقت دستگیره ای است به انتهای نوشتنی مجرا)، داده ها از طریق مجرا به انتهای خواندنی ارسال خواهند شد. فراخوانی ReadFile باید مرتبا تا زمانیکه خروجی تابع صفر شود ادامه یابد (یعنی تا زمانیکه دیگر داده‌ای برای خواندن وجود نداشته باشد). در این مثال داده های دریافتی در یک کنترل تکست باکس قرار گرفته اند.
۶. بستن دستگیره خواندنی مجرا.

www.mohandesyar.com

کد مثال ۱۸ :

```
.386
.model flat,stdcall
option casemap:none
include \masm32\include\windows.inc
include \masm32\include\user32.inc
include \masm32\include\kernel32.inc
include \masm32\include\gdi32.inc
includelib \masm32\lib\gdi32.lib
includelib \masm32\lib\user32.lib
includelib \masm32\lib\kernel32.lib

WinMain PROTO :DWORD,:DWORD,:DWORD,:DWORD

.const
IDR_MAINMENU equ 101        ; the ID of the main menu
IDM_ASSEMBLE equ 40001

.data
ClassName           db "PipeWinClass",0
AppName             db "One-way Pipe Example",0 EditClass db "EDIT",0
CreatePipeError     db "Error during pipe creation",0
CreateProcessError     db "Error during process creation",0
CommandLine     db "ml /c /coff /Cp test.asm",0

.data?
hInstance HINSTANCE ?
hwndEdit dd ?

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

WinMain proc hInst:DWORD,hPrevInst:DWORD,CmdLine:DWORD,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_APPWORKSPACE
    mov wc.lpszMenuName,IDR_MAINMENU
    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+WS_VISIBLE,CW_USEDEFAULT,\
```


www.mohandesyar.com

```
CW_USEDEFAULT,400,200,NULL,NULL,\ hInst,NULL
   mov hwnd,eax
   .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
   LOCAL rect:RECT
   LOCAL hRead:DWORD
   LOCAL hWrite:DWORD
   LOCAL startupinfo:STARTUPINFO
   LOCAL pinfo:PROCESS_INFORMATION
   LOCAL buffer[1024]:byte
   LOCAL bytesRead:DWORD
   LOCAL hdc:DWORD
   LOCAL sat:SECURITY_ATTRIBUTES
   .if uMsg==WM_CREATE
      invoke CreateWindowEx,NULL,addr EditClass, NULL,\
               WS_CHILD+ WS_VISIBLE+ ES_MULTILINE+ ES_AUTOHSCROLL+ ES_AUTOVSCROLL,\
 0, 0, 0, 0, hWnd, NULL, hInstance, NULL
      mov hwndEdit,eax
   .elseif uMsg==WM_CTLCOLOREDIT
      invoke SetTextColor,wParam,Yellow
      invoke SetBkColor,wParam,Black
     invoke GetStockObject,BLACK_BRUSH
      ret
   .elseif uMsg==WM_SIZE
      mov edx,lParam
      mov ecx,edx
      shr ecx,16
      and edx,0ffffh
      invoke MoveWindow,hwndEdit,0,0,edx,ecx,TRUE
   .elseif uMsg==WM_COMMAND
     .if lParam==0
        mov eax,wParam
        .if ax==IDM_ASSEMBLE
           mov sat.nLength,sizeof SECURITY_ATTRIBUTES
           mov sat.lpSecurityDescriptor,NULL
           mov sat.bInheritHandle,TRUE
           invoke CreatePipe,addr hRead,addr hWrite,addr sat,NULL
           .if eax==NULL
              invoke MessageBox, hWnd, addr CreatePipeError, addr AppName, MB_ICONERROR+ MB_OK

           .else
              mov startupinfo.cb,sizeof STARTUPINFO
              invoke GetStartupInfo,addr startupinfo
              mov eax, hWrite
              mov startupinfo.hStdOutput,eax
              mov startupinfo.hStdError,eax
              mov startupinfo.dwFlags, STARTF_USESHOWWINDOW+ STARTF_USESTDHANDLES
              mov startupinfo.wShowWindow,SW_HIDE
              invoke CreateProcess, NULL, addr CommandLine, NULL, NULL,\
```


www.mohandesyar.com

```
                                    TRUE, NULL, NULL, NULL, addr startupinfo, addr pinfo
            .if eax==NULL
               invoke MessageBox,hWnd,addr CreateProcessError,\
                           addr   AppName,MB_ICONERROR+MB_OK
            .else
               invoke CloseHandle,hWrite
               .while TRUE
                  invoke RtlZeroMemory,addr buffer,1024
                  invoke ReadFile,hRead,addr buffer,1023,addr bytesRead,NULL
                  .if eax==NULL
                     .break
                  .endif
                  invoke SendMessage,hwndEdit,EM_SETSEL,-1,0
                  invoke SendMessage,hwndEdit,EM_REPLACESEL,FALSE,addr buffer
               .endw
            .endif
            invoke CloseHandle,hRead
         .endif
      .endif
   .endif
.elseif uMsg==WM_DESTROY
   invoke PostQuitMessage,NULL
.else
   invoke DefWindowProc,hWnd,uMsg,wParam,lParam ret
.endif
xor eax,eax
ret
WndProc endp
end start
```

بررسی کد فوق:

مثال فوق ml.exe را در جهت اسمبل کردن test.asm ، فراخوانی کرده و خروجی ml.exe را به کنترل تکست باکس خود هدایت می کند.

هنگامیکه برنامه بارگذاری می شود ، کلاس پنجره را طبق معمول رجیستر می‌نماید. اولین کاری که برنامه انجام خواهد داد ایجاد کنترل تکست باکس جهت نمایش خروجی ml.exe است. در ادامه رنگ متن و زمینه ی تکست باکس را تغییر خواهیم داد. هنگامی که تکست باکس قصد رنگ آمیزی ناحیه کلاینت خود را داشته باشد ، پیغام WM_CTLCOLOREDIT را به والد خود ارسال خواهد کرد. wParam ، دستگیره ای به بافت ابزار بوده و کنترل تکست باکس از آن برای رنگ آمیزی ناحیه کلاینت خود استفاده می کند.در ادامه :

```
.elseif uMsg==WM_CTLCOLOREDIT
   invoke SetTextColor,wParam,Yellow
   invoke SetTextColor,wParam,Black
   invoke GetStockObject,BLACK_BRUSH
   ret
```


www.mohandesyar.com

تابع SetTextColor ، رنگ متن را به زرد تغییر می‌دهد. در پایان ، دستگیره برس مشکی رنگ اینجاد شده ، به ویندوز جهت رنگ آمیزی زمینه ، بازگشت داده می‌شود. سپس :

```
.if ax==IDM_ASSEMBLE
   mov sat.nLength,sizeof SECURITY_ATTRIBUTES
   mov sat.lpSecurityDescriptor,NULL
   mov sat.bInheritHandle,TRUE
```

هنگامیکه کاربر از منو، گزینه Assemble را انتخاب می کند، یک مجرای بی نام را ایجاد خواهد شد. قبل از فراخوانی تابع CreatePipe ، باید اعضای ساختار مقدار دهی شود. با مقدار دهی نال عضو lpSecurityDescriptor ، مقادیر پیش فرض انتخاب خواهند شد. همچنین عضو bInheritHandle نیز باید به TRUE تنظیم شود. در این حالت پروسه فرزند می‌تواند از دستگیره های مجرا استفاده نمایند.سپس :

```
invoke CreatePipe,addr hRead,addr hWrite,addr sat,NULL
```

در صورت فراخوانی موفقیت آمیز تابع فوق، پارامترهای hRead و hWrite با دستگیره‌های انتهای خواندنی و نوشتنی مجرا مقدار دهی خواهند شد. در ادامه :

```
mov startupinfo.cb,sizeof STARTUPINFO
invoke GetStartupInfo,addr startupinfo
mov eax, hWrite
mov startupinfo.hStdOutput,eax
mov startupinfo.hStdError,eax
mov startupinfo.dwFlags, STARTF_USESHOWWINDOW+ STARTF_USESTDHANDLES
mov startupinfo.wShowWindow,SW_HIDE
```

سپس باید اعضای ساختار STARTUPINFO مقدار دهی شود. با فراخوانی GetStartupInfo ، ساختار STARTUPINFO با مقادیر پیش فرض پروسه والد پر می‌شوند. مقدار دهی اعضای این ساختار الزامی است. پس از پایان کار فراخوانی تابع GetStartupInfo ، می‌توان مقادیر اعضایی را که مهم هستند، ویرایش نمود.

در ادامه دستگیره انتهای مجرا را به hStdOutput and hStdError کپی خواهیم کرد، زیرا تمایل داریم که پروسه فرزند از آن بجای دستگیره‌های استاندارد خروجی و خطا ، استفاده نماید. همچنین قصد داریم تا پنجره کنسول را مخفی نمائیم ، بنابراین عضو wShowWidow را با SW_HIDE ، مقدار دهی خواهیم کرد. در آخر باید مشخص نمائیم که اعضای hStdOutput, hStdError and wShowWindow ، معتبر بوده و باید مورد استفاده قرار گیرند. اینکار را با استفاده از پرچم های STARTF_USESHOWWINDOW and STARTF_USESTDHANDLES بعنوان مقدار dwFlags انجام خواهیم داد. سپس :

```
invoke CreateProcess, NULL, addr CommandLine, NULL, NULL,\

TRUE, NULL, NULL, NULL, addr startupinfo, addr pinfo
```


www.mohandesyar.com

در ادامه پروسه فرزند را ایجاد می‌نمائیم. لازم به ذکر است که جهت استفاده از دستگیره مجرا باید پارامتر bInheritHandles را به TRUE تنظیم کرد.سپس :

```
invoke CloseHandle,hWrite
```

پس از ایجاد موفقیت آمیز پروسه فرزند، باید دستگیره انتهای نوشتنی مجرا را بست. همچنین دستگیره نوشتنی از طریق STARTUPINFO به پروسه فرزند انتقال یافته است. اگر دستگیره نوشتنی بسته نشود ، دو دستگیره انتهای نوشتنی وجود خواهند داشت. در این حالت، مجرا کار نخواهد کرد. سپس:

```
.while TRUE
    invoke RtlZeroMemory,addr buffer,1024
    invoke ReadFile,hRead,addr buffer,1023,addr bytesRead,NULL
    .if eax==NULL
        .break
    .endif
    invoke SendMessage,hwndEdit,EM_SETSEL,-1,0
    invoke SendMessage,hwndEdit,EM_REPLACESEL,FALSE,addr buffer
.endw
```

اکنون آماده ایم تا داده‌ها را از خروجی استاندارد پروسه دریافت کنیم. در اینجا حلقه ای بی پایان تا زمانیکه تمام داده‌های مورد نظر دریافت شوند، ایجاد می گردد.

در ابتدا با فراخوانی RtlZeroMemory ، بافر را با صفر پر کرده ، سپس از ReadFile ، به همراه دستگیره خواندنی مجرا کمک می گیریم. لازم به ذکر است از آنجائیکه نیاز است تا رشته های مختوم به نال را به تکست باکس انتقال دهیم ، هر بار 1023 bytes از داده‌ها را خواهیم خواند.

اگر از تابع SetWindowText جهت انتقال داده ها به تکست باکس استفاده کنیم، هر بار داده‌های جدید بر روی داده‌های قبلی جای نویسی خواهند شد. اما علاقمند هستیم که هر بار داده‌ها ، به انتهای داده‌های موجود اضافه شوند. برای دستیابی به این مقصود، ابتدا هشتک (caret) را به انتهای متن، در کنترل تکست باکس انتقال می دهیم. اینکار با فرستادن پیغام EM_SETSEL به همراه wParam==-1 ، میسر است. سپس برای اضافه کردن داده‌ها در همان مکان از پیغام EM_REPLACESEL استفاده خواهیم کرد. سپس :

```
invoke CloseHandle,hRead
```

هنگامیکه تابع ReadFile ، نال برگرداند ، از حلقه خارج شده و دستگیره خواندن را خواهیم بست.

www.mohandesyar.com

مثال ۱۹– Superclassing

فرض کنید در برنامه به چندین تکست باکس که تقریبا رفتار یکسانی را ارائه می دهند نیاز دارید. برای مثال به ۱۰ تکست باکس که تنها اعداد را می‌پذیرند ، نیاز است. چندین روش برای پیاده سازی این مثال قابل تصور است:

- کلاسی مخصوص این نوع کنترل ایجاد کرده و سپس با نمونه سازی از آن کنترل های مورد نیاز را ایجاد کرد.
- تکست باکس را ایجاد کرده و سپس توسط subclassing خواص لازم را به آنها اعمال کرد.
- Supperclassing تکست باکس.

روش اول نسبتا ملال آور است. در این حالت باید تمام خواص تکست باکس را خودتان پیاده سازی نماید. انجام روش دوم، از روش اول ساده تر بوده و نیاز به کدنویسی کمتری دارد. اما در حالت تعداد کنترل‌های زیاد ، نیاز به کار قابل توجهی خواهد داشت. Supperclassing یکی از بهترین روش‌هایی است که می‌توان برای این حالت پیشنهاد داد.

با استفاده از supperclassing می توان کنترل یک کلاس پنجره ویژه را کاملا دراختیار گرفت. کنترل در اینجا به معنای اصلاح خواص کلاس پنجره مطابق میل و سپس ایجاد تعداد زیادی کنترل از روی آن می‌باشد. مراحل این عملیات به شرح زیر است:

- فراخوانی تابع GetClassInfoEx جهت بدست آوردن اطلاعاتی در مورد کلاس پنجره مورد نظر. این تابع نیاز به اشاره گری به ساختار WNDCLASSEX دارد. درصورت موفقیت آمیز بودن این فراخوانی ، اعضای این ساختار خواص مورد نظر را دریافت خواهند کرد.
- اصلاح مقادیر اعضای ساختار WNDCLASSEX مطابق میل. دو عضو را حتما باید اصلاح نمود:

  hInstance : دستگیره وهله برنامه را باید در این عضو قرار داد.

  lpszClassName : باید اشاره گری به نام کلاس جدید باشد.

  اگر نیاز به تغییر مقدار عضو lpfnWndProc بود، باید مقدار اولیه آنرا با فراخوانی CallWindowProc ذخیره نمود.
- رجیستر کردن کلاس WNDCLASSEX .
- ایجاد پنجره از کلاس جدید.

www.mohandesyar.com

کد برنامه مثال ۱۹ :

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

WM_SUPERCLASS equ WM_USER+5
WinMain PROTO :DWORD,:DWORD,:DWORD,:DWORD
EditWndProc PROTO :DWORD,:DWORD,:DWORD,:DWORD

.data
ClassName  db "SuperclassWinClass",0
AppName    db "Superclassing Demo",0
EditClass  db "EDIT",0
OurClass db "SUPEREDITCLASS",0
Message  db "You pressed the Enter key in the text box!",0

.data?
hInstance dd ?
hwndEdit dd 6 dup(?)
OldWndProc dd ?

.code
start:
   invoke GetModuleHandle, NULL
   mov    hInstance,eax
   invoke WinMain, hInstance,NULL,NULL, 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_APPWORKSPACE
   mov wc.lpszMenuName,NULL
   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+WS_EX_CONTROLPARENT,ADDR C      Name,ADDR
```


www.mohandesyar.com

```
AppName,\

WS_OVERLAPPED+WS_CAPTION+WS_SYSMENU+WS_MINIMIZEBOX+WS_MAXIMIZEBOX+WS_VISI
BLE,CW_USEDEFAULT,\
        CW_USEDEFAULT,350,220,NULL,NULL,\
        hInst,NULL
   mov hwnd,eax

   .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 uses ebx edi hWnd:HWND, uMsg:UINT, wParam:WPARAM, lParam:LPARAM
   LOCAL wc:WNDCLASSEX
   .if uMsg==WM_CREATE
      mov wc.cbSize,sizeof WNDCLASSEX
      invoke GetClassInfoEx,NULL,addr EditClass,addr wc
      push wc.lpfnWndProc
      pop OldWndProc
      mov wc.lpfnWndProc, OFFSET EditWndProc
      push hInstance
      pop wc.hInstance
      mov wc.lpszClassName,OFFSET OurClass
      invoke RegisterClassEx, addr wc
      xor ebx,ebx
      mov edi,20
      .while ebx<6
         invoke CreateWindowEx,WS_EX_CLIENTEDGE,ADDR OurClass,NULL,\
            WS_CHILD+WS_VISIBLE+WS_BORDER,20,\
            edi,300,25,hWnd,ebx,\
            hInstance,NULL
         mov dword ptr [hwndEdit+4*ebx],eax
         add edi,25
         inc ebx
      .endw
      invoke SetFocus,hwndEdit
   .elseif uMsg==WM_DESTROY
      invoke PostQuitMessage,NULL
   .else
      invoke DefWindowProc,hWnd,uMsg,wParam,lParam
      ret
   .endif
   xor eax,eax
   ret
WndProc endp

EditWndProc PROC hEdit:DWORD,uMsg:DWORD,wParam:DWORD,lParam:DWORD
   .if uMsg==WM_CHAR
      mov eax,wParam
      .if (al>="0" && al<="9") || (al>="A" && al<="F") || (al>="a" && al<="f") || al==VK_BACK
         .if al>="a" && al<="f"
            sub al,20h
         .endif
         invoke CallWindowProc,OldWndProc,hEdit,uMsg,eax,lParam
```


www.mohandesyar.com

```
            ret
        .endif
    .elseif uMsg==WM_KEYDOWN
        mov eax,wParam
        .if al==VK_RETURN
            invoke MessageBox,hEdit,addr Message,addr AppName,MB_OK+MB_ICONINFORMATION
            invoke SetFocus,hEdit
        .elseif al==VK_TAB
            invoke GetKeyState,VK_SHIFT
            test eax,80000000
            .if ZERO?
                invoke GetWindow,hEdit,GW_HWNDNEXT
                .if eax==NULL
                    invoke GetWindow,hEdit,GW_HWNDFIRST
                .endif
            .else
                invoke GetWindow,hEdit,GW_HWNDPREV
                .if eax==NULL
                    invoke GetWindow,hEdit,GW_HWNDLAST
                .endif
            .endif
            invoke SetFocus,eax
            xor eax,eax
            ret
        .else
            invoke CallWindowProc,OldWndProc,hEdit,uMsg,wParam,lParam
            ret
        .endif
    .else
        invoke CallWindowProc,OldWndProc,hEdit,uMsg,wParam,lParam
        ret
    .endif
    xor eax,eax
    ret
EditWndProc endp
end start
```

بررسی کد فوق:

برنامه ۶ تکست باکس با خواصی اصلاح شده را در ناحیه کلاینت خود نمایش می‌دهد.این تکست باکس‌ها تنها حروف هگز را می‌پذیرند. در ابتدا:

```
    .if uMsg==WM_CREATE
        mov wc.cbSize,sizeof WNDCLASSEX
        invoke GetClassInfoEx,NULL,addr EditClass,addr wc
```

در اینجا اعضای کلاس WNDCLASSEX باید با اعضای کلاسی که قرار است supperclass شود ، پر گردد (در این مورد کلاس EDIT مورد بررسی قرار می‌گیرد). لازم به ذکر است که عضو cbSize باید پیش از فراخوانی تابع GetClassInfoEx ، مقدار دهی شود. در غیراینصورت، در ادامه، اعضای آن به طرز صحیحی مقدار دهی نخواهند شد. پس از فراخوانی موفقیت آمیز این تابع ، wc با مقادیر اطلاعات کلاس جدید پنجره، پر می‌شود. سپس:

www.mohandesyar.com

```
push wc.lpfnWndProc
pop OldWndProc
mov wc.lpfnWndProc, OFFSET EditWndProc
push hInstance
pop wc.hInstance
mov wc.lpszClassName,OFFSET OurClass
```

اکنون مقادیر تعدادی از اعضای این ساختار را می‌توان اصلاح نمود.اولین عضو مورد نظر ، اشاره‌گری است به رویه پنجره. از آنجائیکه نیاز است تا رویه پنجره خود را با رویه اصلی پنجره به صورت زنجیر در آوریم ، لازم است تا مقدار آنرا در متغییری ذخیره کرده و سپس در فراخوانی‌های CallWindowProc از آن استفاده نمائیم. تا اینجا روش همانند subclassing می‌باشد، با این تفاوت که بجای فراخوانی تابع SetWindowLong ، اعضای ساختار WNDCLASSEX را مستقیما اصلاح می‌نمائیم. دو عضو بعدی باید تغییر یابند تا امکان رجیستر کردن کلاس جدید مهیا شود (hInstance and lpsClassName) . باید hInstance اصلی را با دستگیره وهله برنامه خود جایگزین نمایید و همچنین باید نام جدیدی را برای کلاس در نظر بگیرید. در ادامه :

```
invoke RegisterClassEx, addr wc
```

هنگامیکه همه چیز آماده شد، کلاس جدید رجیستر می‌شود.سپس:

```
xor ebx,ebx
mov edi,20
.while ebx<6
   invoke CreateWindowEx,WS_EX_CLIENTEDGE,ADDR OurClass,NULL,\
       WS_CHILD+WS_VISIBLE+WS_BORDER,20,\
       edi,300,25,hWnd,ebx,\
       hInstance,NULL
   mov dword ptr [hwndEdit+4*ebx],eax
   add edi,25
   inc ebx
.endw
invoke SetFocus,hwndEdit
```

زمانیکه کلاس پنجره جدید رجیستر شد، می‌توان برمبنای آن پنجره‌های جدید را ایجاد نمود.در قطعه کد فوق از ebx بعنوان شمارشگر تعداد پنجره‌های ایجاد شده کمک گرفته می‌شود. همچنین از edi بعنوان مختص y سمت چپ بالا پنجره استفاده شد. هنگامیکه یک پنجره ایجاد می‌گردد ، دستگیره آن در آرایه ای از dwords ذخیره می‌شود. پس از ایجاد تمام پنجره‌ها ، تمرکز ورودی را به اولین پنجره انتقال می‌دهیم. تا اینجا ۶ تکست باکس که تنها حروف هگز را می‌پذیرند ایجاد شده است. رویه پنجره جایگزین شده ، فیلتر مورد نظر را اعمال می‌کند.در حقیقت این رویه پنجره با رویه پنجره ایجاد شده در مثال subclassing یکسان است، اما نیاز به کار اضافی در جهت subclassing آنها نیست. همچنین امکان استفاده از tab نیز فراهم شده است. همانطور که در مثال‌های قبل نیز ملاحظه نمودید اگر تکست باکس‌ها را بر روی یک دیالوگ باکس قرار دهیم ، مدیر داخلی آن اینگونه عملیات را بصورت خودکار بعهده خواهد گرفت. اما در یک پنجره معمول چنین چیزی باید توسط برنامه نویس کنترل گردد. در این حالت با subclassing

www.mohandesyar.com

عملیات جابجایی بین تکست باکس‌ها با فشردن دکمه‌های tab مدیریت می‌گردد. در اینجا چون تمام کنترل ها را supperclass کرده ایم دیگر نیازی نیست تا تک تک آنها را subclass نمائیم. بنابراین می توان برای تمام آنها یک کنترل کننده مرکزی طراحی نمود.در ادامه :

```
.elseif al==VK_TAB
   invoke GetKeyState,VK_SHIFT
   test eax,80000000
   .if ZERO?
      invoke GetWindow,hEdit,GW_HWNDNEXT
      .if eax==NULL
         invoke GetWindow,hEdit,GW_HWNDFIRST
      .endif
   .else
      invoke GetWindow,hEdit,GW_HWNDPREV
      .if eax==NULL
         invoke GetWindow,hEdit,GW_HWNDLAST
      .endif
   .endif
   invoke SetFocus,eax
   xor eax,eax
   ret
```

قطعه کد فوق از رویه EditWndClass نمایش داده شد. در اینجا بررسی می‌شود که آیا کاربر دکمه tab را فشرده است یا خیر؟ اگر بله، بررسی می‌شود که آیا کلید shift نیز فشرده شده است؟ تابع GetKeyState ، توسط خروجی خود که در eax قرار می‌گیرد مشخص می‌کند که آیا کلید خاصی فشرده شده است یا خیر؟ اگر کلید مورد نظر فشرده شده‌باشد بیت بالایی eax تنظیم خواهد گردید. در غیراینصورت بیت بالایی خالی خواهد بود. بنابراین مقدار بازگشتی را با 80000000h مقایسه کرده‌ایم.

اگر کاربر تنها کلید tab را بفشارد، تابع GetWindow به همراه پرچم GW_HWNDNEXT ، دستگیره کنترل بعدی را بر می‌گرداند.این دستگیره در hEdit قرار خواهد گرفت. اگر خروجی تابع NULL باشد بدین معنا است که کنترل بعدی دیگری وجود ندارد. دراینجا به اولین کنترل ، بوسیله پرچم GW_HWNDFIRST بازگشت خواهیم کرد.

www.mohandesyar.com

مثال ۲۰- Tray Icon

در این مثال نحوه ایجاد منوی جهنده (pop up) را به همراه طریقه قرار دادن آیکون برنامه در system tray ، خواهیم آموخت.

مراحل قرار دادن آیکون برنامه در system tray به صورت زیر است:

- مقدار دهی اعضای ساختار NOTIFYICONDATA ، که به صورت زیر تعریف می‌شود:

cbSize : اندازه ساختار

hwnd : دستگیره پنجره ای که هنگام رخ دادن کلیک راست بر روی آیکون برنامه در system tray ، پیغام مناسب را دریافت خواهد کرد.

uID : ثابتی است که بیانگر ID آیکون بکار رفته است.اگر در این حالت برنامه از چند آیکون استفاده کند با استفاده از این ID می‌توان دریافت که پیغام ماوس از کدام آیکون صادر شده است.

uFlags : مشخص کننده اعضای معتبر ساختار است.

NIF_ICON : در این حالت عضو hIcon معتبر خواهد بود.

NIF_MESSAGE : در این حالت عضو uCallbackMessage معتبر خواهد بود.

NIF_TIP : در این حالت عضو szTip معتبر میباشد.

uCallbackMessage : پیغامی سفارشی که توسط ویندوز به پنجره مشخص شده با عضو hwnd ، در زمان کلیک ماوس بر روی آیکون برنامه قرار گرفته در system tray ، ارسال می‌شود. این پیغام توسط ما ایجاد خواهد شد.

hIcon : دستگیره آیکونی است که مایل هستید در system tray قرار گیرد.

szTip : آرایه ای به اندازه 64-byte ، که حاوی متن tooltip ایی است که هنگام قرار گرفتن اشاره گر ماوس بر روی آیکون برنامه در system tray ، نمایش می‌یابد.

www.mohandesyar.com

- فراخوانی تابع Shell_NotifyIcon تعریف شده در shell32.inc . این تابع به صورت زیر تعریف شده است:

```
Shell_NotifyIcon PROTO dwMessage:DWORD ,pnid:DWORD
```

dwMessage :نوع پیغامی است که به shell ارسال خواهد شد و یکی از حالتهای زیر می تواند باشد:

NIM_ADD : آیکونی را به system tray اضافه می کند.

NIM_DELETE : آیکونی را از system tray حذف می کند.

NIM_MODIFY : آیکونی را در system tray اصلاح می‌نماید.

pnid : اشاره گری است به ساختار NOTIFYICONDATA است.

هر چند مطالب فوق در مورد قرار دادن آیکون برنامه در system tray کفایت می کنند اما اغلب اوقات نیاز است تا به رخ دادهای ماوس نیز در این ناحیه ، پاسخ مثبتی ارائه داد. برای انجام این امر باید پیغامهای قرار گرفته در عضو uCallbackMessage ساختار NOTIFYICONDATA را پردازش کرد. این پیغام مقادیر زیر را در wParam and lParam دارد:

wParam : حاوی ID آیکون خواهد بود. این مقدار دقیقا معادل با مقدار عضو uID ، ساختار NOTIFYICONDATA است.

lParam : low word آن حاوی پیغام ماوس خواهد بود. برای مثال اگر کاربر بر روی آیکون مربوطه کلیک راست نمایید این عضو حاوی WM_RBUTTONDOWN خواهد گردید.

همچنین در اغلب اوقات پس از کلیک راست بر روی آیکون برنامه در system tray ، منوی جهنده ای ظاهر می‌شود. این مورد را با ایجاد منویی جهنده و سپس فراخوانی تابع TrackPopupMenu جهت نمایش آن می‌توان انجام داد. این مراحل به شرح زیر است:

- ایجاد یک منوی جهنده با فراخوانی تابع CreatePopupMenu . این تابع یک منوی خالی ایجاد می‌نماید و در صورت موفقیت آمیز بودن فراخوانی آن، دستگیره منوی ایجاد شده را در eax قرار می‌دهد.
- اضافه کردن آیتم های منو به آن با فراخوانی توابع AppendMenu, InsertMenu or InsertMenuItem
- هنگامیکه که نیاز است تا منوی جهنده در محل اشاره‌گر ماوس ظاهر شود، تابع GetCursorPos در جهت دریافت مختصات اشاره‌گر فراخوانی شده و سپس از تابع

www.mohandesyar.com

TrackPopupMenu جهت نمایش منو استفاده می‌شود.هنگامیکه کاربر آیتمی از منو را انتخاب نماید ، ویندوز پیغام WM_COMMAND را به رویه پنجره ارسال می‌نماید.

در هنگام کار با منوهای جهنده و آیکون برنامه در system tray ، به دو نکته زیر دقت داشته باشید:

- زمانیکه منوی جهنده نمایش داده شود، اگر در ناحیه ای خارج از منوی نمایش یافته کلیک نمایید، منوی جهنده ناپدید نخواهد شد. زیرا پنجره ای که پیغامهای مربوط به منوی جهنده را دریافت می کند باید پنجره foreground باشد. بنابراین فراخوانی تابع SetForegroundWindow مشکل را حل خواهد کرد.
- پس از فراخوانی تابع SetForegroundWindow ، هنگامیکه برای بار اول منو نمایش می یابد، درست کار کرده و برای دفعات بعدی نمایش ، سریع ظاهر شده و ناپدید می گردد. مطابق مستندات MSDN اینکار جهت برآورده کردن نیازهای آتی ، عمدا صورت گرفته است! در این حالت باید از PostMessage بجای SendMessage استفاده نمود.

برنامه مثال ۲۰ :

```
.386
.model flat,stdcall
option casemap:none
include \masm32\include\windows.inc
include \masm32\include\user32.inc
include \masm32\include\kernel32.inc
include \masm32\include\shell32.inc
includelib \masm32\lib\user32.lib
includelib \masm32\lib\kernel32.lib
includelib \masm32\lib\shell32.lib

WM_SHELLNOTIFY equ WM_USER+5
IDI_TRAY equ 0
IDM_RESTORE equ 1000
IDM_EXIT equ 1010
WinMain PROTO :DWORD,:DWORD,:DWORD,:DWORD

.data
ClassName  db "TrayIconWinClass",0
AppName    db "TrayIcon Demo",0
RestoreString db "&Restore",0
ExitString   db "E&xit Program",0

.data?
hInstance dd ?
```


www.mohandesyar.com

```
note NOTIFYICONDATA <>
hPopupMenu dd ?

.code
start:
   invoke GetModuleHandle, NULL
   mov    hInstance,eax
   invoke WinMain, hInstance,NULL,NULL, 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 or CS_DBLCLKS
   mov   wc.lpfnWndProc, OFFSET WndProc
   mov   wc.cbClsExtra,NULL
   mov   wc.cbWndExtra,NULL
   push  hInst
   pop   wc.hInstance
   mov   wc.hbrBackground,COLOR_APPWORKSPACE
   mov   wc.lpszMenuName,NULL
   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_OVERLAPPED+WS_CAPTION+WS_SYSMENU+WS_MINIMIZEBOX+WS_MAXIMIZEBOX+WS_VISI
BLE,CW_USEDEFAULT,\
         CW_USEDEFAULT,350,200,NULL,NULL,\
         hInst,NULL
   mov   hwnd,eax
   .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
   LOCAL pt:POINT
   .if uMsg==WM_CREATE
      invoke CreatePopupMenu
      mov hPopupMenu,eax
      invoke AppendMenu,hPopupMenu,MF_STRING,IDM_RESTORE,addr RestoreString
      invoke AppendMenu,hPopupMenu,MF_STRING,IDM_EXIT,addr ExitString
   .elseif uMsg==WM_DESTROY
      invoke DestroyMenu,hPopupMenu
      invoke PostQuitMessage,NULL
   .elseif uMsg==WM_SIZE
      .if wParam==SIZE_MINIMIZED
         mov note.cbSize,sizeof NOTIFYICONDATA
         push hWnd
```


www.mohandesyar.com

```
        pop note.hwnd
        mov note.uID,IDI_TRAY
        mov note.uFlags,NIF_ICON+NIF_MESSAGE+NIF_TIP
        mov note.uCallbackMessage,WM_SHELLNOTIFY
        invoke LoadIcon,NULL,IDI_WINLOGO
        mov note.hIcon,eax
        invoke lstrcpy,addr note.szTip,addr AppName
        invoke ShowWindow,hWnd,SW_HIDE
        invoke Shell_NotifyIcon,NIM_ADD,addr note
      .endif
    .elseif uMsg==WM_COMMAND
      .if lParam==0
        invoke Shell_NotifyIcon,NIM_DELETE,addr note
        mov eax,wParam
        .if ax==IDM_RESTORE
          invoke ShowWindow,hWnd,SW_RESTORE
        .else
          invoke DestroyWindow,hWnd
        .endif
      .endif
    .elseif uMsg==WM_SHELLNOTIFY
      .if wParam==IDI_TRAY
        .if lParam==WM_RBUTTONDOWN
          invoke GetCursorPos,addr pt
          invoke SetForegroundWindow,hWnd
          invoke TrackPopupMenu,hPopupMenu,TPM_RIGHTALIGN,pt.x,pt.y,NULL,hWnd,NULL
          invoke PostMessage,hWnd,WM_NULL,0,0
        .elseif lParam==WM_LBUTTONDBLCLK
          invoke SendMessage,hWnd,WM_COMMAND,IDM_RESTORE,0
        .endif
      .endif
    .else
      invoke DefWindowProc,hWnd,uMsg,wParam,lParam
      ret
    .endif
    xor eax,eax
    ret
WndProc endp

end start
```

بررسی کد فوق:

برنامه یک پنجره ساده را ایجاد کرده و پس از انتخاب گزینه به حداقل رساندن اندازه برنامه، در system tray مخفی خواهد شد. با دوبار کلیک کردن بر روی آیکون برنامه ، برنامه مجددا نمایش داده خواهد شد. با کلیک راست روی آیکون برنامه ، منوی جهنده ای با امکان خروج و یا نمایش مجدد برنامه ، نمایش داده خواهد شد. در ادامه :

```
  .if uMsg==WM_CREATE
      invoke CreatePopupMenu
      mov hPopupMenu,eax
      invoke AppendMenu,hPopupMenu,MF_STRING,IDM_RESTORE,addr RestoreString
      invoke AppendMenu,hPopupMenu,MF_STRING,IDM_EXIT,addr ExitString
```


www.mohandesyar.com

هنگامیکه پنجره برنامه ایجاد می‌شود ، منویی جهنده را ایجاد شده و سپس دو آیتم را به آن اضافه می گردد. تعریف تابع AppendMenu به شرح زیر است:

AppendMenu PROTO hMenu:DWORD, uFlags:DWORD, uIDNewItem:DWORD, lpNewItem:DWORD

hMenu : دستگیره منویی است که می‌خواهید آیتمی را به آن اضافه نمایید.

uFlags : خواص آیتم منو مانند رشته ای بودن ، تصویر بودن و غیره را مشخص می کند. برای مشاهده لیست کامل آنها به MSDN مراجعه نمایید. پرچم MF_STRING مشخص می‌نماید که آیتم منو ما رشته ای است.

uIDNewItem : ID آیتم منو تعریف شده توسط ما است.

lpNewItem : بسته به خاصیت مشخص شده توسط پرچم uFlags ، محتوای آیتم منو در این پارامتر قرار می گیرد.از آنجائیکه ما از پرچم MF_STRING استفاده می‌نمائیم، این پارامتر باید اشاره گری به رشته نمایش داده شده توسط منو باشد.

پس از ایجاد پنجره اصلی ، پنجره منتظر فشرده شدن minimize button خواهد ماند. هنگامیکه پنجره حداقل شود پیغام WM_SIZE را با SIZE_MINIMIZED بعنوان wParam دریافت خواهد کرد.

```
.elseif uMsg==WM_SIZE
  .if wParam==SIZE_MINIMIZED
    mov note.cbSize,sizeof NOTIFYICONDATA
    push hWnd
    pop note.hwnd
    mov note.uID,IDI_TRAY
    mov note.uFlags,NIF_ICON+NIF_MESSAGE+NIF_TIP
    mov note.uCallbackMessage,WM_SHELLNOTIFY
    invoke LoadIcon,NULL,IDI_WINLOGO
    mov note.hIcon,eax
    invoke lstrcpy,addr note.szTip,addr AppName
    invoke ShowWindow,hWnd,SW_HIDE
    invoke Shell_NotifyIcon,NIM_ADD,addr note
  .endif
```

ما از این موقعیت برای پر کردن ساختار NOTIFYICONDATA استفاده خواهیم کرد. IDI_TRAY ، ثابتی است که در ابتدای سورس تعریف شده است. شما آنرا به هر مقداری که مایل باشید می‌توانید مقدار دهی نمایید. لازم به ذکر است که اگر از چندین آیکون برای برنامه در حالت system tray استفاده می‌نمائید ، این ثوابت باید یکتا باشند. همچنین تمامی پرچم‌های موجود را برای پارامتر uFlags تنظیم کرده ایم (زیرا از یک آیکون، پیغام سفارشی و متن tooltip استفاده شده است). پیغام WM_SHELLNOTIFY ، پیغامی است سفارشی که بصورت WM_USER+5 تعریف شده است (تنها باید دقت داشت که این پیغام سفارشی یکتا باشد). بعنوان آیکون می توان از یک فایل ریسورس ، آیکون مربوطه را خوانده (توسط تابع LoadIcon) و سپس دستگیره آنرا در hIcon قرار داد. در آخر عضو szTip را مقدار دهی

www.mohandesyar.com

می کنیم. سپس پنجره برنامه را نیز مخفی خواهیم کرد تا حس قرار گرفتن برنامه در system tray را القاء نمائیم. سپس تابع Shell_NotifyIcon را به همراه پیغام NIM_ADD جهت قرار گرفتن آیکون برنامه در system tray ، فراخوانی می کنیم. اکنون برنامه ما مخفی شده و همچنین در system tray قرار گرفته است. اگر اشاره گر ماوس را بر روی آیکون برنامه قرار دهیم ، متن قرار گرفته شده در szTip ، بعنوان tooltip نمایش داده خواهد شد. اگر دوبار روی آیکون برنامه کلیک کنیم ، آیکون برنامه از system tray حذف شده و پنجره برنامه نمایش داده خواهد شد.

```
.elseif uMsg==WM_SHELLNOTIFY
   .if wParam==IDI_TRAY
      .if lParam==WM_RBUTTONDOWN
         invoke GetCursorPos,addr pt
         invoke SetForegroundWindow,hWnd
         invoke TrackPopupMenu,hPopupMenu,TPM_RIGHTALIGN,pt.x,pt.y,NULL,hWnd,NULL
         invoke PostMessage,hWnd,WM_NULL,0,0
      .elseif lParam==WM_LBUTTONDBLCLK
         invoke SendMessage,hWnd,WM_COMMAND,IDM_RESTORE,0
      .endif
   .endif
```

زمانیکه رخداد ماوس بر روی tray icon قرار گیرد، پنجره شما پیغام WM_SHELLNOTIFY را که پیغامی است سفارشی قرار گرفته در uCallbackMessage ، دریافت خواهد کرد. همانطور که پیش تر نیز ذکر شد ، در هنگام دریافت این پیغام ، wParam حاوی ID آیکون و lParam حاوی پیغام اصلی ماوس خواهد بود.

در کد فوق ابتدا بررسی می کنیم که آیا پیغام رسیده مربوط به tray icon برنامه ما است یا خیر؟ در صورت مثبت بودن پاسخ ، پیغام اصلی ماوس را پردازش خواهیم کرد. از آنجائیکه تنها دوبار کلیک شدن ماوس و همچنین کلیک راست ماوس برای ما مهم است ، پیغامهای WM_RBUTTONDOWN and WM_LBUTTONDBLCLK را پردازش خواهیم کرد. اگر پیغام ماوس WM_RBUTTONDOWN باشد ، ما با فراخوانی تابع GetCursorPos ، مختصات اشاره گر ماوس را در صفحه نمایش دریافت خواهیم کرد. پس از خاتمه کار تابع ، خروجی آن در ساختار POINT قرار می گیرد. برای تبدیل مختصات از مختصات صفحه نمایش به مختصات پنجره ، از تابع ScreenToClient استفاده می‌شود.البته تابع TrackPopupMenu جهت نمایش منوی جهنده در محل شاره گر ماوس ، از مختصات صفحه نمایش استفاده می‌نماید. تعریف تابع TrackPopupMenu به شرح زیر است:

```
TrackPopupMenu PROTO hMenu:DWORD, uFlags:DWORD,\
                     x:DWORD,  y:DWORD, nReserved:DWORD,\
                     hWnd:DWORD, prcRect:DWORD
```

hMenu : دستگیره منوی جهنده ای است که نمایش خواهد یافت.

uFlags : گزینه‌های تابع را مشخص می کند.برای مثال از کدام دکمه ی ماوس برای ردیابی منو استفاده خواهد شد و یا منو در کجا ظاهر شود. برای مثال استفاده از TPM_RIGHTALIGN جهت تعیین محل نمایش اشاره گر ماوس.

www.mohandesyar.com

x and y : موقعیت منو را در مختصات صفحه نمایشی مشخص می کند.

nReserved : باید مساوی NULL قرار گیرد.

hWnd : دستگیره پنجره ای است که که پیغام‌ها را از منو دریافت خواهد کرد.

prcRect : مستطیلی است فرضی در صفحه نمایش که با کلیک شدن در ناحیه کلاینت آن ، منوی جهنده مخفی نخواهد شد. بطور معمول این پارامتر را مساوی نال قرار می دهیم تا با کلیک کاربر در هر نقطه ای از صفحه نمایش ، منوی جهنده مخفی شود.

هنگامیکه کاربر دوبار بر روی آیکون برنامه در system tray کلیک نماید ، پیغام WM_COMMAND را به پنجره برنامه به همراه IDM_RESTORE خواهیم فرستاد. در این حالت پنجره اصلی برنامه دوباره نمایش داده خواهد شد و آیکون برنامه از system tray حذف خواهد شد. برای دریافت دوبار کلیک ماوس، پنجره برنامه باید دارای سبک CS_DBLCLKS باشد.

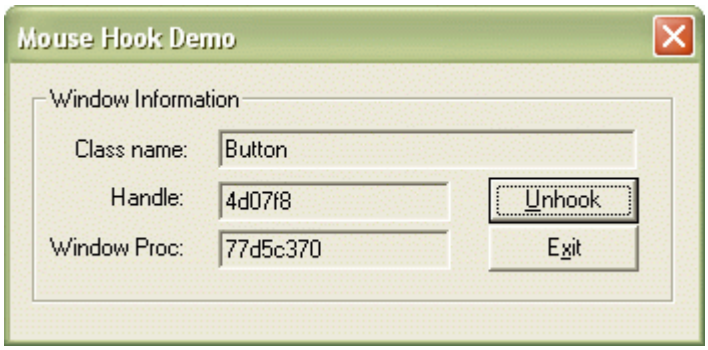
```
invoke Shell_NotifyIcon,NIM_DELETE,addr note
mov eax,wParam
.if ax==IDM_RESTORE
   invoke ShowWindow,hWnd,SW_RESTORE
.else
   invoke DestroyWindow,hWnd
.endif
```

هنگامیکه کاربر گزینه restore را انتخاب کند، برای حذف آیکون آن از system tray ، تابع Shell_NotifyIcon را به همراه پیغام NIM_DELETE ، فراخوانی می‌نمائیم. سپس پنجره را به حالت اصلی خود بازگردانده و نمایش می‌دهیم.

اگر کاربر گزینه خروج را انتخاب نماید، علاوه بر حذف آیکون برنامه از system tray ، پنجره آنرا با فراخوانی تابع DestroyWindow تخریب خواهیم کرد.

www.mohandesyar.com

مثال ۲۱- Windows Hooks



Windowd hooks ابزاری بسیار قدرتمند محسوب می‌شوند. بوسیله آنها می‌توان درون سایر پروسه‌ها کنجاوی نمود و یا گاهی از اوقات رفتار آنها را تغییر داد. همچنین توسط آنها امکان حبس وقایعی که درون پروسه برنامه و یا سایر پروسه‌ها قرار است رخ دهند نیز میسر است.

در طی عملیات hooking ، به ویندوز در مورد یک تابع اطلاعاتی داده می‌شود. به این تابع hook procedure نیز گفته شده و در زمان رخ دادن وقایع مورد علاقه، فراخوانی می گردد. دو نوع از آنها وجود دارند: محلی و از راه دور.

- قلابهای محلی (local hooks)، رخ دادهای درون پروسه برنامه را حبس می کنند (به تله می‌اندازند).
- قلابهای راه دور (Remote hooks)، رخ دادهای درون سایر پروسه‌ها را حبس می کنند.دو نوع از قلابهای راه دور نیز وجود دارند:

۱. بدام انداختن وقایع رخ داده درون تردی مشخص از پروسه ای مشخص خارج از برنامه.
۲. بدام انداختن تمامی وقایع کلیه تردهای تمام پروسه‌های سیستم.

در هنگام استفاده از قلابها باید در نظر داشت که آنها بر روی کارآیی سیستم تاثیر منفی خواهند گذاشت ، خصوصا قلابهای راه دور نوع دوم. زیرا تمامی رخ داده های مربوطه باید از تابع فیلتر کننده‌ی شما عبور نمایند. همچنین لازم به ذکر است که اگر در تابع فیلتر کننده قلاب اشتباهی رخ دهد امکان از کار انداختن دیگر پروسه‌ها بسیار زیاد است.

هنگامیکه قلابی را ایجاد می‌نمائید، ویندوز ساختار داده ای اطلاعات قلاب را در حافظه ایجاد کرده و آنرا به لیست پیوندی قلابهای موجود اضافه می کند. هنگامیکه رخ دادی به وقوع می‌پیوندد، اگر قلابی محلی را نصب کرده باشید، تابع فیلتر کننده قلاب فراخوانی می گردد.اما اگر این قلاب از نوع راه دور باشد ، ویندوز باید کد مربوط به رویه قلاب

www.mohandesyar.com

را به درون فضای آدرس دهی پروسه دیگر تزریق کند و ویندوز این کار را تنها در صورتی انجام خواهد داد که رویه مورد نظر در یک dll قرار داشته باشد. البته دو استثناء برای این قاعده وجود دارد:

Journal record and journal playback hooks

رویه قلاب این دو استثناء باید در تردی که قلاب را نصب می کند ، قرار داشته باشد. دلیل این امر آن است که دو قلاب ذکر شده با رخ دادهای سطح پایین ورودی سخت افزاری سروکار دارند. رخ دادهای ورودی باید ضبط شده و سپس بازپخش شوند تا اثرات آنها پدیدار گردند. اگر رویه این دو قلاب در یک dll قرار گیرد، رخ دادهای ورودی ممکن است بین چندین ترد پراکنده شده و دانستن ترتیب آنها غیرممکن خواهد شد. بنابراین راه حل ارائه شده این است که رویه این دو قلاب باید تنها در یک ترد نصب کننده قلاب قرار گیرند.

۱۴ نوع قلاب وجود دارند:

WH_CALLWNDPROC: با فراخوانی SendMessage ، فراخوانی خواهد شد.

WH_CALLWNDPROCRET: زمانیکه کار تابع SendMessage به پایان برسد، فراخوانی می‌شود.

WH_GETMESSAGE: زمانیکه PeekMessage or GetMessage ، فراخوانی شود، فراخوانی می گردد.

WH_KEYBOARD: زمانیکه PeekMessage or GetMessage ، پیغامهای WM_KEYUP or WM_KEYDOWN را دریافت کنند، فراخوانی می‌شود.

WH_MOUSE: زمانیکه PeekMessage or GetMessage ، پیغامهای ماوس را دریافت کنند، فراخوانی می‌شود.

WH_HARDWARE: هنگامیکه GetMessage or PeekMessage ، پیغامهای سخت افزاری که به ماوس و کیبورد مربوط نمی‌باشند را دریافت کنند، فراخوانی می‌شود.

WH_MSGFILTER: زمانیکه dialog box, menu or scrollbar ، قصد شروع به پردازش پیغامی را داشته باشند ، فراخوانی می گردد.این قلاب از نوع محلی است. این قلاب مخصوص اشیاء ایی است که دارای حلقه مدیریت داخلی هستند.

WH_SYSMSGFILTER: همانند WH_MSGFILTER است اما در سطح کل سیستم عمل می کند.

WH_JOURNALRECORD: زمانیکه ویندوز پیغامی را از ورودی سخت افزاری دریافت نماید، فراخوانی می‌شود.

WH_JOURNALPLAYBACK: زمانیکه رخدادی از طرف ورودی سخت افزاری درخواست شود، فراخوانی می گردد.

www.mohandesyar.com

WH_SHELL: هنگامیکه رخدادی مربوط به shell اجرا می گردد ، فراخوانی می‌شود. برای مثال زمانیکه task bar قصد دارد تا دکمه های خود را مجددا ترسیم نماید.

WH_CBT: برای computer-based training طراحی شده است.

WH_FOREGROUNDIDLE: توسط ویندوز بکار گرفته می‌شود و کاربرد عمومی ندارد.

WH_DEBUG: برای دیباگ کردن رویه قلاب بکار می‌رود.

برای نصب یک قلاب از تابع SetWindowsHookEx استفاده می‌شود. تعریف آن به شرح زیر است:

```
SetWindowsHookEx proto HookType:DWORD, pHookProc:DWORD,\
                       hInstance:DWORD, ThreadID:DWORD
```

HookType: یکی از مقادیری است که در بالا لیست آنها ارائه شد مانند **WH_MOUSE** و غیره.

pHookProc: آدرس رویه قلاب است. از این رویه برای پردازش پیغامهای قلاب مشخص‌شده استفاده می گردد. اگر قلاب از نوع راه دور است ، باید این رویه درون یک dll قرار گیرد. در غیراینصورت باید در پروسه برنامه قرار داشته باشد.

hInstance: دستگیره وهله dll ایی است که رویه قلاب در آن قرار دارد. اگر قلاب از نوع محلی است ، این پارامتر باید مساوی NULL قرار گیرد.

ThreadID: ID تردی است که قلاب در آن نصب خواهد شد. این پارامتر مشخص می‌نماید که قلاب از نوع محلی است یا خیر؟ اگر این پارامتر مساوی نال قرار گیرد، ویندوز این قلاب را system-wide remote hook تفسیر کرده و بر روی تمامی تردهای سیستم تاثیر خواهد داشت. اگر ID تردی را در پروسه برنامه خود مشخص نمائیم ، این قلاب محلی خواهد بود. اگر این ID از تردهای سایر پروسه ها انتخاب شود، این قلاب از نوع مشخص شده با ترد و از راه دور می‌باشد. دو استثناء برای این قاعده وجود دارد: **WH_JOURNALRECORD** and **WH_JOURNALPLAYBACK** که همواره از نوع local system-wide hooks هستند و نیازی نیست تا در dll قرار گیرند. همچنین **WH_SYSMSGFILTER** از نوع system-wide remote hook می‌باشد. در حقیقت این حالت معادل است با **WH_MSGFILTER** زمانیکه ThreadID مساوی صفر قرار گیرد.

اگر فراخوانی تابع فوق موفقیت آمیز باشد ، دستگیره قلاب در eax قرار خواهد گرفت. در غیراینصورت نال بازگشت داده می‌شود. این دستگیره برای unhooking بعدا مورد استفاده قرار خواهد گرفت. برای عزل یک قلاب نصب شده، از فراخوانی تابع **UnhookWindowsHookEx** به همراه دستگیره قلاب بعنوان پارامتر ورودی استفاده می‌شود.

www.mohandesyar.com

رویه یک قلاب هربار که رخدادی وابسته به نوع قلاب نصب شده قصد اجرا دارد، فراخوانی می گردد. برای مثال اگر قلاب **WH_MOUSE** نصب گردد، هنگامیکه رخ داد ماوس روی دهد، رویه قلاب شما اجرا می گردد. صرفنظر از نوع قلاب نصب شده، رویه قلاب تعریف زیر را خواهد داشت:

```
HookProc proto nCode:DWORD, wParam:DWORD, lParam:DWORD
```

nCode : کد قلاب مورد نظر را اجرا مشخص می کند.

wParam and lParam : حاوی اطلاعات اضافی در مورد رخداد است.

البته این رویه تا زمانیکه قالب تعریف فوق را حفظ نماید، می‌تواند هر نام دلخواهی داشته باشد.تفسیر nCode, wParam and lParam وابسته است به نوع قلابی که نصب کرده اید.

برای مثال در حالت **WH_CALLWNDPROC** :

nCode : تنها می‌تواند مساوی HC_ACTION باشد و بدین معنا است که پیغامی به پنجره ارسال شده است.

wParam : اگر صفر نبود، حاوی پیغامی می‌باشد که شروع به ارسال شده است.

lParam : اشاره گری است به ساختار CWPSTRUCT .

خروجی: استفاده ای ندارد و صفر بازگشت داده می‌شود.

مثالی دیگر: در حالت **WH_MOUSE** :

nCode : تنها می‌تواند HC_ACTION or HC_NOREMOVE باشد.

wParam : حاوی پیغامهای ماوس است.

lParam : اشاره گری به MOUSEHOOKSTRUCT خواهد بود.

خروجی: اگر پیغام قرار است پردازش شود، این خروجی باید صفر باشد.

برای مطالعه جزئیات سایر موارد ، به یک مرجع API ویندوز مراجعه نمائید.

بخاطر داشته باشید که قلاب ها در یک لیست پیوندی (linked list) قرار گرفته و جدیدترین قلاب نصب شده در ابتدای لیست قرار می گیرد.هنگامیکه رخدادی قرار است واقع شود، ویندوز تنها اولین قلاب را در لیست پیوندی فراخوانی خواهد کرد. این وظیفه‌ی رویه قلاب شما است که قلاب بعدی را در لیست پیوندی فراخوانی نماید. البته می‌توانید اینکار را هم انجام ندهید اما بهتر است بدانید که در حال انجام چه کاری هستید.در اغلب موارد بهتر است

www.mohandesyar.com

قلاب بعدی نیز فراخوانی شود تا سایر قلاب ها نیز تصویری از رخداد در حال وقوع را داشته باشند. فراخوانی قلاب بعدی با فراخوانی تابع **CallNextHookEx** صورت می گیرد. تعریف این تابع به صورت زیر است:

CallNextHookEx proto hHook:DWORD, nCode:DWORD, wParam:DWORD, lParam:DWORD

hHook : دستگیره قلاب مورد نظر است.

nCode, wParam and lParam : این مقادیر از ویندوز دریافت شده و به تابع فوق ارسال می‌شوند.

نکته‌ی مهمی در باره قلاب‌های راه دور:

کد رویه قلاب باید در dll ایی قرار گیرد که به سایر پروسه‌ها نگاشت خواهد شد. زمانیکه این نگاشت صورت گیرد قسمت data مربوط به dll نگاشت نخواهد شد. بنابراین هر پروسه قسمت data مخصوص به خود را خواهد داشت. در زمان نصب یک قلاب نیاز است تا قسمت data بین تمامی پروسه‌ها مشترک باشد. برای حل این مشکل باید قسمت داده‌ها به صورت به اشتراک گذاشته شده تعریف شود. برای مثال در MASM سوئیچ زیر را باید برای liker بکار برد:

/SECTION:<section name>, S

نام قسمت داده های مقدار دهی اولیه شده، data. بوده و قسمت مقدار دهی اولیه نشده ، bss. است. برای مثال جهت به اشتراک گذاشت قسمت داده های مقدار دهی اولیه نشده ، باید از دستور زیر استفاده کرد:

link /section:.bss,S /DLL /SUBSYSTEM:WINDOWS .....

S در اینجا به معنای به اشتراک گذاشته شده است.

کد مثال ۲۱ :

```
;------------------------------------ This is the source code of the main program ----------------------------------
.386
.model flat,stdcall
option casemap:none
include \masm32\include\windows.inc
include \masm32\include\user32.inc
include \masm32\include\kernel32.inc
include mousehook.inc
includelib mousehook.lib
includelib \masm32\lib\user32.lib
includelib \masm32\lib\kernel32.lib
```


www.mohandesyar.com

```
wsprintfA proto C :DWORD,:DWORD,:VARARG
wsprintf TEXTEQU <wsprintfA>

.const
IDD_MAINDLG             equ 101
IDC_CLASSNAME           equ 1000
IDC_HANDLE              equ 1001
IDC_WNDPROC             equ 1002
IDC_HOOK                equ 1004
IDC_EXIT                equ 1005
WM_MOUSEHOOK            equ WM_USER+6

DlgFunc PROTO :DWORD,:DWORD,:DWORD,:DWORD

.data
HookFlag dd FALSE
HookText db "&Hook",0
UnhookText db "&Unhook",0
template db "%lx",0

.data?
hInstance dd ?
hHook dd ?
.code
start:
    invoke GetModuleHandle,NULL
    mov hInstance,eax
    invoke DialogBoxParam,hInstance,IDD_MAINDLG,NULL,addr DlgFunc,NULL
    invoke ExitProcess,NULL

DlgFunc proc hDlg:DWORD,uMsg:DWORD,wParam:DWORD,lParam:DWORD
    LOCAL hLib:DWORD
    LOCAL buffer[128]:byte
    LOCAL buffer1[128]:byte
    LOCAL rect:RECT
    .if uMsg==WM_CLOSE
        .if HookFlag==TRUE
            invoke UninstallHook
        .endif
        invoke EndDialog,hDlg,NULL
    .elseif uMsg==WM_INITDIALOG
        invoke GetWindowRect,hDlg,addr rect
        invoke SetWindowPos, hDlg, HWND_TOPMOST, rect.left, rect.top, rect.right, rect.bottom,
SWP_SHOWWINDOW
    .elseif uMsg==WM_MOUSEHOOK
        invoke GetDlgItemText,hDlg,IDC_HANDLE,addr buffer1,128
        invoke wsprintf,addr buffer,addr template,wParam
        invoke lstrcmpi,addr buffer,addr buffer1
        .if eax!=0
            invoke SetDlgItemText,hDlg,IDC_HANDLE,addr buffer
        .endif
        invoke GetDlgItemText,hDlg,IDC_CLASSNAME,addr buffer1,128
        invoke GetClassName,wParam,addr buffer,128
        invoke lstrcmpi,addr buffer,addr buffer1
        .if eax!=0
            invoke SetDlgItemText,hDlg,IDC_CLASSNAME,addr buffer
        .endif
        invoke GetDlgItemText,hDlg,IDC_WNDPROC,addr buffer1,128
        invoke GetClassLong,wParam,GCL_WNDPROC
        invoke wsprintf,addr buffer,addr template,eax
```


www.mohandesyar.com

```
      invoke lstrcmpi,addr buffer,addr buffer1
      .if eax!=0
         invoke SetDlgItemText,hDlg,IDC_WNDPROC,addr buffer
      .endif
   .elseif uMsg==WM_COMMAND
      .if lParam!=0
         mov eax,wParam
         mov edx,eax
         shr edx,16
         .if dx==BN_CLICKED
            .if ax==IDC_EXIT
               invoke SendMessage,hDlg,WM_CLOSE,0,0
            .else
               .if HookFlag==FALSE
                  invoke InstallHook,hDlg
                  .if eax!=NULL
                     mov HookFlag,TRUE
                     invoke SetDlgItemText,hDlg,IDC_HOOK,addr UnhookText
                  .endif
               .else
                  invoke UninstallHook
                  invoke SetDlgItemText,hDlg,IDC_HOOK,addr HookText
                  mov HookFlag,FALSE
                  invoke SetDlgItemText,hDlg,IDC_CLASSNAME,NULL
                  invoke SetDlgItemText,hDlg,IDC_HANDLE,NULL
                  invoke SetDlgItemText,hDlg,IDC_WNDPROC,NULL
               .endif
            .endif
         .endif
      .endif
   .else
      mov eax,FALSE
      ret
   .endif
   mov eax,TRUE
   ret
DlgFunc endp

end start
```

;---------- This is the source code of the DLL ----------

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

.const
WM_MOUSEHOOK equ WM_USER+6

.data
hInstance dd 0

.data?
hHook dd ?
hWnd dd ?
```


www.mohandesyar.com

```
.code
DllEntry proc hInst:HINSTANCE, reason:DWORD, reserved1:DWORD
   .if reason==DLL_PROCESS_ATTACH
      push hInst
      pop hInstance
   .endif
   mov  eax,TRUE
   ret
DllEntry Endp

MouseProc proc nCode:DWORD,wParam:DWORD,lParam:DWORD
   invoke CallNextHookEx,hHook,nCode,wParam,lParam
   mov edx,lParam
   assume edx:PTR MOUSEHOOKSTRUCT
   invoke WindowFromPoint,[edx].pt.x,[edx].pt.y
   invoke PostMessage,hWnd,WM_MOUSEHOOK,eax,0
   assume edx:nothing
   xor eax,eax
   ret
MouseProc endp

InstallHook proc hwnd:DWORD
   push hwnd
   pop hWnd
   invoke SetWindowsHookEx,WH_MOUSE,addr MouseProc,hInstance,NULL
   mov hHook,eax
   ret
InstallHook endp

UninstallHook proc
   invoke UnhookWindowsHookEx,hHook
   ret
UninstallHook endp

End DllEntry

;---------------------------------------- This is the makefile of the DLL ----------------------------------------

NAME=mousehook
$(NAME).dll: $(NAME).obj
      Link /SECTION:.bss,S  /DLL /DEF:$(NAME).def /SUBSYSTEM:WINDOWS /LIBPATH:c:\masm\lib
$(NAME).obj
$(NAME).obj: $(NAME).asm
      ml /c /coff /Cp $(NAME).asm
```

بررسی کد فوق:

همانطور که در تصویر ابتدای این مثال نیز مشخص است ، این مثال از یک دیالوگ باکس به همراه سه تکست باکس که با نام کلاس، دستگیره پنجره و آدرس رویه پنجره قرار گرفته در زیر اشاره گر ماوس پر می شوند، تشکیل شده است. دو دکمه hook و exit نیز بر روی پنجره برنامه قرار دارند. همچنین پیغام سفارشی WM_MOUSEHOOK نیز بین

www.mohandesyar.com

برنامه اصلی و dll قلاب ، تعریف شده است. هنگامیکه برنامه اصلی این پیغام را دریافت نماید، wParam حاوی دستگیره پنجره‌ای است که اشاره‌گر ماوس بر روی آن قرار دارد. در ادامه :

```
.if HookFlag==FALSE
   invoke InstallHook,hDlg
   .if eax!=NULL
      mov HookFlag,TRUE
      invoke SetDlgItemText,hDlg,IDC_HOOK,addr UnhookText
   .endif
```

برنامه از یک پرچم به نام HookFlag جهت تعیین وضعیت قلاب استفاده می‌کند. اگر FALSE باشد به این معنا است که قلاب نصب نشده است و برعکس.

زمانیکه کاربر دکمه hook را می فشارد، ابتدا برنامه بررسی می کند که آیا قلاب نصب شده است یا خیر؟ درصورت منفی بودن پاسخ ، تابع InstallHook مربوط به dll قلاب را فراخوانی می‌کند. لازم به ذکر است ، چون دستگیره دیالوگ باکس اصلی بعنوان پارامتر تابع به dll قلاب ارسال شده است ، بنابراین dll ، توانایی ارسال پیغام WM_MOUSEHOOK را به پنجره صحیح دارد.

هنگامیکه برنامه بارگذاری می‌شود، dll قلاب نیز بارگذاری خواهد شد. لازم به ذکر است که تابع entrypoint دی‌ال‌ال، قبل از اجرای اولین اینستراکشن برنامه اصلی بارگذاری می‌شود.

کد زیر را در تابع entrypoint مربوط به dll قلاب قرار داده ایم:

```
.if reason==DLL_PROCESS_ATTACH
   push hInst
   pop hInstance
.endif
```

کد ، دستگیره وهله dll را در یک متغیر عمومی به نام hInstance ذخیره می‌کند.از آن در تابع InstallHook استفاده خواهد شد. از آنجائیکه تابع entrypoint مربوط به dll ، قبل از فراخوانی سایر توابع dll فراخوانی می‌شود، hInstance همواره معتبر خواهد بود. ما hInstance را در قسمت data. قرار داده ایم ، بنابراین این مقدار به ازای هر پروسه حفظ خواهد گردید.

زمانیکه اشاره‌گر ماوس بر روی یک پنجره قرار می گیرد، dll قلاب به درون پروسه آن نگاشت خواهد شد. فرض کنید در این لحظه dll دیگری در آدرس مورد نظر برای dll قلاب جهت نگاشت شدن ، قرار داشته باشد. در این حالت dll قلاب به درون پروسه دیگری مجددا نگاشت خواهد شد. همچنین مقدار hInstance نیز به مقدار جدید مربوط به آدرس جدید ، تغییر خواهد کرد. زمانیکه کاربر دکمه Unhook و سپس دکمه hook را بفشارد ، تابع SetWindowsHookEx دوباره فراخوانی خواهد شد. در این لحظه از آدرس بارگذاری جدید بعنوان دستگیره وهله استفاده خواهد کرد که نادرست می‌باشد، زیرا در پروسه مثال ، آدرس بارگذاری dll قلاب تغییری نکرده است. قلاب، نمونه ای محلی شده و تنها رخدادهای ماوس رخ داده در پنجره کلاینت برنامه را می‌تواند به دام بیاندازد. سپس:

www.mohandesyar.com

```
InstallHook proc hwnd:DWORD
   push hwnd
   pop hWnd
   invoke SetWindowsHookEx,WH_MOUSE,addr MouseProc,hInstance,NULL
   mov hHook,eax
   ret
InstallHook endp
```

تابع InstallHook بسیار ساده می‌باشد. آن پارامتر ورودی خود را در یک متغیر عمومی به نام hWnd جهت استفاده‌های آتی ذخیره می‌کند. سپس تابع SetWindowsHookEx را برای نصب قلاب ماوس فراخوانی می‌کند. خروجی این تابع در متغیر عمومی hHook برای استفاده در تابع UnhookWindowsHookEx ذخیره می‌شود.

پس از فراخوانی SetWindowsHookEx ، قلاب ماوس قابل استفاده می‌باشد. هرگاه در سیستم رخ ماوسی به وقع بپیوندد ، رویه قلاب (MouseProc) اجرا می‌شود:

```
MouseProc proc nCode:DWORD,wParam:DWORD,lParam:DWORD
   invoke CallNextHookEx,hHook,nCode,wParam,lParam
   mov edx,lParam
   assume edx:PTR MOUSEHOOKSTRUCT
   invoke WindowFromPoint,[edx].pt.x,[edx].pt.y
   invoke PostMessage,hWnd,WM_MOUSEHOOK,eax,0
   assume edx:nothing
   xor eax,eax
   ret
MouseProc endp
```

اولین کاری که پروسه فوق انجام می‌دهد، فراخوانی تابع CallNextHookEx است تا به سایر قلابها نیز فرصت دریافت رخدادهای ماوس نیز داده شود. سپس تابع WindowFromPoint جهت دریافت دستگیره پنجره‌ای در مختصات معینی از صفحه نمایش، فراخوانی می‌گردد. از ساختار POINT در ساختار MOUSEHOOKSTRUCT ، که lParam اشاره گری است به آن ، بعنوان مختصات جاری اشاره‌گر ماوس استفاده گردید. سپس دستگیره پنجره را به برنامه اصلی با فراخوانی تابع PostMessage ، به همراه پیغام WM_MOUSEHOOK ، ارسال می‌نمائیم. همانطور که پیشتر نیز ذکر گردید، نباید از تابع SendMessage درون رویه قلاب استفاده کرد و PostMessage توصیه می‌شود. ساختار MOUSEHOOKSTRUCT در ذیل تعریف شده است:

```
MOUSEHOOKSTRUCT STRUCT DWORD
  pt          POINT <>
  hwnd        DWORD     ?
  wHitTestCode  DWORD     ?
  dwExtraInfo   DWORD     ?
MOUSEHOOKSTRUCT ENDS
```

pt : مختصات جاری اشاره‌گر ماوس

hwnd : دستگیره پنجره ای است که پیغام ماوس را دریافت می‌کند. این پنجره عموما، پنجره‌ای است که اشاره‌گر ماوس بر روی آن قرار دارد (اما نه همیشه). اگر پنجره ای تابع

www.mohandesyar.com

SetCapture را فراخوانی نماید، ورودی ماوس به آن پنجره هدایت خواهد شد. به همین جهت در این مثال از hwnd این ساختار استفاده نشد و بجای آن از تابع WindowFromPoint استفاده گردید.

wHitTestCode : توسط این مقدار، اطلاعات بیشتری در مورد موقعیت جاری اشاره‌گر ماوس دریافت می‌شود.بدین وسیله مشخص می‌شود که اشاره‌گر ماوس در کدام قسمت از پنجره قرار دارد. برای اطلاعات بیشتر به راهنمای API ویندوز در مورد WM_NCHITTEST مراجعه بفرمایید.

dwExtraInfo : اطلاعات اضافی همراه پیغام را مشخص می‌کند. بطور معمول این مقدار توسط فراخوانی mouse_event و دریافت اطلاعات آن توسط تابع GetMessageExtraInfo مشخص می‌گردد.

هنگامیکه پنجره اصلی پیغام WM_MOUSEHOOK را دریافت کند، از دستگیره پنجره جهت دریافت اطلاعات مورد نظر استفاده می‌شود.

```
.elseif uMsg==WM_MOUSEHOOK
   invoke GetDlgItemText,hDlg,IDC_HANDLE,addr buffer1,128
   invoke wsprintf,addr buffer,addr template,wParam
   invoke lstrcmpi,addr buffer,addr buffer1
   .if eax!=0
      invoke SetDlgItemText,hDlg,IDC_HANDLE,addr buffer
   .endif
   invoke GetDlgItemText,hDlg,IDC_CLASSNAME,addr buffer1,128
   invoke GetClassName,wParam,addr buffer,128
   invoke lstrcmpi,addr buffer,addr buffer1
   .if eax!=0
      invoke SetDlgItemText,hDlg,IDC_CLASSNAME,addr buffer
   .endif
   invoke GetDlgItemText,hDlg,IDC_WNDPROC,addr buffer1,128
   invoke GetClassLong,wParam,GCL_WNDPROC
   invoke wsprintf,addr buffer,addr template,eax
   invoke lstrcmpi,addr buffer,addr buffer1
   .if eax!=0
      invoke SetDlgItemText,hDlg,IDC_WNDPROC,addr buffer
   .endif
```

برای جلوگیری از چشمک زدن صفحه (flicker) ، مقدار متن دریافتی و متن موجود در کنترل‌های تکست باکس مقایسه شده و تنها مقادیر جدید نمایش خواهند یافت.

برای دریافت نام کلاس از تابع GetClassName استفاده گردید. جهت دریافت آدرس رویه پنجره، از تابع GetClassLong به همراه GCL_WNDPROC کمک گرفته شد. در ادامه :

```
invoke UninstallHook
invoke SetDlgItemText,hDlg,IDC_HOOK,addr HookText
mov HookFlag,FALSE
invoke SetDlgItemText,hDlg,IDC_CLASSNAME,NULL
```


www.mohandesyar.com

```
invoke SetDlgItemText,hDlg,IDC_HANDLE,NULL
invoke SetDlgItemText,hDlg,IDC_WNDPROC,NULL
```

هنگامیکه کاربر دکمه Unhook را بفشارد، تابع UninstallHook در dll قلاب فراخوانی خواهد شد. این تابع ، تنها تابع UnhookWindowsHookEx را فراخوانی می‌نماید.سپس برچسب دکمه به Hook و مقدار HookFlag به FALSE تغییر خواهد یافت. سپس محتوای تکست باکس‌ها خالی می گردد.

همچنین به سوئیچ های بکار گرفته شده در Linker نیز دقت نمائید:

```
Link /SECTION:.bss,S  /DLL /DEF:$(NAME).def /SUBSYSTEM:WINDOWS
```

در این حالت قسمت bss. کد ، بین تمام پروسه های استفاده کننده از dll به اشتراک گذاشته می‌شود. بدون این سوئیچ، dll قلاب کار نخواهد کرد.

www.mohandesyar.com

مثال ۲۲– نمایش یک فایل بیت‌مپ

قالبهای متعددی جهت فایلهای گرافیکی وجود دارند، اما ویندوز به صورت ذاتی و پیش فرض، تنها فایلهای bmp. را پشتیبانی می‌کند (Windows bitmap graphics files). ساده‌ترین راه استفاده از بیت‌مپ‌ها ، قرار دادن آنها در یک فایل ریسورس است. دو راه برای انجام اینکار وجود دارد:

- افزودن آن به فایل rc. و تعریف یک ثابت جهت نمایش بیت مپ:

```
#define  IDB_MYBITMAP   100
IDB_MYBITMAP  BITMAP  "c:\project\example.bmp"
```

ما از ثابت IDB_MYBITMAP جهت ارجاع به بیت مپ در برنامه استفاده خواهیم کرد. خط بعدی محل قرارگیری فایل را مشخص می‌کند.

- روش دیگر استفاده از یک نام (یک رشته)، جهت نمایش بیت مپ می‌باشد.

```
MyBitMap  BITMAP "c:\project\example.bmp"
```

پس از تعریف فایل بیت مپ ، نحوه نمایش آن به صورت زیر است:

www.mohandesyar.com

۱. فراخوانی تابع LoadBitmap جهت دریافت دستگیره بیت مپ. این تابع به صورت زیر تعریف می‌شود:

```
LoadBitmap proto hInstance:HINSTANCE, lpBitmapName:LPSTR
```

hInstance : دستگیره وهله برنامه است.

lpBitmapName : اشاره‌گری است به رشته‌ای که نام بیت مپ می‌باشد. (این حالت مربوط است به روش دوم تعریف فایل بیت مپ). اگر از روش اول استفاده نمایید باید مقدار ثابت را در این پارامتر قرار داد. برای مثال در اینجا ۱۰۰ باید بعنوان مقدار پارامتر وارد شود. مثالی کوتاه:

First Method:

```
.386
.model flat, stdcall
.................
.const
IDB_MYBITMAP    equ 100
.................
.data?
hInstance  dd ?
..............
.code
..............
   invoke GetModuleHandle,NULL
   mov hInstance,eax
.............
   invoke LoadBitmap,hInstance,IDB_MYBITMAP
...........
```

Second Method:

```
.386
.model flat, stdcall
.................
.data
BitmapName  db "MyBitMap",0
.................
.data?
hInstance  dd ?
..............
.code
..............
   invoke GetModuleHandle,NULL
   mov hInstance,eax
.............
   invoke LoadBitmap,hInstance,addr BitmapName
...........
```

۲. دریافت دستگیره بافت ابزار (DC) . این دستگیره با فراخوانی BeginPaint در پاسخ به پیغام WM_PAINT با فراخوانی GetDC بدست می‌آید.

www.mohandesyar.com

۳. ایجاد بافت افزاری در حافظه مشابه با خواص بافت ابزار قسمت ۲ . از این بافت ابزار بعنوان سطح ترسیمی مخفی جهت ترسیم بیت مپ بر روی آن استفاده خواهیم کرد.سپس از آن برای ترسیم بیت مپ بر روی پنجره برنامه استفاده خواهیم کرد. به این تکنیک بافر دوگانه (double-buffer) نیز گفته می‌شود. این سطح ترسیمی مخفی با فراخوانی تابع CreateCompatibleDC ایجاد می گردد. در صورت موفقیت آمیز بودن فراخوانی این تابع، خروجی تابع، دستگیره ای به بافت ابزار ایجاد شده در حافظه خواهد بود.

```
CreateCompatibleDC proto hdc:HDC
```

۴. در ادامه با فراخوانی تابع SelectObject ، عملیات ترسیم بیت مپ بر روی این سطح مخفی انجام می‌شود. پارامتر اول آن دستگیره‌ای است به بافت ابزار ایجاد شده در حافظه و پارامتر دوم آن دستگیره بیت مپ است.

```
SelectObject proto hdc:HDC, hGdiObject:DWORD
```

۵. سپس تصویر ترسیم شده در سطح مخفی ایجاد شده در حافظه باید بر روی پنجره برنامه کپی شود. برای انجام اینکار از توابعی مانند BitBlt و StretchBlt می‌توان استفاده کرد. تابع BitBlt به صورت زیر تعریف می گردد:

```
BitBlt proto hdcDest:DWORD, nxDest:DWORD,\
        nyDest:DWORD, nWidth:DWORD, nHeight:DWORD,\
        hdcSrc:DWORD, nxSrc:DWORD, nySrc:DWORD, dwROP:DWORD
```

hdcDest : دستگیره بافت ابزاری است که تصویر به آنجا کپی خواهد شد.

nxDest, nyDest : مختصات بالا سمت چپ ناحیه نمایش است.

nWidth, nHeight:طول و عرض ناحیه نمایش است.

hdcSrc : دستگیره بافت ابزاری است که تصویر از آنجا کپی خواهد شد.

nxSrc, nySrc : مختصات بالا سمت چپ ناحیه مبداء در حافظه است.

dwROP : raster-operation code . نحوه ترکیب رنگ‌های مبدا و مقصد را مشخص می کند.

۶. پس از ترسیم، برای آزاد سازی منابع تخصیص داده شده از DeleteObject استفاده می گردد.

www.mohandesyar.com

کد مثال ۲۲:

```
.386
.model flat,stdcall
option casemap:none
include \masm32\include\windows.inc
include \masm32\include\user32.inc
include \masm32\include\kernel32.inc
include \masm32\include\gdi32.inc
includelib \masm32\lib\user32.lib
includelib \masm32\lib\kernel32.lib
includelib \masm32\lib\gdi32.lib

WinMain proto :DWORD,:DWORD,:DWORD,:DWORD
IDB_MAIN   equ 1

.data
ClassName db "SimpleWin32ASMBitmapClass",0
AppName  db "Win32ASM Simple Bitmap Example",0

.data?
hInstance HINSTANCE ?
CommandLine LPSTR ?
hBitmap dd ?

.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  hInstance
 pop   wc.hInstance
 mov   wc.hbrBackground,COLOR_WINDOW+1
 mov   wc.lpszMenuName,NULL
 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,NULL,ADDR ClassName,ADDR AppName,\
           WS_OVERLAPPEDWINDOW,CW_USEDEFAULT,\
           CW_USEDEFAULT,CW_USEDEFAULT,CW_USEDEFAULT,NULL,NULL,\
           hInst,NULL
 mov   hwnd,eax
```


www.mohandesyar.com

```
 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
   LOCAL ps:PAINTSTRUCT
   LOCAL hdc:HDC
   LOCAL hMemDC:HDC
   LOCAL rect:RECT
   .if uMsg==WM_CREATE
      invoke LoadBitmap,hInstance,IDB_MAIN
      mov hBitmap,eax
   .elseif uMsg==WM_PAINT
      invoke BeginPaint,hWnd,addr ps
      mov    hdc,eax
      invoke CreateCompatibleDC,hdc
      mov    hMemDC,eax
      invoke SelectObject,hMemDC,hBitmap
      invoke GetClientRect,hWnd,addr rect
      invoke BitBlt,hdc,0,0,rect.right,rect.bottom,hMemDC,0,0,SRCCOPY
      invoke DeleteDC,hMemDC
      invoke EndPaint,hWnd,addr ps
 .elseif uMsg==WM_DESTROY
  invoke DeleteObject,hBitmap
  invoke PostQuitMessage,NULL
 .ELSE
  invoke DefWindowProc,hWnd,uMsg,wParam,lParam
  ret
 .ENDIF
 xor eax,eax
 ret
WndProc endp
end start

;-----------------------------------------------------------------------
;                             The resource script
;-----------------------------------------------------------------------
#define IDB_MAIN 1
IDB_MAIN BITMAP "tweety78.bmp"
```

بررسی کد فوق:

```
#define IDB_MAIN 1
IDB_MAIN BITMAP "tweety78.bmp"
```


www.mohandesyar.com

ثابتی را به نام **IDB_MAIN** تعریف کرده و سپس عدد یک را به آن نسبت می دهیم. سپس از آن بعنوان شناسه بیت‌مپ تعریف شده در فایل ریسورس، استفاده خواهیم کرد. در اینجا چون مسیر فایل مشخص نشده است بنابراین فرض می‌شود که فایل بیت مپ در دایرکتوری جاری فایل ریسورس قرار دارد. سپس :

```
.if uMsg==WM_CREATE
   invoke LoadBitmap,hInstance,IDB_MAIN
   mov hBitmap,eax
```

در پاسخ به پیغام **WM_CREATE** ، فایل بیت مپ را با فراخوانی تابع **LoadBitmap** ، بارگذاری خواهیم کرد. در ادامه :

```
.elseif uMsg==WM_PAINT
   invoke BeginPaint,hWnd,addr ps
   mov    hdc,eax
   invoke CreateCompatibleDC,hdc
   mov    hMemDC,eax
   invoke SelectObject,hMemDC,hBitmap
   invoke GetClientRect,hWnd,addr rect
   invoke BitBlt,hdc,0,0,rect.right,rect.bottom,hMemDC,0,0,SRCCOPY
   invoke DeleteDC,hMemDC
   invoke EndPaint,hWnd,addr ps
```

در پاسخ به پیغام WM_PAINT ، بیت مپ را ترسیم خواهیم کرد. در ابتدا با فراخوانی BeginPaint ، دستگیره‌ی بافت ابزار بدست می‌آید. سپس بافت ابزاری در حافظه توسط CreateCompatibleDC ایجاد خواهد شد. در ادامه توسط فراخوانی SelectObject ، بیت مپ به این بافت ابزار منتقل می‌گردد.

ابعاد ناحیه کلاینت برای ترسیم توسط تابع GetClientRect بدست خواهد آمد. در ادامه با کمک تابع BitBlt ، تصویر ترسیم شده در حافظه به پنجره برنامه منتقل می‌گردد. پس از ترسیم، دیگر نیاز به بافت ابزار ایجاد شده در حافظه نبوده و توسط DeleteDC حذف خواهد شد. در نهایت EndPaint فراخوانی می‌گردد.سپس:

```
.elseif uMsg==WM_DESTROY
 invoke DeleteObject,hBitmap
 invoke PostQuitMessage,NULL
```

در آخر، دیگر نیاز به بیت مپ نبوده و توسط DeleteObject حذف خواهد شد.

www.mohandesyar.com

مثال ۲۳- Splash Screen

Splash screen ، عموما پنجره ای است حاوی یک تصویر، بدون نوار عنوان، منوی سیستمی و حاشیه اطراف آن ، که برای مدتی مشخص ، در ابتدای اجرا یک برنامه ، این تصویر را نمایش می‌دهد و سپس محو خواهد شد.از آن برای نمایش لوگوی برنامه و یا انجام یک سری کارهای مقدماتی در پس زمینه برنامه در حال نمایش این تصویر، استفاده می گردد.

قدم اول برای ایجاد این پنجره ، الحاق کردن یک فایل بیت مپ به فایل ریسورس برنامه است. اگر به عملیات مورد نظر با دقت بنگریم، درخواهیم یافت که اینکار صرفا برای نمایش یک بیت مپ به مدت چند لحظه در ابتدای اجرا برنامه و نگهداری آن در حافظه تا پایان اجرای برنامه ، اتلاف منابع است. روش بهتر، ایجاد یک DLL *resource* می‌باشد که حاوی فایل بیت مپ مورد نظر بوده و بارگذاری و یا خارج ساختن آن از حافظه، در لحظه ای میسر می‌باشد.بنابراین روش عمومی، به صورت زیر خواهد بود:

۱. قرار دادن فایل بیت مپ در یک dll ، بعنوان ریسورس آن.
۲. برنامه اصلی با فراخوانی LoadLibrary ، dll را در حافظه بارگذاری می‌نماید.
۳. تابع entrypoint مربوط به dll فراخوانی می گردد.در این حالت یک تایمر با طولی مشخص ایجاد و سپس کلاس پنجره جدید را بدون عنوان ، حاشیه و غیره، رجیستر می‌نماید.سپس فایل بیت مپ را در این پنجره نمایش می‌دهیم.
۴. پس از پایان زمان در نظر گرفته شده، splash screen از صفحه حذف شده و کنترل به برنامه اصلی منتقل خواهد شد.
۵. برنامه اصلی با فراخوانی FreeLibrary ، dll را از حافظه خارج می‌سازد.

www.mohandesyar.com

جزئیات کار به شرح زیر است:

یک dll را به صورت پویا با فراخوانی تابع LoadLibrary proto lpDLLName:DWORD می‌توان بارگذاری نمود.این تابع، آدرس نام dll را بعنوان ورودی می‌پذیرد.اگر فراخوانی موفقیت آمیز باشد، دستگیره ماژول DLL بازگشت داده خواهد شد، در غیر اینصورت خروجی تابع NULL خواهد بود.

برای از حافظه خارج کردن dll ، می‌توان از FreeLibrary proto hLib:DWORD استفاده کرد.پارامتر ورودی این تابع دستگیره ماژول dll مورد نظر است.

طرز استفاده از تایمر:

با استفاده از تابع SetTimer (به شرح زیر) یک تایمر ایجاد می‌شود:

```
SetTimer proto hWnd:DWORD, TimerID:DWORD, uElapse:DWORD, lpTimerFunc:DWORD
```

hWnd : دستگیره پنجره ای است که پیغامهای تایمر را دریافت خواهد کرد. اگر مساوی NULL قرار گیرد، به این معنا است که این تایمر به هیچ پنجره ای وابسته نخواهد بود.

TimerID : مقداری است تعریف شده توسط کاربر، بعنوان ID تایمر.

uElapse : مدت زمان منقضی شدن تایمر برحسب میلی ثانیه.

lpTimerFunc : آدرس تابعی است که پیغامهای تایمر را پردازش می‌نماید.اگر این پارامتر مساوی نال قرار گیرد، پیغامها به پنجره مشخص شده توسط hWnd ارسال می گردد.

در صورت فراحوانی موفقیت آمیز این تابع، ID تایمر بازگشت داده خواهد شد. بنابراین بهتر است هیچگاه ID تایمر را مساوی صفر قرار ندهید.

یک تایمر را به دو صورت می‌توان ایجاد کرد:

۱. اگر قرار است پنجره برنامه، پیغامهای تایمر را دریافت کند، باید آخرین پارامتر آن مساوی نال قرار گیرد.
۲. اگر پنجره ای وجود نداشته باشد و یا نیازی به پردازش پیغامهای تایمر در رویه پنجره نباشد، باید اولین پارامتر تابع فوق را مساوی نال قرار داد.

www.mohandesyar.com

ما در این مثال از روش اول استفاده خواهیم کرد. هنگامیکه زمان منقضی شدن تایمر فرا رسد، پیغام WM_TIMER به رویه پنجره ارسال خواهد شد. برای مثال اگر uElapse را مساوی ۱۰۰۰ قرار دهید، پیغام WM_TIMER هر ثانیه یکبار به رویه پنجره ارسال می‌گردد. زمانیکه به تایمر نیاز نباشد آنرا می‌توان با KillTimer از بین برد.

کد مثال ۲۳:

```
;-------------------------------------------------------------------------
;                          The main program
;-------------------------------------------------------------------------
.386
.model flat,stdcall
option casemap:none
include \masm32\include\windows.inc
include \masm32\include\user32.inc
include \masm32\include\kernel32.inc
includelib \masm32\lib\user32.lib
includelib \masm32\lib\kernel32.lib

WinMain proto :DWORD,:DWORD,:DWORD,:DWORD

.data
ClassName db "SplashDemoWinClass",0
AppName  db "Splash Screen Example",0
Libname db "splash.dll",0

.data?
hInstance HINSTANCE ?
CommandLine LPSTR ?
.code
start:
 invoke LoadLibrary,addr Libname
 .if eax!=NULL
    invoke FreeLibrary,eax
 .endif
 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  hInstance
 pop   wc.hInstance
 mov   wc.hbrBackground,COLOR_WINDOW+1
```


www.mohandesyar.com

```
 mov   wc.lpszMenuName,NULL
 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,NULL,ADDR ClassName,ADDR AppName,\
           WS_OVERLAPPEDWINDOW,CW_USEDEFAULT,\
           CW_USEDEFAULT,CW_USEDEFAULT,CW_USEDEFAULT,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
 .ELSE
  invoke DefWindowProc,hWnd,uMsg,wParam,lParam
  ret
 .ENDIF
 xor eax,eax
 ret
WndProc endp
end start

;-------------------------------------------------------------------
;                          The Bitmap DLL
;-------------------------------------------------------------------
.386
.model flat, stdcall
include \masm32\include\windows.inc
include \masm32\include\user32.inc
include \masm32\include\kernel32.inc
include \masm32\include\gdi32.inc
includelib \masm32\lib\user32.lib
includelib \masm32\lib\kernel32.lib
includelib \masm32\lib\gdi32.lib
.data
BitmapName db "MySplashBMP",0
ClassName db "SplashWndClass",0
hBitMap dd 0
TimerID dd 0

.data
hInstance dd ?

.code
```


www.mohandesyar.com

```
DllEntry proc hInst:DWORD, reason:DWORD, reserved1:DWORD
   .if reason==DLL_PROCESS_ATTACH  ; When the dll is loaded
      push hInst
      pop hInstance
      call ShowBitMap
   .endif
   mov eax,TRUE
   ret
DllEntry Endp
ShowBitMap proc
        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  hInstance
        pop   wc.hInstance
        mov   wc.hbrBackground,COLOR_WINDOW+1
        mov   wc.lpszMenuName,NULL
        mov   wc.lpszClassName,OFFSET ClassName
        invoke LoadIcon,NULL,IDI_APPLICATION
        mov   wc.hIcon,eax
        mov   wc.hIconSm,0
        invoke LoadCursor,NULL,IDC_ARROW
        mov   wc.hCursor,eax
        invoke RegisterClassEx, addr wc
        INVOKE CreateWindowEx,NULL,ADDR ClassName,NULL,\
           WS_POPUP,CW_USEDEFAULT,\
           CW_USEDEFAULT,250,250,NULL,NULL,\
           hInstance,NULL
        mov   hwnd,eax
        INVOKE ShowWindow, hwnd,SW_SHOWNORMAL
        .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
ShowBitMap endp
WndProc proc hWnd:DWORD,uMsg:DWORD,wParam:DWORD,lParam:DWORD
        LOCAL ps:PAINTSTRUCT
        LOCAL hdc:HDC
        LOCAL hMemoryDC:HDC
        LOCAL hOldBmp:DWORD
        LOCAL bitmap:BITMAP
        LOCAL DlgHeight:DWORD
        LOCAL DlgWidth:DWORD
        LOCAL DlgRect:RECT
        LOCAL DesktopRect:RECT

        .if uMsg==WM_DESTROY
                .if hBitMap!=0
                        invoke DeleteObject,hBitMap
                .endif
                invoke PostQuitMessage,NULL
        .elseif uMsg==WM_CREATE
                invoke GetWindowRect,hWnd,addr DlgRect
```


www.mohandesyar.com

```
                invoke GetDesktopWindow
                mov ecx,eax
                invoke GetWindowRect,ecx,addr DesktopRect
                push  0
                mov  eax,DlgRect.bottom
                sub  eax,DlgRect.top
                mov  DlgHeight,eax
                push eax
                mov  eax,DlgRect.right
                sub  eax,DlgRect.left
                mov  DlgWidth,eax
                push eax
                mov  eax,DesktopRect.bottom
                sub  eax,DlgHeight
                shr  eax,1
                push eax
                mov  eax,DesktopRect.right
                sub  eax,DlgWidth
                shr  eax,1
                push eax
                push hWnd
                call MoveWindow
                invoke LoadBitmap,hInstance,addr BitmapName
                mov hBitMap,eax
                invoke SetTimer,hWnd,1,2000,NULL
                mov TimerID,eax
        .elseif uMsg==WM_TIMER
                invoke SendMessage,hWnd,WM_LBUTTONDOWN,NULL,NULL
                invoke KillTimer,hWnd,TimerID
        .elseif uMsg==WM_PAINT
                invoke BeginPaint,hWnd,addr ps
                mov hdc,eax
                invoke CreateCompatibleDC,hdc
                mov hMemoryDC,eax
                invoke SelectObject,eax,hBitMap
                mov hOldBmp,eax
                invoke GetObject,hBitMap,sizeof BITMAP,addr bitmap
                invoke StretchBlt,hdc,0,0,250,250,\
                       hMemoryDC,0,0,bitmap.bmWidth,bitmap.bmHeight,SRCCOPY
                invoke SelectObject,hMemoryDC,hOldBmp
                invoke DeleteDC,hMemoryDC
                invoke EndPaint,hWnd,addr ps
        .elseif uMsg==WM_LBUTTONDOWN
                invoke DestroyWindow,hWnd
        .else
                invoke DefWindowProc,hWnd,uMsg,wParam,lParam
                ret
        .endif
        xor eax,eax
        ret
WndProc endp

End DllEntry
```

بررسی کد فوق:

در ابتدا کد برنامه اصلی را بررسی خواهیم کرد.

www.mohandesyar.com

```
invoke LoadLibrary,addr Libname
.if eax!=NULL
   invoke FreeLibrary,eax
.endif
```

از تابع LoadLibrary جهت بارگذاری dll ایی به نام "splash.dll" استفاده خواهیم کرد.کار این تابع تا زمان انجام مقداردهی‌های اولیه در dll ، به پایان نخواهد رسید. در ادامه :

```
.if reason==DLL_PROCESS_ATTACH  ; When the dll is loaded
   push hInst
   pop hInstance
   call ShowBitMap
```

هنگامیکه dll فراخوانی می‌شود، ویندوز تابع entrypoint آنرا به همراه پرچم DLL_PROCESS_ATTACH فراخوانی خواهد کرد.ما از این موقعیت برای نمایش splash screen استفاده خواهیم کرد.

در ابتدا دستگیره وهله dll را برای استفاده بعدی ذخیره خواهیم کرد. سپس تابع ShowBitMap را جهت انجام کار اصلی (ایجاد پنجره) فراخوانی می‌نمائیم. قسمت جالب کد آن ، نحوه فراخوانی تابع زیر است:

```
INVOKE CreateWindowEx,NULL,ADDR ClassName,NULL,\
   WS_POPUP,CW_USEDEFAULT,\
   CW_USEDEFAULT,250,250,NULL,NULL,\
   hInstance,NULL
```

سبک پنجره WS_POPUP ، پنجره ای بدون حاشیه و نوار عنوان ایجاد می‌نماید.همچنین طول و عرض پنجره را نیز به 250x250 pixels محدود کرده ایم. در زمانیکه پنجره در حال ایجاد است ، با پردازش پیغام WM_CREATE ، پنجره را به میانه صفحه نمایش منتقل خواهیم کرد :

```
invoke GetWindowRect,hWnd,addr DlgRect
invoke GetDesktopWindow
mov ecx,eax
invoke GetWindowRect,ecx,addr DesktopRect
push  0
mov  eax,DlgRect.bottom
sub  eax,DlgRect.top
mov  DlgHeight,eax
push eax
mov  eax,DlgRect.right
sub  eax,DlgRect.left
mov  DlgWidth,eax
push eax
mov  eax,DesktopRect.bottom
sub  eax,DlgHeight
shr  eax,1
push eax
mov  eax,DesktopRect.right
sub  eax,DlgWidth
shr  eax,1
push eax
```


www.mohandesyar.com

```
push hWnd
call MoveWindow
```

در ادامه :

```
invoke LoadBitmap,hInstance,addr BitmapName
mov hBitMap,eax
invoke SetTimer,hWnd,1,2000,NULL
mov TimerID,eax
```

بیت مپ از ریسورس dll ، بارگذاری شده و سپس تایمری با ID مساوی یک و فواصل زمانی ارسال پیغام های آن به مدت ۲ ثانیه ، ایجاد می گردد. در ادامه :

```
.elseif uMsg==WM_PAINT
        invoke BeginPaint,hWnd,addr ps
        mov hdc,eax
        invoke CreateCompatibleDC,hdc
        mov hMemoryDC,eax
        invoke SelectObject,eax,hBitMap
        mov hOldBmp,eax
        invoke GetObject,hBitMap,sizeof BITMAP,addr bitmap
        invoke StretchBlt,hdc,0,0,250,250,\
               hMemoryDC,0,0,bitmap.bmWidth,bitmap.bmHeight,SRCCOPY
        invoke SelectObject,hMemoryDC,hOldBmp
        invoke DeleteDC,hMemoryDC
        invoke EndPaint,hWnd,addr ps
```

زمانیکه ویندوز پیغام WM_PAINT را دریافت می کند، بافت ابزاری را در حافظه ایجاد ، بیت مپ را به آن منتقل ، اندازه بیت مپ را با کمک تابع GetObject بدست آورده و بیت مپ را با فراخوانی StretchBlt که توانایی پرکردن تمامی پنجره را با بزرگ کردن اندازه تصویر دارد، بر روی پنجره ترسیم خواهیم کرد. در آخر بافت ابزار ایجاد شده در حافظه را تخریب می کنیم. در ادامه :

```
.elseif uMsg==WM_LBUTTONDOWN
        invoke DestroyWindow,hWnd
```

اگر کاربر بر روی تصویر کلیک نماید، پنجره محو خواهد شد. سپس :

```
.elseif uMsg==WM_TIMER
        invoke SendMessage,hWnd,WM_LBUTTONDOWN,NULL,NULL
        invoke KillTimer,hWnd,TimerID
```

اگر کاربر بر روی تصویر کلیک ننماید، پس از ۲ ثانیه (به همراه پیغام WM_TIMER) ، پنجره نمایش یافته محو خواهد شد. (جهت جلوگیری از دوباره نویسی کد، از تابع SendMessage استفاده شد)

www.mohandesyar.com

مثال ۲۴- Tooltip Control

Tooltip ، ناحیه مستطیلی کوچکی است که با نگه داشتن اشاره‌گر ماوس روی شیءایی، جهت ارائه توضیحاتی در مورد آن نمایان می‌شود. پس از دور شدن اشاره‌گر ماوس از شیء ذکر شده، این ناحیه مستطیلی محو خواهد شد.

روش ایجاد یک tooltip به شرح زیر است:

۱. ایجاد کنترل tooltip با فراخوانی CreateWindowEx
۲. تعریف ناحیه‌ای برای نظارت کردن بر حرکات اشاره‌گر ماوس
۳. باخبرسازی کنترل از ناحیه فوق
۴. ارسال پیغامهای ماوس در ناحیه نظارتی به کنترل

در ادامه این موارد را به تفصیل بررسی خواهیم کرد:

tooltip یکی از common controls است. بنابراین باید در قسمتی دلخواه از کد، فراخوانی InitCommonControls ، جهت بارگذاری ضمنی comctl32.dll توسط MASM ، صورت گیرد. این کنترل با فراخوانی CreateWindowEx ایجاد می‌گردد.

```
.data
TooltipClassName db "Tooltips_class32",0
.code
.....
invoke InitCommonControls
invoke CreateWindowEx, NULL, addr TooltipClassName, NULL,\
                       TIS_ALWAYSTIP, CW_USEDEFAULT, CW_USEDEFAULT,\
                       CW_USEDEFAULT, CW_USEDEFAULT, NULL, NULL, hInstance, NULL
```

سبک پنجره TIS_ALWAYSTIP ، باعث می‌شود حتی اگر ناحیه تعریف شده جهت نمایش tooltip ، غیرفعال هم باشد، بازهم tooltip نمایش یابد. همچنین نیازی در بکار بردن WS_POPUP and WS_EX_TOOLWINDOW نیست، زیرا رویه پنجره کنترل tooltip ، آنها را به صورت خودکار اضافه خواهد نمود.

نیازی به ذکر مختصات نمایش و یا طول و عرض پنجره tooltip نیز نمی‌باشد. این کنترل، موارد ذکر شده را بصورت خودکار تنظیم خواهد کرد. بنابراین در هر چهار پارامتر مذکور از CW_USEDEFAULT استفاده گردید.

www.mohandesyar.com

کنترل tooltip بلافاصله پس از ایجاد، نمایش داده نخواهد شد. نیاز است این پنجره تنها زمانی نمایش یابد که اشاره‌گر ماوس بر روی ناحیه مشخص شده‌ای قرار گیرد. اکنون زمان مشخص نمودن این ناحیه است. به این ناحیه در ادامه، tool خواهیم گفت و کار نظارت کردن بر حرکت اشاره‌گر ماوس را برعهده خواهد داشت. اگر اشاره گر ماوس بر روی tool قرار گرفت ، پنجره tooltip نمایش داده خواهد شد.

این ناحیه می‌تواند کل ناحیه کلاینت را پوشش داده و یا تنها بر روی قسمتی از آن تعریف گردد. بنابراین tool را به دو نوع می‌توان تقسیم کرد : نوع اول به صورت یک پنجره پیاده سازی خواهد شد و نوع دوم مستطیلی است تعریف شده در ناحیه کلاینت یک پنجره. حالت اول بیشتر با پنجره‌های کنترل‌هایی مانند دکمه‌ها، کنترل تکست باکس و امثال آن بکار می‌رود.در این حالت نیازی به تعیین مختصات و همچنین طول و عرض پنجره مربوطه نیست. حالت دوم، برای تقسیم بندی ناحیه کلاینت پنجره به چند قسمت، بدون استفاده از پنجره‌های کنترل ها کاربرد دارد. در این حالت باید مختصات سمت چپ بالای پنجره و همچنین طول و عرض آنرا مشخص کرد.

ساختار تعیین کننده نحوه نمایش و خواص یک tool ، به شرح زیر است:

```
TOOLINFO STRUCT
  cbSize              DWORD      ?
  uFlags              DWORD      ?
  hWnd                DWORD      ?
  uId                 DWORD      ?
  rect                RECT      <>
  hInst               DWORD      ?
  lpszText            DWORD      ?
  lParam              LPARAM     ?
TOOLINFO ENDS
```

cbSize : اندازه ساختار TOOLINFO است. این عضو حتما باید مقداردهی گردد. در غیراینصورت باید منتظر نتایج غیرمنتظره‌ای بود.

uFlags : پرچمی است تعیین کننده خواص ظاهری tool . ترکیبی از موارد زیر نیز می‌تواند باشد:

TTF_IDISHWND : (ID is hWnd) بدین معنا که tool ، کل ناحیه کلاینت را خواهد پوشاند (حالت اول ذکر شده در بالا). در این حالت باید عضو uId این ساختار را با hWnd پنجره کنترل مورد نظر مقدار دهی کرد. اگر نیاز به استفاده از حالت دوم باشد باید عضو rect ساختار را مقدار دهی نمود.

TTF_CENTERTIP : در این حالت tooltip در زیر ناحیه مشخص شده و در میانه آن نمایش خواهد یافت.

TTF_RTLREADING : برای نمایش فارسی، عربی و یا عبری مناسب است.

TTF_SUBCLASS : در این حالت کنترل tooltip ، برای subclassing رویه پنجره مربوطه جهت پردازش پیغامهای ماوس اقدام می‌نماید.

www.mohandesyar.com

hWnd : دستگیره پنجره حاوی tool است. اگر TTF_IDISHWND را پیشتر ذکر کرده باشید، از این فیلد صرفنظر خواهد شد. زیرا ویندوز دستگیره پنجره ذکر شده در عضو uId را استفاده خواهد نمود.این عضو تنها در موارد زیر باید مقدار دهی شود:

- هنگام استفاده از نواحی مستطیلی (حالت دوم) و استفاده نکردن از TTF_IDISHWND .
- مقدار دهی عضو lpszText با LPSTR_TEXTCALLBACK . در این حالت هنگام نمایش متن tooltip ، از برنامه در مورد متن مورد نظر را درخواست خواهد کرد (با ارسال پیغام TTN_NEEDTEXT به رویه پنجره). بنابراین در این حالت امکان تغییر پویای متن مهیا می‌باشد.

uId : در مورد مقدار و معنای این عضو پیشتر توضیح داده شد.

rect : ساختار RECT ، بیانگر ابعاد tool که ناحیه مشخصی از پنجره کلاینت را می‌پوشاند، می‌باشد. اگر پرچم TTF_IDISHWND تنظیم شود، از این عضو صرفنظر خواهد شد.

hInst : دستگیره وهله‌ای است که حاوی ریسورس رشته ای مورد استفاده در tooltip است.اگر عضو lpszText حاوی شناسه یک رشته در فایل ریسورس نباشد، از این عضو صرفنظر خواهد شد.توضیحات بیشتر در ادامه خواهد آمد.

lpszText : این عضو مقادیر متعددی می‌تواند داشته باشد:

- اگر مقدار این عضو را مساوی LPSTR_TEXTCALLBACK قرار دهیم، کنترل tooltip پیغام TTN_NEEDTEXT را به پنجره مشخص شده توسط عضو hWnd ، برای نمایش رشته متنی در tooltip ، ارسال خواهد کرد.به این صورت می‌توان رشته های نمایش داده شده در این کنترل را به صورت پویا مقدار دهی نمود.
- اگر مقدار این عضو را مساوی شناسه یک رشته در فایل ریسورس قرار دهید، برای نمایش متن ، string table مربوط به وهله مشخص شده با hInst ، جستجو خواهد شد. اینکار برای تعریف زبانهای متعدد بدون دستکاری سورس برنامه بسیار مفید است. شناسه رشته قرار گرفته در فایل ریسورس، از بررسی high word این عضو بدست می‌آید. از آنجائیکه این شناسه ۱۶ بیتی است، بنابراین high word آن همیشه صفر خواهد بود.

www.mohandesyar.com

- اگر مقدار این عضو LPSTR_TEXTCALLBACK نبوده و high word آن صفر نباشد، این عضو بعنوان اشاره‌گری به رشته مورد نظر برای نمایش بعنوان متن tooltip تفسیر خواهد شد. این روش بسیار ساده بوده اما انعطاف پذیر نیست.

بنابراین ساختار TOOLINFO پیش از ارسال آن به کنترل tooltip ، باید مقدار دهی شود. این ساختار خواص tool مورد نظر را بیان می‌کند.در ادامه باید این ساختار در اختیار کنترل قرار گیرد. از آنجائیکه یک کنترل tooltip توانایی سرویس دهی به tool های مختلفی را دارد، نیازی به تعریف چندین کنترل tooltip نمی‌باشد.برای رجیستر کردن یک tool در کنترل tooltip ، باید پیغام TTM_ADDTOOL را به کنترل ارسال نمود. در این حالت از wParam استفاده نشده و lParam حاوی آدرس TOOLINFO جهت رجیستر شدن ، خواهد بود.

```
.data?
ti TOOLINFO <>
.......
.code
.......
<fill the TOOLINFO structure>
.......
invoke SendMessage, hwndTooltip, TTM_ADDTOOL, NULL, addr ti
```

در صورت رجیستر شدن موفقیت آمیز ساختار در کنترل tooltip ، خروجی تابع SendMessage ، TRUE خواهد بود. جهت انجام عملیات عکس، از پیغام TTM_DELTOOL استفاده می‌گردد.

در این مرحله، کنترل tooltip می‌داند که چه ناحیه ای را باید تحت نظارت قرار داد و چه متنی را نمایش دهد.
برای پردازش پیغام‌های ماوس در کنترل tooltip ، مدت زمان توقف اشاره‌گر ماوس بر روی ناحیه نظارتی، اندازه گیری شده و سپس نسبت به آن در جهت نمایش tooltip اقدام خواهد شد. دو روش برای انجام این مقصود وجود دارند: با کمک پنجره حاوی tool و یا بدون کمک آن.

- ارسال پیغام های TTM_RELAYEVENT به کنترل tooltip توسط پنجره حاوی tool . در این حالت lParam پیغام حاوی آدرس ساختار MSG است. این ساختار بیانگر پیغامی است که به کنترل ارسال می‌گردد. یک کنترل tooltip تنها پیغامهای زیر را پردازش می‌کند:

WM_LBUTTONDOWN
WM_MOUSEMOVE
WM_LBUTTONUP
WM_RBUTTONDOWN
WM_MBUTTONDOWN
WM_RBUTTONUP
WM_MBUTTONUP

www.mohandesyar.com

از سایر پیغامها صرفنظر خواهد شد. بنابراین رویه پنجره شبیه به عبارات زیر خواهد بود:

```
WndProc proc hWnd:DWORD, uMsg:DWORD, wParam:DWORD, lParam:DWORD
.....
   if uMsg==WM_CREATE
      ..............
   elseif uMsg==WM_LBUTTONDOWN || uMsg==WM_MOUSEMOVE || uMsg==WM_LBUTTONUP ||
uMsg==WM_RBUTTONDOWN || uMsg==WM_MBUTTONDOWN || uMsg==WM_RBUTTONUP ||
uMsg==WM_MBUTTONUP
      invoke SendMessage, hwndTooltip, TTM_RELAYEVENT, NULL, addr msg
      ..........
```

- با مشخص کردن پرچم TTF_SUBCLASS در عضو uFlags ساختار TOOLINFO، سبب subclassing پنجره حاوی tool گشته و بدون نیاز به کمک پنجره آن، پیغامهای ماوس را پردازش خواهد کرد. این روش ساده‌تر بوده و نیاز به کد نویسی کمتری دارد.

تا اینجا کار ایجاد و نمایش کنترل tooltip به پایان می‌رسد. در ادامه با یک سری از پیغام‌های وابسته به کنترل tooltip آشنا خواهیم شد:

TTM_ACTIVATE: برای فعال و یا غیرفعال کردن کنترل tooltip به صورت پویا می‌توان از این پیغام کمک گرفت. اگر wParam مساوی TRUE باشد، کنترل فعال خواهد شد و برعکس. کنترل tooltip در اولین بار ایجاد آن فعال می‌باشد، بنابراین نیازی به ارسال این پیغام نخواهد بود.

TTM_GETTOOLINFO and TTM_SETTOOLINFO: برای دریافت و یا تغییر پویای مقادیر اعضای ساختار TOOLINFO، پس از ارسال آن به کنترل tooltip، از این پیغام استفاده می‌شود.این کنترل جهت تغییر با مقادیر uId و hWnd مشخص می‌گردد. اگر نیاز به تغییر عضو rect باشد، از پیغام TTM_NEWTOOLRECT استفاده خواهد شد. برای تغییر متن tooltip از پیغام TTM_UPDATETIPTEXT استفاده می‌شود.

TTM_SETDELAYTIME: مدت زمان نمایش متن tooltip را مشخص می‌نماید.

www.mohandesyar.com

کد مثال ۲۴ :

در این مثال، پنجره کلاینت به چهار قسمت تقسیم گردیده و به هر قسمت یک tool انتساب داده شد. همچنین دو کنترل دکمه قرار گرفته روی فرم نیز کنترل های tooltip خاص خود را دارند.

```
.386
.model flat,stdcall
option casemap:none
include \masm32\include\windows.inc
include \masm32\include\kernel32.inc
include \masm32\include\user32.inc
include \masm32\include\comctl32.inc
includelib \masm32\lib\comctl32.lib
includelib \masm32\lib\user32.lib
includelib \masm32\lib\kernel32.lib
DlgProc proto :DWORD,:DWORD,:DWORD,:DWORD
EnumChild proto :DWORD,:DWORD
SetDlgToolArea proto :DWORD,:DWORD,:DWORD,:DWORD,:DWORD
.const
IDD_MAINDIALOG equ 101
.data
ToolTipsClassName db "Tooltips_class32",0
MainDialogText1 db "This is the upper left area of the dialog",0
MainDialogText2 db "This is the upper right area of the dialog",0
MainDialogText3 db "This is the lower left area of the dialog",0
MainDialogText4 db "This is the lower right area of the dialog",0
.data?
hwndTool dd ?
hInstance dd ?
.code
start:
    invoke GetModuleHandle,NULL
    mov hInstance,eax
    invoke DialogBoxParam,hInstance,IDD_MAINDIALOG,NULL,addr DlgProc,NULL
    invoke ExitProcess,eax

DlgProc proc hDlg:DWORD,uMsg:DWORD,wParam:DWORD,lParam:DWORD
    LOCAL ti:TOOLINFO
    LOCAL id:DWORD
    LOCAL rect:RECT
    .if uMsg==WM_INITDIALOG
        invoke InitCommonControls
        invoke CreateWindowEx,NULL,ADDR ToolTipsClassName,NULL,\
            TTS_ALWAYSTIP,CW_USEDEFAULT,\
            CW_USEDEFAULT,CW_USEDEFAULT,CW_USEDEFAULT,NULL,NULL,\
            hInstance,NULL
        mov hwndTool,eax
        mov id,0
        mov ti.cbSize,sizeof TOOLINFO
        mov ti.uFlags,TTF_SUBCLASS
        push hDlg
        pop ti.hWnd
        invoke GetWindowRect,hDlg,addr rect
        invoke SetDlgToolArea,hDlg,addr ti,addr MainDialogText1,id,addr rect
        inc id
        invoke SetDlgToolArea,hDlg,addr ti,addr MainDialogText2,id,addr rect
        inc id
```


www.mohandesyar.com

```
        invoke SetDlgToolArea,hDlg,addr ti,addr MainDialogText3,id,addr rect
        inc id
        invoke SetDlgToolArea,hDlg,addr ti,addr MainDialogText4,id,addr rect
        invoke EnumChildWindows,hDlg,addr EnumChild,addr ti
    .elseif uMsg==WM_CLOSE
        invoke EndDialog,hDlg,NULL
    .else
        mov eax,FALSE
        ret
    .endif
    mov eax,TRUE
    ret
DlgProc endp

EnumChild proc uses edi hwndChild:DWORD,lParam:DWORD
    LOCAL buffer[256]:BYTE
    mov edi,lParam
    assume edi:ptr TOOLINFO
    push hwndChild
    pop [edi].uId
    or [edi].uFlags,TTF_IDISHWND
    invoke GetWindowText,hwndChild,addr buffer,255
    lea eax,buffer
    mov [edi].lpszText,eax
    invoke SendMessage,hwndTool,TTM_ADDTOOL,NULL,edi
    assume edi:nothing
    ret
EnumChild endp

SetDlgToolArea proc uses edi esi
hDlg:DWORD,lpti:DWORD,lpText:DWORD,id:DWORD,lprect:DWORD
    mov edi,lpti
    mov esi,lprect
    assume esi:ptr RECT
    assume edi:ptr TOOLINFO
    .if id==0
        mov [edi].rect.left,0
        mov [edi].rect.top,0
        mov eax,[esi].right
        sub eax,[esi].left
        shr eax,1
        mov [edi].rect.right,eax
        mov eax,[esi].bottom
        sub eax,[esi].top
        shr eax,1
        mov [edi].rect.bottom,eax
    .elseif id==1
        mov eax,[esi].right
        sub eax,[esi].left
        shr eax,1
        inc eax
        mov [edi].rect.left,eax
        mov [edi].rect.top,0
        mov eax,[esi].right
        sub eax,[esi].left
        mov [edi].rect.right,eax
        mov eax,[esi].bottom
        sub eax,[esi].top
        mov [edi].rect.bottom,eax
    .elseif id==2
        mov [edi].rect.left,0
```


www.mohandesyar.com

```
        mov eax,[esi].bottom
        sub eax,[esi].top
        shr eax,1
        inc eax
        mov [edi].rect.top,eax
        mov eax,[esi].right
        sub eax,[esi].left
        shr eax,1
        mov [edi].rect.right,eax
        mov eax,[esi].bottom
        sub eax,[esi].top
        mov [edi].rect.bottom,eax
    .else
        mov eax,[esi].right
        sub eax,[esi].left
        shr eax,1
        inc eax
        mov [edi].rect.left,eax
        mov eax,[esi].bottom
        sub eax,[esi].top
        shr eax,1
        inc eax
        mov [edi].rect.top,eax
        mov eax,[esi].right
        sub eax,[esi].left
        mov [edi].rect.right,eax
        mov eax,[esi].bottom
        sub eax,[esi].top
        mov [edi].rect.bottom,eax
    .endif
    push lpText
    pop [edi].lpszText
    invoke SendMessage,hwndTool,TTM_ADDTOOL,NULL,lpti
    assume edi:nothing
    assume esi:nothing
    ret
SetDlgToolArea endp
end start
```

آنالیز کد فوق:

پس از ایجاد پنجره اصلی، کنترل tooltip با فراخوانی تابع CreateWindowEx ایجاد می‌شود.

```
invoke InitCommonControls
invoke CreateWindowEx,NULL,ADDR ToolTipsClassName,NULL,\
       TTS_ALWAYSTIP,CW_USEDEFAULT,\
       CW_USEDEFAULT,CW_USEDEFAULT,CW_USEDEFAULT,NULL,NULL,\
       hInstance,NULL
mov hwndTool,eax
```

سپس چهار tool جهت چهار ناحیه پنجره اصلی، ایجاد خواهیم کرد.

```
    mov id,0        ; used as the tool ID
    mov ti.cbSize,sizeof TOOLINFO
    mov ti.uFlags,TTF_SUBCLASS
```


www.mohandesyar.com

```
 ; tell the tooltip control to subclass the dialog window.
push hDlg
pop ti.hWnd    ; handle to the window that contains the tool
invoke GetWindowRect,hDlg,addr rect
               ; obtain the dimension of the client area
invoke SetDlgToolArea,hDlg,addr ti,addr MainDialogText1,id,addr rect
```

در ادامه اعضای TOOLINFO را مقدار دهی خواهیم کرد. از آنجائیکه نیاز است تا پنجره اصلی را به چهار ناحیه تقسیم کنیم، نیاز است تا ابعاد پنجره مشخص شوند. به همین جهت تابع GetWindowRect فراخوانی خواهد شد.همچنین جهت استفاده از مزیت های subclassing از پرچم TIF_SUBCLASS استفاده گردید. تابع SetDlgToolArea برای تنظیم ناحیه مستطیلی هر tool و رجیستر کردن tool به کنترل tooltip بکار می‌رود. سپس پیغام TTM_ADDTOOL به کنترل tooltip ارسال شده و آدرس ساختار TOOLINFO در lParam ، قرار می گیرد.

```
invoke SendMessage,hwndTool,TTM_ADDTOOL,NULL,lpti
```

پس از رجیستر کردن ۴ ناحیه مستطیلی، نوبت به کنترل‌های tooltip مربوط به دکمه ها می‌رسد. برای رجیستر کردن هر کنترل می‌توان از ID دکمه‌ها روی فرم استفاده کرد. اینکار نسبتا زمانبر و در حالت تعداد کنترل‌های زیاد، خسته کننده خواهد بود. به همین جهت از تابع EnumChildWindows جهت دستیابی سریع به کنترل‌های قرار گرفته بر روی فرم استفاده خواهیم کرد. تعریف این تابع به شرح زیر است:

EnumChildWindows proto hWnd:DWORD, lpEnumFunc:DWORD, lParam:DWORD

hWnd : دستگیره‌ای است به پنجره والد.

lpEnumFunc : آدرس تابع EnumChildProc که به ازای هر کنترل فراخوانی می گردد.

lParam : مقدار تعیین شده توسط برنامه که به تابع EnumChildProc ارسال خواهد شد. این تابع به صورت زیر تعریف می گردد:

EnumChildProc proto hwndChild:DWORD, lParam:DWORD

hwndChild : دستگیره کنترلی است که توسط تابع EnumChildWindows یکایک شمرده خواهد شد (enumerated) . برای مثال :

```
invoke EnumChildWindows,hDlg,addr EnumChild,addr ti
```

در اینجا آدرس ساختار TOOLINFO را در پارامتر lParam ، قرار خواهیم داد. زیرا هر کنترل قرار گرفته بر روی پنجره را به کنترل tooltip در تابع EnumChild رجیستر خواهیم کرد. در غیراینصورت باید ti را به صورت یک متغیر عمومی تعریف نمائیم و عموما متغیرهای عمومی سرمنشاء بسیاری از باگها می‌باشند.

www.mohandesyar.com

با فراخوانی EnumChildWindows ، ویندوز کنترل های قرار گرفته بر روی فرم را یک به یک شمارش کرده و به ازای هر یک کنترل، یکبار تابع EnumChild را فراخوانی خواهد کرد. توسط تابع EnumChild ، اعضای مرتبط TOOLINFO پر شده و tool را به کنترل tooltip رجیستر خواهد کرد.

```
EnumChild proc uses edi hwndChild:DWORD,lParam:DWORD
    LOCAL buffer[256]:BYTE
    mov edi,lParam
    assume edi:ptr TOOLINFO
    push hwndChild
    pop [edi].uId    ; we use the whole client area of the control as the tool
    or [edi].uFlags,TTF_IDISHWND
    invoke GetWindowText,hwndChild,addr buffer,255
    lea eax,buffer    ; use the window text as the tooltip text
    mov [edi].lpszText,eax
    invoke SendMessage,hwndTool,TTM_ADDTOOL,NULL,edi
    assume edi:nothing
    ret
EnumChild endp
```

لازم به ذکر است که در این حالت نوعی از tool که کل ناحیه کلاینت را می‌پوشاند، تعریف شده است. بنابراین نیاز است تا عضو uID را با دستگیره پنجره حاوی tool ، مقدار دهی نمائیم. همچنین عضو uFlags نیز باید با پرچم TTF_IDISHWND ، مقدار دهی گردد.

www.mohandesyar.com

مثال ۲۵- Win32 Debug API – قسمت اول

ویندوز، شامل چندین APIs است که به برنامه نویسان امکان استفاده از توانایی‌های آن را بعنوان یک دیباگر می‌دهد. به آنها Win32 Debug APIs or primitives نیز گفته می‌شود.با استفاده از آنها اعمال زیر را می‌توان انجام داد:

- بارگذاری یک برنامه و یا الحاق به برنامه ای در حال اجرا جهت دیباگ آن
- بدست آوردن اطلاعاتی سطح پایین (low-level) ، در باره برنامه تحت دیباگ مانند process ID ، آدرس تابع entrypoint و غیره
- باخبر شدن از رخ دادن رویدادهای مربوطه مانند شروع و یا خاتمه یک ترد، بارگذاری یک dll در حافظه و غیره.
- ویرایش و انجام تغییراتی در پروسه و یا ترد در حال دیباگ

به زبان ساده تر با استفاده از این APIs ، می‌توان یک دیباگر ساده نوشت. با توجه به گستردگی این بحث، این مثال به چند قسمت تقسیم شد. در این مثال اصول مقدماتی بررسی خواهند گردید.

مراحل استفاده از Win32 Debug APIs به شرح زیر هستند:

۱. ایجاد پروسه‌ای جدید و یا الحاق شدن به پروسه‌ای در حال اجرا.

- ایجاد پروسه دیباگر توسط تابع **CreateProcess** . جهت ایجاد پروسه ای برای دیباگ کردن ، باید از پرچم **DEBUG_PROCESS** استفاده نمود. در این حالت، ویندوز رخدادهای مهم برنامه در حال دیباگ (debug events) را به برنامه ما ارسال خواهد کرد. پروسه برنامه در حال دیباگ بلافاصله به حالت تعلیق در می‌آید تا زمانیکه برنامه ما آماده گردد. اگر در این حین برنامه در حال دیباگ پروسه‌های فرزندی را نیز ایجاد کند ، ویندوز رخدادهای مربوط به تمام این پروسه‌های فرزند را نیز به برنامه ما ارسال خواهد کرد. این رفتار عموما مطلوب کار ما نبوده و با تنظیم پرچم‌های ترکیبی **DEBUG_ONLY_THIS_PROCESS** و **DEBUG_PROCESS**، می‌توان این رفتار را غیرفعال کرد.
- برای الحاق برنامه دیباگ کننده به پروسه در حال اجرا، از تابع **DebugActiveProcess** استفاده می گردد.

www.mohandesyar.com

۲. منتظر دریافت رخ دادهای دیباگ باقی ماندن. پس از ایجاد برنامه دیباگر و بارگذاری برنامه مربوطه ، ترد اصلی برنامه تحت دیباگ به حالت تعلیق در آمده و تا زمان فراخوانی تابع **WaitForDebugEvent** ، بکار خود ادامه نخواهد داد. عملکرد این تابع نیز همانند سایر توابع WaitForXXX می‌باشد. تعریف آن به شرح زیر است:

```
WaitForDebugEvent  proto  lpDebugEvent:DWORD, dwMilliseconds:DWORD
```

lpDebugEvent : آدرس ساختار **DEBUG_EVENT** بوده که با اطلاعات رویداد اتفاق افتاده در برنامه تحت دیباگ، مقدار دهی می‌شود.

dwMilliseconds : مدت زمانی است بر حسب میلی ثانیه ، که برای رخ دادن رویداد دیباگ صبر می‌شود.اگر این پارامتر مساوی **INFINITE** قرار گیرد، تا زمان رخ دادن رویداد دیباگ صبر خواهد شد.

ساختار DEBUG_EVENT به شکل زیر تعریف می‌شود:

```
DEBUG_EVENT  STRUCT
  dwDebugEventCode dd ?
  dwProcessId dd ?
  dwThreadId dd ?
  u DEBUGSTRUCT <>
DEBUG_EVENT  ENDS
```

dwDebugEventCode:مشخص می‌کند که چه نوع رویداد دیباگی رخ داده است. مقادیر ممکنه به شرح زیر هستند:

CREATE_PROCESS_DEBUG_EVENT : یعنی پروسه دیباگ ایجاد شده است (و هنوز در حال اجرا نیست) ، و یا برنامه صرفا خود را به پروسه‌ای در حال اجرا با فراخوانی **DebugActiveProcess** ضمیمه کرده است.

EXIT_PROCESS_DEBUG_EVENT : اتمام پروسه دیباگ

CREATE_THREAD_DEBUG_EVENT : تردی جدید در برنامه در حال دیباگ رخ داده و یا برنامه صرفا خود را به برنامه در حال دیباگ متصل کرده است. لازم به ذکر است که این رویداد در هنگام ایجاد ترد اولیه و اصلی برنامه ، رخ نخواهد داد.

www.mohandesyar.com

EXIT_THREAD_DEBUG_EVENT : تـردی در برنامـه تحـت دیبـاگ خاتمه یافته است. این رویداد برای ترد اولیه دریافت نخواهـد شـد. زمانیکـه رخداد **CREATE_PROCESS_DEBUG_EVENT** دریافـت شـود، هماننـد این است که **CREATE_THREAD_DEBUG_EVENT** رویداده باشد.

LOAD_DLL_DEBUG_EVENT : برنامـــه تحـــت دیبـــاگ dll ایـــی را بارگذاری کرده است (توسط فراخوانی تابع LoadLibrary ویا لینک مستقیم به فایل dll در ابتدای بارگذاری برنامه).

UNLOAD_DLL_DEBUG_EVENT : یـک dll توسـط برنامـه در حـال دیباگ از حافظه خارج گردیده است.

EXCEPTION_DEBUG_EVENT : یـک exception در برنامـه تحـت دیباگ رویداده است. ایـن exception در حقیقـت debug break مـی‌باشـد (int 3h) و قبـل از اجـرای اولـین اینستراکـشن برنامـه تحـت دیبـاگ رخ می‌دهد. برای از سرگیری مجدد دیباگ، از تابع **ContinueDebugEvent** بـه همـراه پـرچم **DBG_CONTINUE** اسـتفاده خواهـد شـد.از پـرچم **DBG_EXCEPTION_NOT_HANDLED** اسـتفاده ننماییـد زیـرا در ایـن حالت برنامه در مورد اجرا تحت ویندوزهای NT دچـار سـردرگمی خواهـد شد (تحت ویندوز ۹۸ بخوبی کار خواهد کرد).

OUTPUT_DEBUG_STRING_EVENT : این رخداد زمانیکه برنامه تحت دیباگ ، از تـابع **DebugOutputString** جهـت ارسـال پیغـام رشـته ای بـه برنامه استفاده نماید، روی می‌دهد.

RIP_EVENT : خطایی در سیستم دیباگ کردن رخ داده است.

dwProcessId and **dwThreadId**: id های پروسه و تردی است که وقایع دیبـاگ در آن رخ می‌دهند. لازم بـه ذکـر اسـت کـه اگـر از تـابع **CreateProcess** اسـتفاده شـود، ایـن مقـادیر از سـاختار **PROCESS_INFO** قابـل دریافـت مـی‌باشـند. از ایـن مقادیر جهت تمایز قائل شدن بین رخ دادهای اتفاق افتاده در برنامـه تحـت دیبـاگ و پروســه‌هــای فرزنــد آن مــی‌تــوان اســتفاده کــرد (در حــالتی کــه از پــرچم **DEBUG_ONLY_THIS_PROCESS** استفاده نشود).

u : union بوده و حاوی اطلاعات بیـشتری در مـورد رخ داد اسـت. وابـسته بـه مقـدار **dwDebugEventCode** ، یکی از ساختارهای زیر می‌تواند باشد:

www.mohandesyar.com

value in dwDebugEventCode	Interpretation of u
CREATE_PROCESS_DEBUG_EVENT	A **CREATE_PROCESS_DEBUG_INFO** structure named **CreateProcessInfo**
EXIT_PROCESS_DEBUG_EVENT	An **EXIT_PROCESS_DEBUG_INFO structure named** ExitProcess
CREATE_THREAD_DEBUG_EVENT	A **CREATE_THREAD_DEBUG_INFO** structure named **CreateThread**
EXIT_THREAD_DEBUG_EVENT	An **EXIT_THREAD_DEBUG_EVENT** structure named **ExitThread**
LOAD_DLL_DEBUG_EVENT	A **LOAD_DLL_DEBUG_INFO** structure named **LoadDll**
UNLOAD_DLL_DEBUG_EVENT	An **UNLOAD_DLL_DEBUG_INFO** structure named **UnloadDll**
EXCEPTION_DEBUG_EVENT	An **EXCEPTION_DEBUG_INFO** structure named **Exception**
OUTPUT_DEBUG_STRING_EVENT	An **OUTPUT_DEBUG_STRING_INFO** structure named **DebugString**
RIP_EVENT	A **RIP_INFO** structure named **RipInfo**

در مورد تمامی ساختارهای فوق اینجا بحث نخواهد شد و تنها ساختار **CREATE_PROCESS_DEBUG_INFO** بیشتر توضیح داده می‌شود.فرض کنید که تابع **WaitForDebugEvent** فراخوانی شده و نیز کارش به پایان رسیده است. اولین کاری که باید صورت گیرد بررسی مقدار **dwDebugEventCode** است تا مشخص شود کدام نوع از رویدادها اتفاق افتاده است. برای مثال اگر این مقدار مساوی **CREATE_PROCESS_DEBUG_EVENT** باشد، عدد قرار گرفته در **u** را بعنوان **CreateProcessInfo** می توان تفسیر کرد و توسط **u.CreateProcessInfo** به آن دسترسی داشت.

۳. ارائه واکنش دلخواه به رخ داد. زمانی که کار تابع **WaitForDebugEvent** به پایان می‌رسد، به این معنا است که رویداد دیباگ رخ داده است و یا زمان مورد نظر جهت صبر کردن برای رخ دادن یک رویداد، منقضی شده است. در ادامه باید برنامه با بررسی مقدار **dwDebugEventCode** ، به رخداد مورد نظر پاسخ دهد. (شبیه به پردازش پیغام های پنجره می‌باشد)

۴. اجازه دادن به ادامه اجرای برنامه تحت دیباگ. زمانیکه یک رخداد دیباگ روی می‌دهد، ویندوز برنامه تحت دیباگ را به حالت تعلیق در می‌آورد. پس از پردازش رویداد باید مجددا اجرای برنامه را از سر گرفت. اینکار با فراخوانی تابع **ContinueDebugEvent** قابل انجام است.تعریف این تابع به شرح زیر است:

```
ContinueDebugEvent  proto  dwProcessId:DWORD,\
                           dwThreadId:DWORD, dwContinueStatus:DWORD
```


www.mohandesyar.com

dwProcessId and **dwThreadId**: id پروسه و تردی هستند که کار آن باید مجدد از سرگرفته شود. این دو مقدار از ساختار **DEBUG_EVENT** قابل دریافت هستند.

dwContinueStatus: نحوه از سرگیری مجدد ترد را مشخص می‌کند.برای آن دو مقدار **DBG_CONTINUE** and **DBG_EXCEPTION_NOT_HANDLED** مجاز هستند.این دو مقدار برای تمامی رخ دادها بجز **EXCEPTION_DEBUG_EVENT** ، به معنای از سرگیری مجدد کد است. در حالت رخ دادن خطا، با ذکر پرچم **DBG_CONTINUE** ، بدون در نظر گرفتن خطا، اجرای ادامه کد از سر گرفته خواهد شد. اگر از **DBG_EXCEPTION_NOT_HANDLED** استفاده شود، بدین معنا است که برنامه ما مدیریت خطا را بعهده نخواهد گرفت و برنامه تحت دیباگ باید از روالهای خاص خود در جهت اینکار کمک گیرد. همانگونه که پیشتر در مورد debug break صحبت شد، در این حالت باید از **DBG_CONTINUE** استفاده گردد.

۵. تکرار این مراحل در یک حلقه بی پایان (همانند حلقه پیغامها) تا زمان خاتمه پروسه دیباگ.

برای مثال:

```
.while TRUE
    invoke WaitForDebugEvent, addr DebugEvent, INFINITE
   .break .if DebugEvent.dwDebugEventCode==EXIT_PROCESS_DEBUG_EVENT
   <Handle the debug events>
   invoke ContinueDebugEvent, DebugEvent.dwProcessId,\
                              DebugEvent.dwThreadId, \
                              DBG_EXCEPTION_NOT_HANDLED
.endw
```

لازم به ذکر است زمانیکه دیباگ یک برنامه آغاز می‌شود تا پایان آن امکان خاتمه برنامه دیباگر وجود ندارد.

خلاصه موارد فوق به صورت زیر است:

۱. ایجاد پروسه و یا ضمیمه شدن به پروسه‌ای در حال اجرا
۲. منتظر پیغام‌های دیباگ ماندن
۳. پردازش پیغام‌های دیباگ رسیده
۴. اجازه دادن به برنامه تحت دیباگ جهت اجرای ادامه کد خود
۵. ادامه این پروسه در حلقه‌ای بی‌پایان تا زمان خاتمه برنامه تحت دیباگ

www.mohandesyar.com

کد مثال برنامه ۲۵ :

```
.386
.model flat,stdcall
option casemap:none
include \masm32\include\windows.inc
include \masm32\include\kernel32.inc
include \masm32\include\comdlg32.inc
include \masm32\include\user32.inc
includelib \masm32\lib\kernel32.lib
includelib \masm32\lib\comdlg32.lib
includelib \masm32\lib\user32.lib
.data
AppName db "Win32 Debug Example no.1",0
ofn OPENFILENAME <>
FilterString db "Executable Files",0,"*.exe",0
            db "All Files",0,"*.*",0,0
ExitProc db "The debuggee exits",0
NewThread db "A new thread is created",0
EndThread db "A thread is destroyed",0
ProcessInfo db "File Handle: %lx ",0dh,0Ah
            db "Process Handle: %lx",0Dh,0Ah
            db "Thread Handle: %lx",0Dh,0Ah
            db "Image Base: %lx",0Dh,0Ah
            db "Start Address: %lx",0
.data?
buffer db 512 dup(?)
startinfo STARTUPINFO <>
pi PROCESS_INFORMATION <>
DBEvent DEBUG_EVENT <>
.code
start:
mov ofn.lStructSize,sizeof ofn
mov ofn.lpstrFilter, offset FilterString
mov ofn.lpstrFile, offset buffer
mov ofn.nMaxFile,512
mov ofn.Flags, OFN_FILEMUSTEXIST or OFN_PATHMUSTEXIST or OFN_LONGNAMES or OFN_EXPLORER or
OFN_HIDEREADONLY
invoke GetOpenFileName, ADDR ofn
.if eax==TRUE
invoke GetStartupInfo,addr startinfo
invoke CreateProcess, addr buffer, NULL, NULL, NULL, FALSE, DEBUG_PROCESS+
DEBUG_ONLY_THIS_PROCESS, NULL, NULL, addr startinfo, addr pi
.while TRUE
   invoke WaitForDebugEvent, addr DBEvent, INFINITE
   .if DBEvent.dwDebugEventCode==EXIT_PROCESS_DEBUG_EVENT
      invoke MessageBox, 0, addr ExitProc, addr AppName, MB_OK+MB_ICONINFORMATION
      .break
   .elseif DBEvent.dwDebugEventCode==CREATE_PROCESS_DEBUG_EVENT
      invoke wsprintf, addr buffer, addr ProcessInfo, DBEvent.u.CreateProcessInfo.hFile,
DBEvent.u.CreateProcessInfo.hProcess, DBEvent.u.CreateProcessInfo.hThread,
DBEvent.u.CreateProcessInfo.lpBaseOfImage, DBEvent.u.CreateProcessInfo.lpStartAddress
      invoke MessageBox,0, addr buffer, addr AppName, MB_OK+MB_ICONINFORMATION
   .elseif DBEvent.dwDebugEventCode==EXCEPTION_DEBUG_EVENT
      .if DBEvent.u.Exception.pExceptionRecord.ExceptionCode==EXCEPTION_BREAKPOINT
        invoke ContinueDebugEvent, DBEvent.dwProcessId, DBEvent.dwThreadId, DBG_CONTINUE
       .continue
      .endif
```


www.mohandesyar.com

```
  .elseif DBEvent.dwDebugEventCode==CREATE_THREAD_DEBUG_EVENT
     invoke MessageBox,0, addr NewThread, addr AppName, MB_OK+MB_ICONINFORMATION
  .elseif DBEvent.dwDebugEventCode==EXIT_THREAD_DEBUG_EVENT
     invoke MessageBox,0, addr EndThread, addr AppName, MB_OK+MB_ICONINFORMATION
  .endif
  invoke ContinueDebugEvent, DBEvent.dwProcessId, DBEvent.dwThreadId,
DBG_EXCEPTION_NOT_HANDLED
.endw
invoke CloseHandle,pi.hProcess
invoke CloseHandle,pi.hThread
.endif
invoke ExitProcess, 0
end start
```

آنالیز کد فوق:

برنامه اعضای ساختار OPENFILENAME را مقدار دهی کرده و سپس با فراخوانی GetOpenFileName به کاربر امکان انتخاب برنامه ای را جهت دیباگ می‌دهد.سپس:

```
invoke GetStartupInfo,addr startinfo
invoke CreateProcess, addr buffer, NULL, NULL, NULL, FALSE, DEBUG_PROCESS+
DEBUG_ONLY_THIS_PROCESS, NULL, NULL, addr startinfo, addr pi
```

هنگامیکه کاربر برنامه‌ای را انتخاب می‌نماید، با فراخوانی CreateProcess ، برنامه را بارگذاری می کند و با فراخوانی GetStartupInfo ، ساختار STARTUPINFO با مقادیر پیش فرض مقداردهی می گردد. لازم به ذکر است که از پرچم های ترکیبی DEBUG_PROCESS و DEBUG_ONLY_THIS_PROCESS ، جهت دیباگ کردن این برنامه و نه پروسه های فرزند آن استفاده گردید. در ادامه :

```
.while TRUE
  invoke WaitForDebugEvent, addr DBEvent, INFINITE
```

هنگامیکه برنامه مورد نظر جهت دیباگ فراخوانی می گردد، با ورود به حلقه‌ای بی پایان، تابع WaitForDebugEvent را فراخوانی می‌نمائیم. با توجه به استفاده از مقدار INFINITE بعنوان پارامتر دوم تابع، تا زمانیکه رخداد دیباگ روی ندهد، کار این تابع به پایان نخواهد رسید. پس از رویدادن رخ داد دیباگ، کار این تابع پایان یافته و DBEvent مقدار دهی می‌شود. سپس :

```
  .if DBEvent.dwDebugEventCode==EXIT_PROCESS_DEBUG_EVENT
     invoke MessageBox, 0, addr ExitProc, addr AppName, MB_OK+MB_ICONINFORMATION
     .break
```


www.mohandesyar.com

در ابتدا مقدار dwDebugEventCode بررسی می‌شود. اگر مقدار آن EXIT_PROCESS_DEBUG_EVENT بود، با نمایش پیغام The debuggee exits ، حلقه خاتمه می‌یابد. در ادامه:

```
.elseif DBEvent.dwDebugEventCode==CREATE_PROCESS_DEBUG_EVENT
    invoke wsprintf, addr buffer, addr ProcessInfo, DBEvent.u.CreateProcessInfo.hFile,
DBEvent.u.CreateProcessInfo.hProcess, DBEvent.u.CreateProcessInfo.hThread,
DBEvent.u.CreateProcessInfo.lpBaseOfImage, DBEvent.u.CreateProcessInfo.lpStartAddress
    invoke MessageBox,0, addr buffer, addr AppName, MB_OK+MB_ICONINFORMATION
```

اگر مقدار dwDebugEventCode مساوی CREATE_PROCESS_DEBUG_EVENT باشد، اطلاعات جالبی را در مورد برنامه تحت دیباگ نمایش خواهیم داد. ما این اطلاعات را از u.CreateProcessInfo بدست می‌آوریم. CreateProcessInfo ساختاری از نوع CREATE_PROCESS_DEBUG_INFO است. اطلاعات بیشتری در مورد این ساختار را می‌توان از MSDN بدست آورد.

```
.elseif DBEvent.dwDebugEventCode==EXCEPTION_DEBUG_EVENT
    .if DBEvent.u.Exception.pExceptionRecord.ExceptionCode==EXCEPTION_BREAKPOINT
      invoke ContinueDebugEvent, DBEvent.dwProcessId, DBEvent.dwThreadId, DBG_CONTINUE
     .continue
    .endif
```

اگر مقدار dwDebugEventCode مساوی EXCEPTION_DEBUG_EVENT باشد، باید نوع دقیق exception رخ داده را بررسی نمود. این مورد را با بررسی مقدار ExceptionCode می‌توان یافت. اگر مقدار ExceptionCode مساوی EXCEPTION_BREAKPOINT باشد و نیز برای اولین بار رخ دهد و یا اطمینان دارید که int 3h دیگری نیست ، می توان فرض کرد که این استثناء هنگام اجرای اولین اینستراکشن برنامه رخ داده است. سپس ادامه دیباگ را باید با فراخوانی تابع ContinueDebugEvent به همراه پرچم DBG_CONTINUE ، میسر ساخت. در ادامه به سایر رخ دادهای دیباگ خواهیم پرداخت:

```
.elseif DBEvent.dwDebugEventCode==CREATE_THREAD_DEBUG_EVENT
    invoke MessageBox,0, addr NewThread, addr AppName, MB_OK+MB_ICONINFORMATION
.elseif DBEvent.dwDebugEventCode==EXIT_THREAD_DEBUG_EVENT
    invoke MessageBox,0, addr EndThread, addr AppName, MB_OK+MB_ICONINFORMATION
.endif
```

اگر مقدار dwDebugEventCode مساوی CREATE_THREAD_DEBUG_EVENT و یا EXIT_THREAD_DEBUG_EVENT باشد ، پیغامی را نمایش خواهیم داد. در ادامه:

```
  invoke ContinueDebugEvent, DBEvent.dwProcessId,\
                             DBEvent.dwThreadId, DBG_EXCEPTION_NOT_HANDLED
.endw
```


www.mohandesyar.com

تنها برای حالت EXCEPTION_DEBUG_EVENT ، تابع ContinueDebugEvent را با پرچم DBG_EXCEPTION_NOT_HANDLED ، جهت از سرگیری مجدد ترد فراخوانی خواهیم کرد. سپس:

```
invoke CloseHandle,pi.hProcess
invoke CloseHandle,pi.hThread
```

هنگامیکه برنامه دیباگ خاتمه می‌یابد، از حلقه دیباگ خارج شده ایم. بنابراین باید دستگیره‌های ترد و پروسه را بست. البته بستن آنها به معنای خاتمه ترد و یا پروسه نیست، بلکه بدین معنا است که دیگر قصد استفاده از آنها را نداریم.

www.mohandesyar.com

مثال ۲۶- Win32 Debug API – قسمت دوم

در ادامه این مبحث، نحوه اصلاح و ایجاد تغییرات را در برنامه تحت دیباگ خواهیم آموخت (در حقیقت نوعی تزریق در پروسه). برای انجام این مقصود، چندین APIs وجود دارند.

- **ReadProcessMemory**: با استفاده از این تابع می‌توان حافظه پروسه مشخصی را خواند و به صورت زیر تعریف می‌شود:

```
ReadProcessMemory proto  hProcess:DWORD, lpBaseAddress:DWORD,\
                         lpBuffer:DWORD, nSize:DWORD,\
                         lpNumberOfBytesRead:DWORD
```

hProcess: دستگیره پروسه ای است که قصد خواندن حافظه آنرا داریم.
lpBaseAddress:آدرس پروسه مقصد جهت خواندن است. برای مثال اگر قصد خواندن ۴ بایت از پروسه دیباگ شروع شده از 401000h را داریم، این پارامتر باید مساوی 401000h قرارگیرد.
lpBuffer:آدرس بافری است که بایت‌های خوانده شده در آن قرار خواهند گرفت.
nSize:تعداد بایتهایی است که باید خوانده شوند.
lpNumberOfBytesRead:آدرس متغیری است که تعداد بایتهایی که واقعا خوانده شده‌اند در آن قرار می‌گیرد. اگر این مورد برای شما اهمیتی ندارد می‌توانید آنرا مساوی NULL قرار دهید.

- **WriteProcessMemory**: این تابع همتای **ReadProcessMemory** است و جهت نوشتن بر روی حافظه پروسه مقصد بکار گرفته می‌شود.پارامترهای آن با تابع **ReadProcessMemory** یکی می‌باشد.

جهت معرفی دو تابع بعدی نیاز به مقداری توضیح می‌باشد. تحت سیستم عاملی چندکاره مانند ویندوز، امکان در حال اجرا بودن چندین برنامه در یک زمان وجود دارد. ویندوز به هر ترد یک قطعه زمانی را اختصاص می‌دهد (timeslice). هنگامیکه این قطعه زمانی منقضی شود، ویندوز این ترد را متوقف ساخته و به تردی دیگر با بالاترین سطح تقدم سوئیچ

www.mohandesyar.com

خواهد کرد. ویندوز قبل از سوئیچ کردن به تردی دیگر، مقادیر ترد فعلی را ذخیره ساخته و هنگام از سرگیری مجدد، دوباره از آنها استفاده خواهد نمود.به مقادیر ذخیره شده یک ترد در رجیسترها، context گفته می‌شود.

هنگامیکه رویداد مربوط به دیباگ رخ می‌دهد، ویندوز برنامه تحت دیباگ را به حالت تعلیق درآورده و context آنرا ذخیره می‌نماید.از آنجائیکه برنامه تحت دیباگ به حالت تعلیق درآمده، می توان اطمینان داشت که مقادیر context آن بدون تغییر باقی خواهند ماند. این مقادیر را می‌توان توسط تابع GetThreadContext بدست آورد و یا با استفاده از تابع SetThreadContext آنها را تنظیم کرد.این دو API بسیار قوی هستند و هرگونه تغییری که در مقادیر context صورت گیرد، پس از ازسرگیری مجدد ترد، به برنامه تحت دیباگ منعکس خواهند شد. برای مثال حتی می‌توان مقادیر رجیستر eip را نیزتغییر داد (که در شرایط عادی میسر نیست) و جریان اجرای برنامه را به مسیری دیگر سوق داد!

GetThreadContext proto hThread:DWORD, lpContext:DWORD

hThread : دستگیره تردی است که می‌خواهیم context آنرا بدست آوریم.

lpContext : آدرس ساختار **CONTEXT** است که پس از فراخوانی موفقیت آمیز تابع، مقدار دهی خواهد شد.

تابع **SetThreadContext** نیز پارامترهای یکسانی دارد. ساختار CONTEXT به شکل زیر است:

- CONTEXT STRUCT
- ContextFlags dd ?
  ;---
  ; This section is returned if ContextFlags contains the value CONTEXT_DEBUG_REGISTERS
- ;---
  iDr0 dd ?
  iDr1 dd ?
  iDr2 dd ?
  iDr3 dd ?
  iDr6 dd ?
  iDr7 dd ?
- ;---
  ; This section is returned if ContextFlags contains the value CONTEXT_FLOATING_POINT
- ;---
- FloatSave FLOATING_SAVE_AREA <>
- ;---
  ; This section is returned if ContextFlags contains the value CONTEXT_SEGMENTS
- ;---
- regGs dd ?
  regFs dd ?
  regEs dd ?
  regDs dd ?
- ;---
  ; This section is returned if ContextFlags contains the value CONTEXT_INTEGER
- ;---
- regEdi dd ?
  regEsi dd ?

www.mohandesyar.com

```
    regEbx dd ?
    regEdx dd ?
    regEcx dd ?
    regEax dd ?
• ;-------------------------------------------------------------------------------------------------------
    ; This section is returned if ContextFlags contains the value CONTEXT_CONTROL
• ;-------------------------------------------------------------------------------------------------------
• regEbp dd ?
    regEip dd ?
    regCs dd ?
    regFlag dd ?
    regEsp dd ?
    regSs dd ?
• ;-------------------------------------------------------------------------------------------------------
    ; This section is returned if ContextFlags contains the value CONTEXT_EXTENDED_REGISTERS
• ;-------------------------------------------------------------------------------------------------------

ExtendedRegisters db MAXIMUM_SUPPORTED_EXTENSION dup(?) CONTEXT ENDS
```

همانطور که ملاحظه می‌نمائید اعضای این ساختار تقلیدی از رجیسترهای یک پردازنده هستند. قبل از استفاده از آن باید توسط پارامتر ContextFlags مشخص کرد که قصد استفاده از کدامیک از اعضای آنرا داریم. برای مثال اگر قصد استفاده از تمام اعضای آنرا داریم، باید این پارامتر را مساوی CONTEXT_FULL قرار داد. اگر تنها قصد استفاده از رجیسترهای regEbp, regEip, regCs, regFlag, regEsp or regSs را دارید، این پارامتر باید مساوی CONTEXT_CONTROL قرار گیرد.

هنگام استفاده از این ساختار حتما باید به نکته زیر دقت نمود، در غیراینصورت نتایج نامطلوبی بدست خواهد آمد:

```
align  dword
MyContext  CONTEXT <>
```

برنامه مثال ۲۶:

برنامه اول نمایشی است از فراخوانی تابع DebugActiveProcess. در ابتدا باید برنامه win.exe را اجرا نموده و سپس برنامه را اجرا کرد. در این حالت برنامه به win.exe متصل شده و کدهای آنرا تغییر می‌دهد.

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


www.mohandesyar.com

```
includelib \masm32\lib\comdlg32.lib
includelib \masm32\lib\user32.lib

.data
AppName db "Win32 Debug Example no.2",0
ClassName db "SimpleWinClass",0
SearchFail db "Cannot find the target process",0
TargetPatched db "Target patched!",0
buffer dw 9090h

.data?
DBEvent DEBUG_EVENT <>
ProcessId dd ?
ThreadId dd ?
align dword
context CONTEXT <>

.code
start:
invoke FindWindow, addr ClassName, NULL
.if eax!=NULL
    invoke GetWindowThreadProcessId, eax, addr ProcessId
    mov ThreadId, eax
    invoke DebugActiveProcess, ProcessId
    .while TRUE
      invoke WaitForDebugEvent, addr DBEvent, INFINITE
      .break .if DBEvent.dwDebugEventCode==EXIT_PROCESS_DEBUG_EVENT
      .if DBEvent.dwDebugEventCode==CREATE_PROCESS_DEBUG_EVENT
        mov context.ContextFlags, CONTEXT_CONTROL
        invoke GetThreadContext,DBEvent.u.CreateProcessInfo.hThread, addr context
        invoke WriteProcessMemory, DBEvent.u.CreateProcessInfo.hProcess,\

                                          context.regEip ,addr buffer, 2, NULL
        invoke MessageBox, 0, addr TargetPatched, addr AppName, MB_OK+MB_ICONINFORMATION
      .elseif DBEvent.dwDebugEventCode==EXCEPTION_DEBUG_EVENT
        .if DBEvent.u.Exception.pExceptionRecord.ExceptionCode==EXCEPTION_BREAKPOINT
          invoke ContinueDebugEvent, DBEvent.dwProcessId,DBEvent.dwThreadId, DBG_CONTINUE
          .continue
        .endif
      .endif
      invoke ContinueDebugEvent, DBEvent.dwProcessId, DBEvent.dwThreadId,
DBG_EXCEPTION_NOT_HANDLED
   .endw
.else
    invoke MessageBox, 0, addr SearchFail, addr AppName,MB_OK+MB_ICONERROR .endif
invoke ExitProcess, 0
end start

;-----------------------------------------------------------------
; The partial source code of win.asm, our debuggee. It's actually
; the simple window example in tutorial 2 with an infinite loop inserted
; just before it enters the message loop.
;-----------------------------------------------------------------

.....
mov wc.hIconSm,eax
invoke LoadCursor,NULL,IDC_ARROW
```


www.mohandesyar.com

```
mov wc.hCursor,eax
invoke RegisterClassEx, addr wc
INVOKE CreateWindowEx,NULL,ADDR ClassName,ADDR AppName,\
WS_OVERLAPPEDWINDOW,CW_USEDEFAULT,\
CW_USEDEFAULT,CW_USEDEFAULT,CW_USEDEFAULT,NULL,NULL,\ hInst,NULL
mov hwnd,eax
jmp $ <---- Here's our infinite loop. It assembles to EB FE
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
```

بررسی کد فوق:

```
invoke  FindWindow, addr ClassName, NULL
```

برنامه ما نیاز دارد تا خود را توسط فراخونی تابع DebugActiveProcess ، به برنامه تحت دیباگ جهت تزریق کد خود، متصل سازد. برای این منظور به process Id نیاز است. این مقدار را توسط GetWindowThreadProcessId می‌توان بدست آورد که پارامتر ورودی آن دستگیره پنجره است. برای بدست آوردن دستگیره پنجره، از تابع FindWindow استفاده خواهد شد.با دریافت نام کلاس پنجره مورد نظر، دستگیره آن بازگشت داده خواهد شد. در صورت شکست ، نال را بر می گرداند. در ادامه :

```
.if eax!=NULL
   invoke GetWindowThreadProcessId, eax, addr ProcessId
   mov ThreadId, eax
   invoke DebugActiveProcess, ProcessId
```

پس از بدست آوردن process Id ، می‌توان DebugActiveProcess را فراخوانی کرد. سپس می‌توان وارد حلقه انتظار رخ دادها شد. سپس :

```
   .if DBEvent.dwDebugEventCode==CREATE_PROCESS_DEBUG_EVENT
      mov context.ContextFlags, CONTEXT_CONTROL
      invoke GetThreadContext,DBEvent.u.CreateProcessInfo.hThread, addr context
```


www.mohandesyar.com

دریافت CREATE_PROCESS_DEBUG_INFO ، بدین معنا است که پروسه تحت دیباگ به حالت تعلیق درآمده است و منتظر عملیات جراحی ما می‌باشد! در این مثال، اینستراکشن‌های حلقه بی پایان برنامه تحت دیباگ را (0EBh 0FEh) با NOPs (90h 90h) جایگزین می کنیم.

در ابتدا نیاز است تا آدرس اینستراکشن را بدست آوریم. از آنجائیکه برنامه تحت دیباگ در حلقه برنامه ما که به آن متصل شده است قرار دارد، eip همواره به اینستراکشن فوق اشاره می کند.بنابراین نیاز است تا مقدار eip را بدست آورد. از تابع GetThreadContext برای این منظور کمک خواهیم گرفت. همچنین پارامتر ContextFlags آنرا مساوی CONTEXT_CONTROL قرار می‌دهیم تا مقادیر اعضای کنترلی آنرا بدست آوریم. در ادامه:

```
invoke WriteProcessMemory, DBEvent.u.CreateProcessInfo.hProcess,\
context.regEip ,addr buffer, 2, NULL
```

پس از بدست آوردن eip ، با فراخوانی WriteProcessMemory ، "jmp $" را با NOPs جای نویسی می کنیم. اینکار سبب می‌شود تا از حلقه بی پایان برنامه تحت دیباگ ، خارج شویم. پس از آن پیغامی را به کاربر نمایش داده و برنامه تحت دیباگ را به کمک ContinueDebugEvent ، از سر خواهیم گرفت.

مثالی دیگر از روشی متفاوت برای خارج شدن از حلقه برنامه تحت دیباگ استفاده می‌نماید.

```
.....
.....
.if DBEvent.dwDebugEventCode==CREATE_PROCESS_DEBUG_EVENT
   mov context.ContextFlags, CONTEXT_CONTROL
   invoke GetThreadContext,DBEvent.u.CreateProcessInfo.hThread, addr context
   add context.regEip,2
   invoke SetThreadContext,DBEvent.u.CreateProcessInfo.hThread, addr context
   invoke MessageBox, 0, addr LoopSkipped, addr AppName, MB_OK+MB_ICONINFORMATION
.....
.....
```

کد فوق نیز از GetThreadContext جهت مقدار فعلی eip استفاده می کند، اما بجای جای نویسی "jmp $" ، مقدار رجیستر regEip را دو واحد افزایش می‌دهد (تا از روی اینستراکشن پرش نماید). بنابراین زمانیکه ترد برنامه تحت دیباگ مجددا شروع به اجرا نماید، از اینستراکشن بعد از "jmp $" ، شروع به اجرا خواهد نمود.

www.mohandesyar.com

مثال ۲۷- Win32 Debug API - قسمت سوم

در ادامه مثالهای قبلی ، در این مثال نحوه دنبال کردن (trace) یک برنامه تحت دیباگ را بررسی خواهیم کرد. اگر تابحال از برنامه های دیباگر استفاده کرده باشید، حتما با روش های دنبال کردن کد آشنایی دارید. زمانیکه برنامه ای را دنبال می کنید، برنامه پس از اجرای هر اینستراکشن متوقف شده و امکان بررسی مقادیر موجود در رجیسترها و حافظه را میسر می‌سازد.به این روش Single-stepping نیز گفته می‌شود و توسط CPU مهیا می گردد. هشتمین بیت رجیستر پرچم در اینجا **trap flag** نام دارد. اگر این پرچم تنظیم شده باشد، پردازشگر در حالت single-step اجرا می گردد.پردازشگر پس از اجرای هر اینستراکشن، یک debug exception تولید خواهد کرد.پس از ایجاد این استثناء ، trap flag به صورت خودکار پاک می گردد.پیاده سازی روش single-stepping توسط API ویندوز نیز میسر است. روش کار به شرح زیر است:

۱. فراخوانی GetThreadContext و مشخص نمودن CONTEXT_CONTROL بعنوان مقدار پارامتر ContextFlags آن، جهت بدست آوردن مقدار flag register .
۲. تنظیم نمودن بیت trap در عضو regFlag ساختار CONTEXT .
۳. فراخوانی SetThreadContext .
۴. منتظر رخ دادهای دیباگ ماندن. در این حالت برنامه تحت دیباگ در حالت single -step اجرا شده و پس از اجرای هر اینستراکشن ، رخ داد EXCEPTION_DEBUG_EVENT ، به همراه مقدار EXCEPTION_SINGLE_STEP در u.Exception.pExceptionRecord.ExceptionCode دریافت می گردد.
۵. اگر نیاز به دنبال کردن اینستراکشن بعدی است ، مجدد باید بیت trap تنظیم گردد.

کد مثال ۲۷ :

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

.data
AppName db "Win32 Debug Example no.4",0
ofn OPENFILENAME <>
FilterString db "Executable Files",0,"*.exe",0
```


www.mohandesyar.com

```
        db "All Files",0,"*.*",0,0
ExitProc db "The debuggee exits",0Dh,0Ah
       db "Total Instructions executed : %lu",0
TotalInstruction dd 0

.data?
buffer db 512 dup(?)
startinfo STARTUPINFO <>
pi PROCESS_INFORMATION <>
DBEvent DEBUG_EVENT <>
align dword
context CONTEXT <>

.code
start:
mov ofn.lStructSize,SIZEOF ofn
mov ofn.lpstrFilter, OFFSET FilterString
mov ofn.lpstrFile, OFFSET buffer
mov ofn.nMaxFile,512
mov ofn.Flags, OFN_FILEMUSTEXIST or OFN_PATHMUSTEXIST or OFN_LONGNAMES or OFN_EXPLORER or
OFN_HIDEREADONLY
invoke GetOpenFileName, ADDR ofn
.if eax==TRUE
   invoke GetStartupInfo,addr startinfo
   invoke CreateProcess, addr buffer, NULL, NULL, NULL, FALSE, DEBUG_PROCESS+
DEBUG_ONLY_THIS_PROCESS, NULL, NULL, addr startinfo, addr pi
   .while TRUE
     invoke WaitForDebugEvent, addr DBEvent, INFINITE
     .if DBEvent.dwDebugEventCode==EXIT_PROCESS_DEBUG_EVENT
       invoke wsprintf, addr buffer, addr ExitProc, TotalInstruction
       invoke MessageBox, 0, addr buffer, addr AppName, MB_OK+MB_ICONINFORMATION
       .break
     .elseif DBEvent.dwDebugEventCode==EXCEPTION_DEBUG_EVENT          .if
DBEvent.u.Exception.pExceptionRecord.ExceptionCode==EXCEPTION_BREAKPOINT
          mov context.ContextFlags, CONTEXT_CONTROL
          invoke GetThreadContext, pi.hThread, addr context
          or context.regFlag,100h
          invoke SetThreadContext,pi.hThread, addr context
          invoke ContinueDebugEvent, DBEvent.dwProcessId, DBEvent.dwThreadId, DBG_CONTINUE
          .continue
       .elseif DBEvent.u.Exception.pExceptionRecord.ExceptionCode==EXCEPTION_SINGLE_STEP
          inc TotalInstruction
          invoke GetThreadContext,pi.hThread,addr context or context.regFlag,100h
          invoke SetThreadContext,pi.hThread, addr context
          invoke ContinueDebugEvent, DBEvent.dwProcessId, DBEvent.dwThreadId,DBG_CONTINUE
          .continue
       .endif
     .endif
     invoke ContinueDebugEvent, DBEvent.dwProcessId, DBEvent.dwThreadId,
DBG_EXCEPTION_NOT_HANDLED
   .endw
.endif
invoke CloseHandle,pi.hProcess
invoke CloseHandle,pi.hThread
invoke ExitProcess, 0
end start
```


www.mohandesyar.com

بررسی کد فوق:

برنامه در ابتدا دیالوگ باکس گشودن فایل را نمایش می‌دهد. هنگامیکه فایل اجرایی انتخاب می گردد، برنامه آنرا در حالت single-step اجرا خواهد کرد. در ادامه:

```
.elseif DBEvent.dwDebugEventCode==EXCEPTION_DEBUG_EVENT          .if
DBEvent.u.Exception.pExceptionRecord.ExceptionCode==EXCEPTION_BREAKPOINT
```

از این موقعیت برای قرار دادن برنامه تحت دیباگ در حالت single-step استفاده خواهیم کرد. همانطور که در قسمت‌های قبل نیز مشاهده نمودید، ویندوز قبل از اجرای اولین اینستراکشن برنامه، رخ داد EXCEPTION_BREAKPOINT را به برنامه خواهد فرستاد. سپس :

```
mov context.ContextFlags, CONTEXT_CONTROL
invoke GetThreadContext, pi.hThread, addr context
```

با فراخوانی GetThreadContext ، اعضای ساختار CONTEXT ، با مقادیر جاری موجود در رجیسترها، مقدار دهی خواهند شد. خصوصا به مقدار جاری رجیستر پرچم نیاز است:

```
or context.regFlag,100h
```

در این حالت بیت trap (بیت هشتم) ، تنظیم خواهد گردید. در ادامه :

```
invoke SetThreadContext,pi.hThread, addr context
invoke ContinueDebugEvent, DBEvent.dwProcessId, DBEvent.dwThreadId, DBG_CONTINUE
.continue
```

سپس با فراخوانی SetThreadContext ، مقادیر ساختار CONTEXT با مقادیر جدید جایگزین خواهند شد. سپس با فراخوانی ContinueDebugEvent به همراه پرچم DBG_CONTINUE ، اجرای برنامه تحت دیباگ از سر گرفته خواهد شد. سپس :

```
.elseif DBEvent.u.Exception.pExceptionRecord.ExceptionCode==EXCEPTION_SINGLE_STEP
  inc TotalInstruction
```

با اجرای هر اینستراکشن در برنامه تحت دیباگ، رخ داد EXCEPTION_DEBUG_EVENT دریافت خواهد گردید. در ادامه باید مقدار u.Exception.pExceptionRecord.ExceptionCode بررسی گردد.در حالت single-step ، مقدار EXCEPTION_SINGLE_STEP دریافت خواهد شد. در این حالت می‌توان مقدار TotalInstruction را یک واحد افزایش داد ، زیرا اطمینان داریم که تنها یک اینستراکشن اجرا شده است. در ادامه:

```
invoke GetThreadContext,pi.hThread,addr context or context.regFlag,100h
invoke SetThreadContext,pi.hThread, addr context
invoke ContinueDebugEvent, DBEvent.dwProcessId, DBEvent.dwThreadId,DBG_CONTINUE
.continue
```


www.mohandesyar.com

با توجه به اینکه trap flag پس از اجرای هر اینستراکشن به صورت خودکار پاک می‌شود، باید مجدد آنرا تنظیم نمود تا بتوان روش single-stepping را ادامه داد.

اخطار!

با توجه به کند بودن پروسه دنبال کردن ، برای تست این برنامه از فایلهای اجرایی حجیم استفاده نکنید.

www.mohandesyar.com

مثال ۲۸- Listview Control

listview control یکی از انواع common controls همانند treeview و غیره است. این کنترل نمونه بهبود یافته listbox محسوب می گردد.برای ایجاد این کنترل به دو روش می‌توان عمل کرد. روش اول که ساده نیز می‌باشد استفاده از یک ریسورس ادیتور است. همچنین فراخوانی InitCommonControls را در قسمتی از کد خود نیز فراموش نکنید. روش دیگر استفاده از CreateWindowEx است. در اینجا نام کلاس مربوط به این کنترل SysListView32 است (و نه WC_LISTVIEW). چهار حالت برای مشاهده داده‌ها در کنترل listview مهیا است :

icon, small icon, list and report views

در ابتدای ایجاد این کنترل حتما باید یکی از حالتهای فوق انتخاب شده باشد. پس از نمایش، با فراخوانی تابع SetWindowLong به همراه پرچم GWL_STYLE ، می‌توان این حالت را تغییر داد. در ادامه، نحوه ایجاد و استفاده این کنترل را خواهیم آموخت:

۱. ایجاد کنترل، با فراخوانی CreateWindowEx به همراه SysListView32 بعنوان نام کلاس. حالت اولیه نمایش کنترل را در اینجا می‌توان تنظیم کرد.
۲. درصورت نیاز باید image lists ایجاد و مقدار دهی شوند.
۳. افزودن ستون‌ها به کنترل. (این مرحله در صورت استفاده از حالت report view ، ضروری است)
۴. افزودن آیتم‌ها و موارد مربوطه به کنترل.

ستون‌ها:

در یک listview ، یک یا چندین ستون وجود دارد. نحوه قرار گیری داده‌ها در این کنترل به صورت یک جدول به همراه سطر و ستون است. حداقل یک ستون باید در این کنترل تعریف شود. در سایر حالت‌های این گزارش بجز حالت

www.mohandesyar.com

report ، نیازی به اضافه کردن ستون وجود ندارد، زیرا در این حالت‌ها فقط و فقط یک ستون وجود دارد. با ارسال پیغام LVM_INSERTCOLUMN به کنترل، یک ستون جدید اضافه خواهد شد.

LVM_INSERTCOLUMN
wParam = iCol
lParam = pointer to a **LV_COLUMN** structure

iCol : شماره ستون آغاز شده از صفر است.

LV_COLUMN : حاوی اطلاعاتی در مورد ستون اضافه شده است. این ساختار به صورت زیر تعریف می‌شود:

```
LV_COLUMN  STRUCT
  imask dd ?
  fmt dd ?
  lx dd ?
  pszText dd ?
  cchTextMax dd ?
  iSubItem dd ?
  iImage dd ?
  iOrder dd ?
LV_COLUMN  ENDS
```

imask : مجموعه‌ای از پرچم‌ها که مشخص می کنند کدامیک از اعضای ساختار معتبر هستند (احتمال استفاده نشدن از تعدادی از اعضای ساختار وجود دارد). مقادیر معتبر آن به شرح زیر هستند:

LVCF_FMT = The **fmt** member is valid.
LVCF_SUBITEM = The **iSubItem** member is valid.
LVCF_TEXT = The **pszText** member is valid.
LVCF_WIDTH = The **lx** member is valid.

fmt : محل قرارگیری آیتم‌ها را درستون‌ها مشخص می کند. مقادیر مجاز آن به صورت زیر هستد:

LVCFMT_CENTER = Text is centered.
LVCFMT_LEFT = Text is left-aligned.
LVCFMT_RIGHT = Text is right-aligned.

lx : عرض ستون برحسب pixel . امکان تغییر آن بعدا بر اساس LVM_SETCOLUMNWIDTH میسر است.

pszText : اشاره گری است به نام ستون‌ها اگر از این اشاره گر برای تنظیم آنها استفاده شود.اگر از این ساختار برای دریافت خواص ستون استفاده شود، این فیلد اشاره گری خواهد بود به بافری که حاوی این نامها است. در این حالت باید اندازه ای بافر ذکر شده در عضو بعدی (cchTextMax) ذکر گردد. اگر قصد تنظیم متن آنرا دارید با توجه به اینکه ویندوز توانایی تعیین طول رشته‌های مختوم به نال را دارد، می توان از عضو cchTextMax صرفنظر کرد.

cchTextMax : در مورد آن پیشتر توضیح داده شد.

www.mohandesyar.com

iSubItem :اندیس subitem وابسته به این ستون را مشخص می‌کند. برای دریافت اطلاعات از این کنترل می‌توان پیغام LVM_GETCOLUMN را به همراه مقدار **LVCF_SUBITEM** در عضو **imask** به آن ارسال کرد.

iImage and iOrder : با IE 3.0 و یا بالاتر بکار می‌رود.

پس از ایجاد کنترل، باید یک یا چند ستون را بدان افزود. برای این مقصود باید اعضای ساختار LV_COLUMN را مقدار دهی کرده و سپس آنرا توسط پیغام LVM_INSERTCOLUMN به کنترل ارسال کرد.برای مثال :

```
LOCAL lvc:LV_COLUMN
mov lvc.imask,LVCF_TEXT+LVCF_WIDTH
mov lvc.pszText,offset Heading1
mov lvc.lx,150
invoke SendMessage,hList, LVM_INSERTCOLUMN,0,addr lvc
```

آیتم‌ها و ساب آیتم‌ها:

آیتم‌ها ، ورودی‌های اصلی در کنترل listview هستند. در حالتهای دیگر بجز report ، شما تنها آیتم‌ها را خواهید دید. ساب آیتم‌ها جزئیات آیتم‌ها هستند. هر آیتم می‌تواند یک یا چندین ساب آیتم داشته باشد. برای مثال اگر نام فایل یک آیتم باشد، ساب آیتم‌های آن می‌تواند اندازه فایل، تاریخ ایجاد آن و غیره باشد. ستونی که در سمت چپ قرار می‌گیرد، حاوی آیتم ها بوده و ساب آیتم‌ها در ستون‌های بعدی قرار خواهند گرفت. حداقل مورد نیاز، چند آیتم می‌باشد و ساب آیتم ها ضروری نیستند.

برای اضافه کردن آیتم‌ها به کنترل، از پیغام LVM_INSERTITEM استفاده می‌گردد. همچنین نیاز است تا آدرس ساختار LV_ITEM ، به lParam ارسال می‌شود. ساختار LV_ITEM به صورت زیر است:

```
LV_ITEM  STRUCT
  imask dd ?
  iItem dd ?
  iSubItem dd ?
  state dd ?
  stateMask dd ?
  pszText dd ?
  cchTextMax dd ?
  iImage dd ?
  lParam dd ?
  iIndent dd ?
LV_ITEM  ENDS
```


www.mohandesyar.com

imask : مجموعه ای از پرچم‌ها که مشخص می کنند کدامیک از اعضای این ساختار معتبر هستند. در حالت کلی این فیلد شبیه عضو imask ساختار LV_COLUMN است. به MSDN برای مشاهده جزئیات آن مراجعه بفرمایید.

iItem :اندیس آیتمی است که این ساختار به آن ارجاع دارد و از صفر شروع می‌شود.

iSubItem : اندیس ساب آیتمی است که توسط آیتم فوق مشخص شده است. به این فیلد بعنوان ستون یک جدول می‌توان نگریست. برای مثال اگر اولین آیتم مورد نظر را بخواهیم به کنترل اضافه کنیم، مقدار iItem مساوی صفر بوده و مقدار iSubItem نیز مساوی صفر خواهد بود. اگر یک کنترل listview دارای چهار ستون باشد، اولین ستون حاوی آیتم ها خواهد بود. مابقی ستون‌ها ساب آیتم‌ها می‌باشند. اگر نیاز به افزودن ساب آیتم به به چهارمین ستون کنترل باشد، مقدار iSubItem مساوی سه خواهد بود.

state :این فیلد حاوی پرچمی است که حالت یک آیتم را نشان می‌دهد. این حالت توسط برنامه و یا همچنین اعمال کاربران قابل تغییر است. برای مثال نمایانگر انتخاب شدن و غیره می‌باشد.همچنین می‌تواند بیانگر اندیس تصویر مورد استفاده (شروع شده از یک) جهت آیتم باشد.

stateMask : مشخص کننده این است که در فیلد state ، آیا از پرچم‌ها استفاده خواهد شد و یا اندیس تصاویر.

pszText : آدرس رشته‌ای مختوم به نال، بعنوان برچسب یک عنوان است. در حالتی که از این فیلد جهت دریافت مقادیر آن استفاده می‌شود، این فیلد باید آدرس بافری باشد که با برچسب آیتم مقداردهی می گردد.

cchTextMax : این فیلد تنها زمانی بکار می‌رود که از این ساختار جهت دریافت اطلاعاتی در مورد کنترل استفاده گردد. در این حالت این فیلد اندازه بافر مشخص شده در pszText برحسب بایت می‌باشد.

iImage :اندیس آیکونی است در یک image list که جهت آیتم بکار می‌رود.

lParam : مقداری است تعریف شده توسط کاربر که هنگام مرتب ساختن عناصر کنترل بکار می‌رود.هنگامیکه کنترل listview پیغام مرتب کردن عناصر را دریافت کند ، زوج آیتم‌ها را با هم مقایسه خواهد کرد.در این حالت مقدار lParam هر دو آیتم ارسال شده و بر این اساس می توان تصمیم گرفت که کدامیک در ابتدا نمایش داده شود.

بنابراین خلاصه مراحل افزودن آیتم‌ها و ساب آیتم‌ها به کنترل listview به صورت زیر است:

۱. ایجاد متغیری از نوع ساختار **LV_ITEM** .

www.mohandesyar.com

۲. پر کردن آن با اطلاعات لازم.
۳. فرستادن پیغام **LVM_INSERTITEM** به کنترل listview جهت افزودن آیتمی به آن. همچنین برای افزودن یک ساب آیتم، از پیغام **LVM_SETITEM** استفاده خواهد شد. در مورد مفهوم آیتم و ساب آیتم‌ها، پیشتر توضیح داده شد.

پیغام‌های listview:

پس از آموختن نحوه ایجاد یک کنترل listview ، گام بعدی تبادل اطلاعات با آن می‌باشد.این کنترل توسط پیغام WM_NOTIFY با پنجره والد ارتباط برقرار می کند.

مرتب سازی آیتم‌ها و ساب آیتم‌های یک کنترل listview:

نحوه مرتب سازی پیش فرض عناصر این کنترل را با مشخص کردن یکی از سبک‌های تابع CreateWindowEx ، به صورت LVS_SORTASCENDING و یا LVS_SORTDESCENDING ، می‌توان تعیین کرد. این نحوه مرتب سازی صرفا براساس متن برچسب آیتم‌ها انجام می‌شود. اگر نیاز به مرتب سازی آیتم‌ها به نحوی دیگر باشد، باید پیغام LVM_SORTITEMS به کنترل ارسال شود.

```
LVM_SORTITEMS
wParam = lParamSort
lParam = pCompareFunction
```

lParamSort: مقداری است تعریف شده توسط کاربر که به تابع مقایسه ارسال می‌شود.
pCompareFunction: آدرس تابع تعریف شده توسط کاربری است که کار مرتب کردن داده‌های دریافت شده را انجام خواهد داد. این تابع به صورت زیر تعریف می‌شود:

```
CompareFunc  proto  lParam1:DWORD, lParam2:DWORD, lParamSort:DWORD
```

lParam1 and **lParam2** : مقادیر عضو **lParam** ساختار **LV_ITEM** هستند که هنگام افزودن آیتمی به کنترل ، تعیین می‌شوند.

www.mohandesyar.com

lParamSort : مقدار wParam که با **LVM_SORTITEMS** ارسال می‌شوند.

هنگامیکه کنترل، پیغام LVM_SORTITEMS را دریافت می‌کند، تابع مقایسه مشخص شده در **lParam** را فراخوانی می‌نماید. به صورت خلاصه این تابع تصمیم می‌گیرد که کدامیک از دو مقدار ورودی آن باید در ابتدا قرار گیرند. اگر خروجی تابع منفی بود، یعنی اولین آیتم دریافتی توسط lParam1 ، بر دیگری مقدم است در غیر اینصورت خروجی مثبت خواهد بود. اگر دو آیتم با هم مساوی بودند خروجی تابع صفر می‌باشد.بنابراین مقدار مشخص شده در **lParam** ساختار **LV_ITEM** ، جهت عملکرد صحیح این روش مهم می‌باشد.

جهت بدست آوردن اطلاعاتی در مورد آیتم‌ها از پیغام **LVM_GETITEM** استفاده می‌گردد. هنگامیکه آیتم‌ها مرتب می‌شوند، اندیس آنها نیز تغییر می‌کند. بنابراین باید مقادیر lParam جهت انعکاس این تغییرات به روز شوند. زمانیکه یک کاربر بر روی سرتیتری جهت مرتب شدن آن ستون کلیک می‌کند ، پیغام **LVN_COLUMNCLICK** باید در رویه پنجره پردازش گردد. این پیغام از طریق پیغام WM_NOTIFY دریافت خواهد شد.

کد مثال ۲۸ :

```
.386
.model flat,stdcall
option casemap:none
include \masm32\include\windows.inc
include \masm32\include\user32.inc
include \masm32\include\kernel32.inc
include \masm32\include\comctl32.inc
includelib \masm32\lib\comctl32.lib
includelib \masm32\lib\user32.lib
includelib \masm32\lib\kernel32.lib

WinMain proto :DWORD,:DWORD,:DWORD,:DWORD

IDM_MAINMENU equ 10000
IDM_ICON equ LVS_ICON
IDM_SMALLICON equ LVS_SMALLICON
IDM_LIST equ LVS_LIST
IDM_REPORT equ LVS_REPORT

RGB macro red,green,blue
  xor eax,eax
  mov ah,blue
  shl eax,8
  mov ah,green
  mov al,red
endm
```


www.mohandesyar.com

```
.data
ClassName db "ListViewWinClass",0
AppName db "Testing a ListView Control",0
ListViewClassName db "SysListView32",0
Heading1 db "Filename",0
Heading2 db "Size",0
FileNamePattern db "*.*",0
FileNameSortOrder dd 0
SizeSortOrder dd 0
template db "%lu",0

.data?
hInstance HINSTANCE ?
hList dd ?
hMenu dd ?

.code
start:
  invoke GetModuleHandle, NULL
  mov hInstance,eax
  invoke WinMain, hInstance,NULL, NULL, SW_SHOWDEFAULT
  invoke ExitProcess,eax
  invoke InitCommonControls
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, NULL
  mov wc.lpfnWndProc, OFFSET WndProc
  mov wc.cbClsExtra,NULL
  mov wc.cbWndExtra,NULL
  push hInstance
  pop wc.hInstance
  mov wc.hbrBackground,COLOR_WINDOW+1
  mov wc.lpszMenuName,IDM_MAINMENU
  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,NULL,ADDR ClassName,\
                        ADDR AppName, WS_OVERLAPPEDWINDOW,CW_USEDEFAULT,\
                        CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT,\
                        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
```


www.mohandesyar.com

```
  ret
WinMain endp

InsertColumn proc
  LOCAL lvc:LV_COLUMN

  mov lvc.imask,LVCF_TEXT+LVCF_WIDTH
  mov lvc.pszText,offset Heading1
  mov lvc.lx,150
  invoke SendMessage,hList, LVM_INSERTCOLUMN, 0, addr lvc
  or lvc.imask,LVCF_FMT
  mov lvc.fmt,LVCFMT_RIGHT
  mov lvc.pszText,offset Heading2
  mov lvc.lx,100
  invoke SendMessage,hList, LVM_INSERTCOLUMN, 1 ,addr lvc
  ret
InsertColumn endp

ShowFileInfo proc uses edi row:DWORD, lpFind:DWORD
  LOCAL lvi:LV_ITEM
  LOCAL buffer[20]:BYTE
  mov edi,lpFind
  assume edi:ptr WIN32_FIND_DATA
  mov lvi.imask,LVIF_TEXT+LVIF_PARAM
  push row
  pop lvi.iItem
  mov lvi.iSubItem,0
  lea eax,[edi].cFileName
  mov lvi.pszText,eax
  push row
  pop lvi.lParam
  invoke SendMessage,hList, LVM_INSERTITEM,0, addr lvi
  mov lvi.imask,LVIF_TEXT
  inc lvi.iSubItem
  invoke wsprintf,addr buffer, addr template,[edi].nFileSizeLow
  lea eax,buffer
  mov lvi.pszText,eax
  invoke SendMessage,hList,LVM_SETITEM, 0,addr lvi
  assume edi:nothing
  ret
ShowFileInfo endp

FillFileInfo proc uses edi
  LOCAL finddata:WIN32_FIND_DATA
  LOCAL FHandle:DWORD

  invoke FindFirstFile,addr FileNamePattern,addr finddata
  .if eax!=INVALID_HANDLE_VALUE
    mov FHandle,eax
    xor edi,edi
    .while eax!=0
      test finddata.dwFileAttributes,FILE_ATTRIBUTE_DIRECTORY
      .if ZERO?
        invoke ShowFileInfo,edi, addr finddata
        inc edi
      .endif
      invoke FindNextFile,FHandle,addr finddata
    .endw
```


www.mohandesyar.com

```
    invoke FindClose,FHandle
  .endif
  ret
FillFileInfo endp

String2Dword proc uses ecx edi edx esi String:DWORD
  LOCAL Result:DWORD

  mov Result,0
  mov edi,String
  invoke lstrlen,String
  .while eax!=0
    xor edx,edx
    mov dl,byte ptr [edi]
    sub dl,"0"
    mov esi,eax
    dec esi
    push eax
    mov eax,edx
    push ebx
    mov ebx,10
    .while esi > 0
      mul ebx
      dec esi
    .endw
    pop ebx
    add Result,eax
    pop eax
    inc edi
    dec eax
  .endw
  mov eax,Result
  ret
String2Dword endp

CompareFunc proc uses edi lParam1:DWORD, lParam2:DWORD, SortType:DWORD
  LOCAL buffer[256]:BYTE
  LOCAL buffer1[256]:BYTE
  LOCAL lvi:LV_ITEM

  mov lvi.imask,LVIF_TEXT
  lea eax,buffer
  mov lvi.pszText,eax
  mov lvi.cchTextMax,256
  .if SortType==1
    mov lvi.iSubItem,1
    invoke SendMessage,hList,LVM_GETITEMTEXT,lParam1,addr lvi
    invoke String2Dword,addr buffer
    mov edi,eax
    invoke SendMessage,hList,LVM_GETITEMTEXT,lParam2,addr lvi
    invoke String2Dword,addr buffer
    sub edi,eax
    mov eax,edi
  .elseif SortType==2
    mov lvi.iSubItem,1
    invoke SendMessage,hList,LVM_GETITEMTEXT,lParam1,addr lvi
    invoke String2Dword,addr buffer
    mov edi,eax
```


www.mohandesyar.com

```
    invoke SendMessage,hList,LVM_GETITEMTEXT,lParam2,addr lvi
    invoke String2Dword,addr buffer
    sub eax,edi
  .elseif SortType==3
    mov lvi.iSubItem,0
    invoke SendMessage,hList,LVM_GETITEMTEXT,lParam1,addr lvi
    invoke lstrcpy,addr buffer1,addr buffer
    invoke SendMessage,hList,LVM_GETITEMTEXT,lParam2,addr lvi
    invoke lstrcmpi,addr buffer1,addr buffer
  .else
    mov lvi.iSubItem,0
    invoke SendMessage,hList,LVM_GETITEMTEXT,lParam1,addr lvi
    invoke lstrcpy,addr buffer1,addr buffer
    invoke SendMessage,hList,LVM_GETITEMTEXT,lParam2,addr lvi
    invoke lstrcmpi,addr buffer,addr buffer1
  .endif
  ret
CompareFunc endp

UpdatelParam proc uses edi
   LOCAL lvi:LV_ITEM

   invoke SendMessage,hList, LVM_GETITEMCOUNT,0,0
   mov edi,eax
   mov lvi.imask,LVIF_PARAM
   mov lvi.iSubItem,0
   mov lvi.iItem,0
   .while edi>0
     push lvi.iItem
     pop lvi.lParam
     invoke SendMessage,hList, LVM_SETITEM,0,addr lvi
     inc lvi.iItem
     dec edi
   .endw
   ret
UpdatelParam endp

ShowCurrentFocus proc
   LOCAL lvi:LV_ITEM
   LOCAL buffer[256]:BYTE

   invoke SendMessage,hList,LVM_GETNEXTITEM,-1, LVNI_FOCUSED
   mov lvi.iItem,eax
   mov lvi.iSubItem,0
   mov lvi.imask,LVIF_TEXT
   lea eax,buffer
   mov lvi.pszText,eax
   mov lvi.cchTextMax,256
   invoke SendMessage,hList,LVM_GETITEM,0,addr lvi
   invoke MessageBox,0, addr buffer,addr AppName,MB_OK
   ret
ShowCurrentFocus endp

WndProc proc hWnd:HWND, uMsg:UINT, wParam:WPARAM, lParam:LPARAM
  .if uMsg==WM_CREATE
    invoke CreateWindowEx, NULL, addr ListViewClassName,\
                           NULL, LVS_REPORT+WS_CHILD+WS_VISIBLE, 0,0,\
```


www.mohandesyar.com

```
                                0,0,hWnd, NULL, hInstance, NULL
  mov hList, eax
  invoke InsertColumn
  invoke FillFileInfo
  RGB 255,255,255
  invoke SendMessage,hList,LVM_SETTEXTCOLOR,0,eax
  RGB 0,0,0
  invoke SendMessage,hList,LVM_SETBKCOLOR,0,eax
  RGB 0,0,0
  invoke SendMessage,hList,LVM_SETTEXTBKCOLOR,0,eax
  invoke GetMenu,hWnd
  mov hMenu,eax
  invoke CheckMenuRadioItem,hMenu,IDM_ICON,IDM_LIST, IDM_REPORT,MF_CHECKED
.elseif uMsg==WM_COMMAND
  .if lParam==0
    invoke GetWindowLong,hList,GWL_STYLE
    and eax,not LVS_TYPEMASK
    mov edx,wParam
    and edx,0FFFFh
    push edx
    or eax,edx
    invoke SetWindowLong,hList,GWL_STYLE,eax
    pop edx
    invoke CheckMenuRadioItem,hMenu,IDM_ICON,IDM_LIST, edx,MF_CHECKED
  .endif
.elseif uMsg==WM_NOTIFY
  push edi
  mov edi,lParam
  assume edi:ptr NMHDR
  mov eax,[edi].hwndFrom
  .if eax==hList
    .if [edi].code==LVN_COLUMNCLICK
      assume edi:ptr NM_LISTVIEW
      .if [edi].iSubItem==1
        .if SizeSortOrder==0 || SizeSortOrder==2
          invoke SendMessage,hList,LVM_SORTITEMS,1,addr CompareFunc
          invoke UpdatelParam
          mov SizeSortOrder,1
        .else
          invoke SendMessage,hList,LVM_SORTITEMS,2,addr CompareFunc
          invoke UpdatelParam
          mov SizeSortOrder,2
        .endif
      .else
        .if FileNameSortOrder==0 || FileNameSortOrder==4
          invoke SendMessage,hList,LVM_SORTITEMS,3,addr CompareFunc
          invoke UpdatelParam
          mov FileNameSortOrder,3
        .else
          invoke SendMessage,hList,LVM_SORTITEMS,4,addr CompareFunc
          invoke UpdatelParam
          mov FileNameSortOrder,4
        .endif
      .endif
      assume edi:ptr NMHDR
    .elseif [edi].code==NM_DBLCLK
      invoke ShowCurrentFocus
    .endif
```


www.mohandesyar.com

```
    .endif
    pop edi
  .elseif uMsg==WM_SIZE
    mov eax,lParam
    mov edx,eax
    and eax,0ffffh
    shr edx,16
    invoke MoveWindow,hList, 0, 0, eax,edx,TRUE
  .elseif uMsg==WM_DESTROY
    invoke PostQuitMessage,NULL
  .else
    invoke DefWindowProc,hWnd,uMsg,wParam,lParam
    ret
  .endif
  xor eax,eax
  ret
WndProc endp
end start
```

بررسی کد فوق:

اولین کار ایجاد کنترل listview می‌باشد:

```
  .if uMsg==WM_CREATE
    invoke CreateWindowEx, NULL, addr ListViewClassName, NULL,\

                           LVS_REPORT+WS_CHILD+WS_VISIBLE, 0,0,0,\

                           0,hWnd, NULL, hInstance, NULL
    mov hList, eax
```

تابع CreateWindowEx ، به همراه نام کلاس "SysListView32" ، با طرز نمایش پیش فرض LVS_REPORT ، فراخوانی گردیده است. در ادامه:

```
    invoke InsertColumn
```

پس از ایجاد کنترل، ستونهایی را به آن خواهیم افزود.سپس :

```
  LOCAL lvc:LV_COLUMN

  mov lvc.imask,LVCF_TEXT+LVCF_WIDTH
  mov lvc.pszText,offset Heading1
  mov lvc.lx,150
  invoke SendMessage,hList, LVM_INSERTCOLUMN, 0, addr lvc
```

در اینجا برچسب و عرض ستون اول ، جهت ذخیره کردن نام فایلها، در ساختار LV_COLUMN مشخص می‌شوند. بنابراین عضو **imask** باید با پرچمهای **LVCF_TEXT** and **LVCF_WIDTH** مقدار دهی گردد.عضو **pszText** با آدرس متن برچسب و **lx** با عرض ستون برحسب pixel ، مقداردهی می‌شوند.پس از آن با ارسال پیغام LVM_INSERTCOLUMN به کنترل، ساختار فوق را جهت افزودن ستونی جدید به آن خواهیم فرستاد. در ادامه:

www.mohandesyar.com

```
or lvc.imask,LVCF_FMT
mov lvc.fmt,LVCFMT_RIGHT
```

پس از افزودن اولین ستون، ستون دیگری را جهت ذخیره سازی اندازه فایلها ایجاد می کنیم. از آنجائیکه می‌خواهیم اندازه فایلها از سمت راست نمایش داده شوند، باید پرچم LVCFMT_RIGHT در عضو fmt قرار گیرد. همچنین پرچم LVCF_FMT نیز باید در imask ذکر گردد. سپس :

```
mov lvc.pszText,offset Heading2
mov lvc.lx,100
invoke SendMessage,hList, LVM_INSERTCOLUMN, 1 ,addr lvc
```

در ادامه آدرس برچسب در pszText قرار گرفته و مقدار عرض آن با lx تعیین شده است.سپس این اطلاعات از طریق پیغام LVM_INSERTCOLUMN به کنترل ارسال می گردد. پس از اینکه ستون ها ایجاد شدند می‌توان آنها را با اطلاعات مورد نظر مقدار دهی کرد:

```
invoke FillFileInfo
```

این تابع به صورت زیر تعریف شده است:

```
FillFileInfo proc uses edi
LOCAL finddata:WIN32_FIND_DATA
LOCAL FHandle:DWORD

invoke FindFirstFile,addr FileNamePattern,addr finddata
```

با فراخوانی تابع FindFirstFile ، اولین فایلی که با شرایط جستجو مطابقت داشته باشد ، یافت می‌شود. این تابع به صورت زیر تعریف می گردد:

```
FindFirstFile  proto pFileName:DWORD, pWin32_Find_Data:DWORD
```

pFileName : آدرس نام فایل جهت جستجو است. این رشته می تواند حاوی wildcards نیز باشد. برای مثال از *.* جهت جستجوی تمامی فایلها در دایرکتوری جاری استفاده گردید.

pWin32_Find_Data : آدرس ساختار **WIN32_FIND_DATA** است که اعضای آن با اطلاعات فایل یافت شده پر می گردند.

در صورتیکه فایلی یافت نشود، خروجی تابع که در eax قرار می گیرد مساوی **INVALID_HANDLE_VALUE** خواهد بود. در غیراینصورت خروجی آن دستگیره جستجویی است که توسط **FindNextFile** های متعاقب استفاده می گردد. سپس :

```
.if eax!=INVALID_HANDLE_VALUE
  mov FHandle,eax
  xor edi,edi
```


www.mohandesyar.com

اگر فایلی پیدا شود، دستگیره جستجو را در متغییری ذخیره کرده و سپس مقدار edi را که بعنوان اندیسی به آیتم ها بکار خواهد رفت (تعداد ردیف‌ها) ، صفر می کنیم. در ادامه :

```
.while eax!=0
  test finddata.dwFileAttributes,FILE_ATTRIBUTE_DIRECTORY
  .if ZERO?
```

در این مثال کاری با فولدرها نخواهیم داشت، بنابراین آنها را با بررسی مقدار **dwFileAttributes** فیلتر خواهیم کرد (FILE_ATTRIBUTE_DIRECTORY).سپس :

```
    invoke ShowFileInfo,edi, addr finddata
    inc edi
  .endif
  invoke FindNextFile,FHandle,addr finddata
.endw
```

با فراخوانی تابع ShowFileInfo ، نام و اندازه فایلها را به کنترل اضافه خواهیم کرد.سپس شماره ردیف جاری را در edi افزایش خواهیم داد. در ادامه فراخوانی FindNextFile را تا زمانیکه تابع صفر برگرداند (دیگر فایلی وجود نداشته باشد)، ادامه خواهیم داد. سپس :

```
  invoke FindClose,FHandle
.endif
ret
FillFileInfo endp
```

پس از اینکه تمامی فایلهای موجود در دایرکتوری جاری مورد بررسی قرار گرفتند، دستگیره جستجو باید بسته شود. تابع **ShowFileInfo** دو پارامتر را می‌پذیرد. اندیس آیتم مورد نظر (شماره ردیف) و آدرس ساختار WIN32_FIND_DATA . در ادامه :

```
ShowFileInfo proc uses edi row:DWORD, lpFind:DWORD
  LOCAL lvi:LV_ITEM
  LOCAL buffer[20]:BYTE
  mov edi,lpFind
  assume edi:ptr WIN32_FIND_DATA
```

ذخیره سازی آدرس WIN32_FIND_DATA در edi انجام گردیده است. سپس :

```
  mov lvi.imask,LVIF_TEXT+LVIF_PARAM
  push row
  pop lvi.iItem
  mov lvi.iSubItem,0
```

از آنجائیکه برچسب آیتم و مقدار lParam را تامین خواهیم کرد، پرچم‌های **LVIF_TEXT** and **LVIF_PARAM** را در imask قرار خواهیم داد.در ادامه iItem به صفر تنظیم شده (زیرا آیتم اصلی است) . سپس :

www.mohandesyar.com

```
lea eax,[edi].cFileName
mov lvi.pszText,eax
push row
pop lvi.lParam
```

در ادامه آدرس برچسب را که در اینجا نام فایل می باشد ، در عضو pszText ساختار **WIN32_FIND_DATA** ، قرار می‌دهیم. از آنجائیکه ما ذخیره سازی داده‌ها را در کنترل listview پیاده سازی می‌کنیم، باید مقداری را در **lParam** قرار دهیم. در اینجا تصمیم گرفته شده است تا شماره ردیف را در این عضو قرار دهیم. بنابراین اطلاعات مربوط به آیتم‌ها را می‌توان با استفاده از اندیس آنها بدست آورد. در ادامه :

```
invoke SendMessage,hList, LVM_INSERTITEM,0, addr lvi
```

زمانیکه تمام اعضای LV_ITEM مقدار دهی شدند، پیغام **LVM_INSERTITEM** را به کنترل listview جهت افزودن آیتم جدیدی، ارسال می‌نمائیم. سپس :

```
mov lvi.imask,LVIF_TEXT
inc lvi.iSubItem
invoke wsprintf,addr buffer, addr template,[edi].nFileSizeLow
lea eax,buffer
mov lvi.pszText,eax
```

در ادامه ساب آیتم مربوط به ستون دوم را تنظیم خواهیم کرد. یک ساب آیتم تنها می‌تواند دارای یک برچسب باشد. بنابراین مقدار imask را مساوی **LVIF_TEXT** قرار خواهیم داد. سپس ستونی را مشخص خواهیم کرد که ساب آیتم متعلق به آن است. در این حالت با افزودن مقدار **iSubItem** ، آنرا به یک تنظیم خواهیم کرد. برچسبی را که استفاده خواهیم کرد، اندازه فایل است. جهت استفاده از آن، باید آنرا با استفاده از تابع **wsprintf** به رشته تبدیل کرد. سپس آدرس آن رشته را در pszText قرار خواهیم داد. در ادامه:

```
  invoke SendMessage,hList,LVM_SETITEM, 0,addr lvi
  assume edi:nothing
  ret
ShowFileInfo endp
```

زمانیکه تمام فیلدهای لازم **LV_ITEM** ، مقدار دهی شدند ، آدرس این ساختار را توسط پیغام **LVM_SETITEM** به listview ارسال خواهیم کرد.

پس از نمایش تمام آیتم‌ها و ساب آیتم‌های کنترل، رنگ زمینه و متن را تعویض خواهیم کرد:

```
RGB 255,255,255
invoke SendMessage,hList,LVM_SETTEXTCOLOR,0,eax
RGB 0,0,0
invoke SendMessage,hList,LVM_SETBKCOLOR,0,eax
RGB 0,0,0
invoke SendMessage,hList,LVM_SETTEXTBKCOLOR,0,eax
```


www.mohandesyar.com

از RGB macro برای ترکیب رنگ‌های قرمز، سبز و آبی و قرار دادن آن در eax ، کمک گرفته شد. تغییر رنگ متن و زمینه یک متن با ارسال پیغامهای **LVM_SETTEXTCOLOR** and **LVM_SETTEXTBKCOLOR** انجام می‌شود.برای تغییر رنگ زمینه listview از پیغام **LVM_SETBKCOLOR** استفاده می‌شود:

```
invoke GetMenu,hWnd
mov hMenu,eax
invoke CheckMenuRadioItem,hMenu,IDM_ICON,IDM_LIST, IDM_REPORT,MF_CHECKED
```

در برنامه امکان انتخاب نحوه مشاهده آیتم‌ها توسط منوها وجود دارد. بنابراین باید دستگیره منو را در ابتدا بدست آورد. برای اینکه ردیابی تغییرات انتخاب شده توسط کاربر ساده باشد، از دکمه‌های رادیویی در منوها استفاده گردید. به همین دلیل تابع CheckMenuRadioItem فراخوانی می گردد.

لازم به ذکر است که در ابتدا طول و عرض کنترل listview به صفر تنظیم گردیده است. این کنترل زمانیکه پنجره والد تغییر اندازه داد، به اندازه اصلی خود درخواهد آمد. به این صورت می‌توان اطمینان حاصل کرد که اندازه کنترل listview همواره با اندازه پنجره والد یکی خواهد بود و تمام پنجره را می‌پوشاند:

```
.elseif uMsg==WM_SIZE
  mov eax,lParam
  mov edx,eax
  and eax,0ffffh
  shr edx,16
  invoke MoveWindow,hList, 0, 0, eax,edx,TRUE
```

زمانیکه پنجره والد پیغام WM_SIZE را دریافت می کند، low word مربوط به lParam عرض جدید ناحیه کلاینت بوده و high word آن مساوی ارتفاع جدید است. سپس با فراخوانی MoveWindow ، کنترل listview را به اندازه کل ناحیه کلاینت در می‌آوریم.

هنگامیکه کاربر از طریق منو نحوه نمایش آیتم ها را انتخاب می کند با استفاده از تابع SetWindowLong ، سبک جدیدی را به کنترل listview اعمال خواهیم کرد.

```
.elseif uMsg==WM_COMMAND
  .if lParam==0
    invoke GetWindowLong,hList,GWL_STYLE
    and eax,not LVS_TYPEMASK
```

در ابتدا سبک جاری نمایش دریافت شده و سپس پرچم بازگشت داده شده آن را پاک خواهیم کرد.ثابت LVS_TYPEMASK ، ترکیبی است از ۴ مقدار ممکن برای نحوه نمایش آیتم ها در کنترل listview ، یعنی :

LVS_ICON+LVS_SMALLICON+LVS_LIST+LVS_REPORT

بنابراین وقتی عملیات and بر روی پرچم سبک جاری و مقدار not LVS_TYPEMASK انجام شود، این پرچم پاک خواهد شد.

جهت سهولت کار در این مثال هنگام طراحی منوها از ثوابت نحوه نمایش آیتم‌ها بعنوان menu IDs استفاده گردید.

www.mohandesyar.com

```
IDM_ICON equ LVS_ICON
IDM_SMALLICON equ LVS_SMALLICON
IDM_LIST equ LVS_LIST
IDM_REPORT equ LVS_REPORT
```

بنابراین زمانیکه پنجره والد پیغام WM_COMMAND را دریافت کند، سبک مورد نظر در low word مربوط به wParam آی دی منو قرار خواهد داشت . سپس high word مربوط به wParam صفر شده :

```
mov edx,wParam
and edx,0FFFFh
```

و سبک دلخواه به سبک فعلی اضافه می گردد. عملیات تنظیم این مقدار توسط تابع SetWindowLong انجام خواهد شد :

```
push edx
or eax,edx

invoke SetWindowLong,hList,GWL_STYLE,eax
```

سپس :

```
pop edx
invoke CheckMenuRadioItem,hMenu,IDM_ICON,IDM_LIST, edx,MF_CHECKED
.endif
```

همچنین نیاز است تا کنار آیتم منوی انتخابی ، یک دکمه رادیویی قرار گیرد، بنابراین از تابع CheckMenuRadioItem به همراه آرگومان مناسب مربوطه استفاده شد.

هنگامیکه کاربر بر روی سرتیتر یک ستون کلیک نماید نیاز است تا اعضای آن ستون مرتب شوند. بنابراین پیغام WM_NOTIFY را پردازش خواهیم کرد.

```
.elseif uMsg==WM_NOTIFY
  push edi
  mov edi,lParam
  assume edi:ptr NMHDR
  mov eax,[edi].hwndFrom
  .if eax==hList
```

هنگام دریافت این پیغام، lParam حاوی اشاره گری به ساختار NMHDR می‌باشد.با مقایسه مقدار عضو hwndFrom ساختار NMHDR با دستگیره کنترل listview ، درخواهیم یافت که آیا این پیغام از طرف کنترل صادر شده است یا خیر؟ در ادامه :

```
.if [edi].code==LVN_COLUMNCLICK
  assume edi:ptr NM_LISTVIEW
```


www.mohandesyar.com

اگر پیغام رسیده از جانب کنترل باشد، بررسی خواهیم کرد که آیا code با LVN_COLUMNCLICK مساوی است یا خیر؟ اگر این شرط صادق باشد بدین معنا است که کاربر بر روی سرتیتر یک ستون در کنترل listview کلیک کرده است. در این حالت می‌توان فرض کرد که lParam حاوی اشاره گری است به ساختار NM_LISTVIEW . در ادامه نیاز است تا بررسی شود که کاربر بر روی کدام ستون کلیک کرده است.با بررسی iSubItem این مطلب روشن خواهد شد. مقدار این عضو بیانگر شماره ستون های شروع شده از صفر است.

```
.if [edi].iSubItem==1
  .if SizeSortOrder==0 || SizeSortOrder==2
```

از متغیرهای حالت جهت نگهداری نحوه مرتب شدن ستون‌ها استفاده خواهیم کرد.صفر به معنای این است که ستون هنوز مرتب نشده است ، یک به معنای مرتب شدن صعودی و دو به معنای مرتب شدن نزولی است.در ادامه:

```
invoke SendMessage,hList,LVM_SORTITEMS,1,addr CompareFunc
```

در اینجا آدرس تابع مرتب سازی به کنترل ارسال می گردد. تابع مقایسه به شکل زیر است:

```
CompareFunc proc uses edi lParam1:DWORD, lParam2:DWORD, SortType:DWORD
  LOCAL buffer[256]:BYTE
  LOCAL buffer1[256]:BYTE
  LOCAL lvi:LV_ITEM

  mov lvi.imask,LVIF_TEXT
  lea eax,buffer
  mov lvi.pszText,eax
  mov lvi.cchTextMax,256
```

دو آیتمی که باید مقایسه شوند توسط lParam1 و lParam2 مربوط به LV_ITEM ، توسط کنترل به این تابع ارسال می‌شوند. لازم به یادآوری است که اندیس آیتم‌ها را در lParam قرار داده بودیم. به این طریق اطلاعات مورد نیاز که مربوط به برچسب آیتم‌ها و ساب آیتم‌ها جهت مرتب سازی است ، قابل دریافت هستند. بنابراین ساختار LV_ITEM را با مشخص کردن LVIF_TEXT بعنوان imask ، آدرس بافر در pszText و اندازه بافر در cchTextMax ، مهیا خواهیم ساخت. سپس :

```
.if SortType==1
  mov lvi.iSubItem,1
  invoke SendMessage,hList,LVM_GETITEMTEXT,lParam1,addr lvi
```

اگر مقدار SortType مساوی یک یا دو باشد، مشخص خواهد شد که بر روی ستون اندازه کلیک شده است. یک به معنای مرتب سازی صعودی است و دو برعکس. بنابراین iSubItem را مساوی یک قرار داده (جهت انتخاب ستون اندازه) و پیغام LVM_GETITEMTEXT را به کنترل listview ، جهت دریافت برچسب ساب آیتم‌ها ارسال خواهیم کرد. در ادامه :

www.mohandesyar.com

```
invoke String2Dword,addr buffer
mov edi,eax
```

توسط تابع String2Dword ، رشته اندازه را به مقداری از نوع dword تبدیل می کند و خروجی آن در eax قرار خواهد گرفت.این مقدار را در edi جهت مقایسه بعدی ذخیره خواهیم کرد.

```
invoke SendMessage,hList,LVM_GETITEMTEXT,lParam2,addr lvi
invoke String2Dword,addr buffer
sub edi,eax
mov eax,edi
```

نحوه عملکرد و خروجی‌های تابع مقایسه نیز پیشتر معرفی شدند. در ادامه :

```
.elseif SortType==3
  mov lvi.iSubItem,0
  invoke SendMessage,hList,LVM_GETITEMTEXT,lParam1,addr lvi
  invoke lstrcpy,addr buffer1,addr buffer
  invoke SendMessage,hList,LVM_GETITEMTEXT,lParam2,addr lvi
  invoke lstrcmpi,addr buffer1,addr buffer
```

در حالتیکه کاربر بر روی سرتیتر ستون نام فایلها کلیک کند باید آیتم‌ها بر اساس نام فایلها مرتب شوند. مقایسه نامها پس از دریافت نام آنها، با استفاده از تابع lstrcmpi انجام می‌شود.

پس از مرتب شدن اعضای یک ستون نیاز است تا اندیس آنها توسط lParam نیز به روز شوند. اینکار با فراخوانی تابع UpdatelParam انجام خواهد شد، در غیراینصورت مرتب سازی بعدی کار نخواهد کرد :

```
invoke UpdatelParam
mov SizeSortOrder,1
```

در ادامه :

```
.elseif [edi].code==NM_DBLCLK
  invoke ShowCurrentFocus
.endif
```

هنگامیکه کاربر دوبار بر روی آیتمی کلیک نماید، برچسب آن به همراه یک پیغام به کاربر نمایش داده خواهد شد. بنابراین باید مقدار code مربوط به NMHDR را بررسی نمائیم. در ادامه :

```
ShowCurrentFocus proc
  LOCAL lvi:LV_ITEM
  LOCAL buffer[256]:BYTE

  invoke SendMessage,hList,LVM_GETNEXTITEM,-1, LVNI_FOCUSED
```

برای تشخیص دوبار کلیک شدن بر روی یک آیتم، باید آیتمی که تمرکز را در اختیار گرفته است، یافت. اینکار را با فرستادن پیغام LVM_GETNEXTITEM به همراه حالتی دلخواه در lParam ، می‌توان انجام داد. مساوی 1- بودن wParam به این معنا است که تمام آیتم ها جستجو خواهند شد. اندیس آیتم در eax بازگشت داده خواهد شد. سپس :

```
mov lvi.iItem,eax
mov lvi.iSubItem,0
mov lvi.imask,LVIF_TEXT
```


www.mohandesyar.com

```
lea eax,buffer
mov lvi.pszText,eax
mov lvi.cchTextMax,256
invoke SendMessage,hList,LVM_GETITEM,0,addr lvi
```

برای دریافت برچسب آیتم، پیغام LVM_GETITEM را به کنترل ارسال خواهیم کرد.

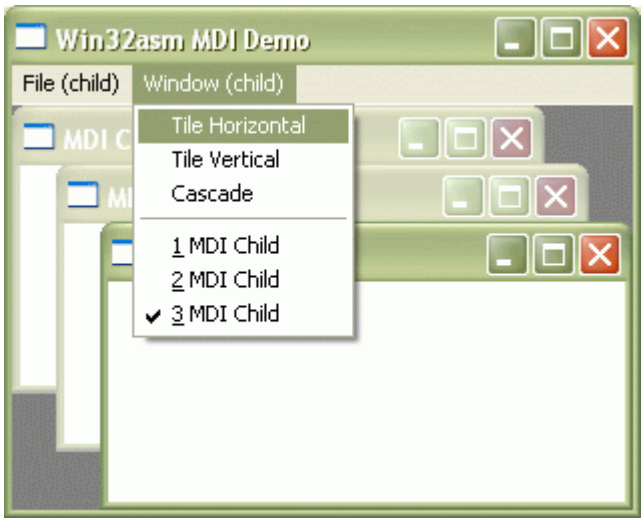
```
invoke MessageBox,0, addr buffer,addr AppName,MB_OK
```

در آخر (!) برچسب را در یک message box نمایش می‌دهیم.

نحوه اضافه کردن آیکون به این کنترل همانند روشی مشابه برای کنترل tree view است.

www.mohandesyar.com

مثال ۲۹- Multiple Document Interface (MDI)



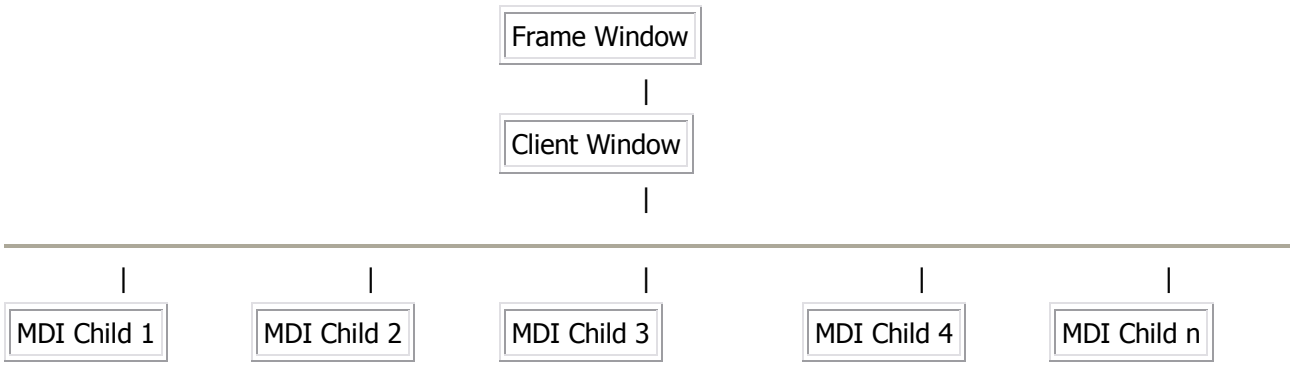
برنامه های MDI توانایی کار با چندین سند را به صورت همزمان دارند. برای مثال برنامه ای مانند notepad ویندوز یک برنامه SDI است. این برنامه در هر زمان تنها توانایی کار با یک سند را دارد. برخلاف آن، Microsoft Word توانایی کار با چندین سند را در آن واحد دارا است.

برنامه های MDI دارای چندین ویژگی خاص خود هستند:

- به همراه پنجره اصلی، امکان وجود چندین پنجره فرزند در ناحیه کلاینت پنجره اصلی میسر است.
- در اینجا هنگام به حداقل رساندن اندازه یک پنجره فرزند، این پنجره در ناحیه سمت چپ پایین پنجره اصلی به حداقل خواهد رسید.
- هنگام به حداکثر رساندن اندازه یک پنجره فرزند، کل ناحیه اصلی را پوشانده و عنوان آن با عنوان پنجره اصلی یکی خواهد شد.
- امکان بستن پنجره فرزند با فشردن Ctrl+F4 میسر است.

به پنجره اصلی که حاوی پنجره‌های فرزند است، frame window گفته می‌شود.برای کنترل تعداد دلخواهی پنجره فرزند، نیاز به پنجره ویژه ای به نام client window است. این پنجره را می‌توان همانند پنجره ای شفاف که تمام ناحیه کلاینت را پوشانده است تصور کرد و والد اصلی پنجره های فرزند محسوب می‌گردد. (شکل زیر این سلسله مراتب را نشان می‌دهد)

www.mohandesyar.com



ایجاد پنجره چهارچوب (frame) :

این پنجره همانند یک پنجره متداول با فراخوانی CreateWindowEx تولید می گردد. اما دو تفاوت عمده با یک پنجره متداول دارد. تفاوت اول این است که در اینجا باید DefFrameProc بجای DefWindowProc ، جهت پردازش پیغامهایی که رویه پنجره مایل به پردازش آنها نیست استفاده گردد.این روشی است که به ویندوز اجازه داده می دهد تا کار مدیریت یک برنامه MDI را بعهده بگیرد. (این فراخوانی اجباری است). تعریف این تابع به صورت زیر است:

```
DefFrameProc    proc    hwndFrame:DWORD,
                        hwndClient:DWORD,
                        uMsg:DWORD,
                        wParam:DWORD,
                        lParam:DWORD
```

این تابع در مقایسه با تابع DefWindowProc یک آرگومان بیشتر دارد. پارامتر اضافی دستگیره ای است به پنجره کلاینت. این دستگیره ضروری بوده و توسط آن امکان ارسال پیغامهای مرتبط با MDI به پنجره کلاینت میسر می‌شود.

دومین تفاوت ضرورت فراخوانی تابع TranslateMDISysAccel ، در حلقه پیغامهای پنجره چهارچوب است. این تابع جهت پردازش فشرده شدن کلیدهایی مانند Ctrl+F4, Ctrl+Tab ضروری است و به صورت زیر تعریف می گردد:

```
TranslateMDISysAccel proc hwndClient:DWORD,
                          lpMsg:DWORD
```

اولین پارامتر آن دستگیره‌ای است به پنجره کلاینت (والد تمامی پنجره های فرزند) . پارامتر دوم آدرس ساختار MSG است و با GetMessage پر می‌شود و از آن برای بررسی فشردن شدن کلیدهای مرتبط با MDI استفاده می‌شود.در این صورت پیغام را پردازش کرده و عددی غیر صفر را بر می گرداند. در غیراینصورت خروجی صفر خواهد بود.

مراحل ایجاد یک پنجره چهارچوب به شرح زیر است:

۱. مقدار دهی اعضای WNDCLASSEX به صورت معمول.

www.mohandesyar.com

۲. رجیستر کردن کلاس پنجره چهارچوب توسط RegisterClassEx .
۳. ایجاد پنجره چهارچوب با فراخوانی CreateWindowEx .
۴. فراخوانی TranslateMDISysAccel به همراه حلقه پیغامها.
۵. ارسال پیغامهای پردازش نشده به DefFrameProc ، درون رویه پنجره.

ایجاد پنجره کلاینت :

پس از ایجاد پنجره چهارچوب، نوبت به ایجاد پنجره کلاینت می‌رسد. کلاس این پنجره توسط ویندوز رجیستر شده و دارای نام استاندارد MDICLIENT است. همچنین نیاز است تا آدرس CLIENTCREATESTRUCT را به تابع CreateWindowEx ارسال کرد. این ساختار به صورت زیر تعریف می‌شود:

```
CLIENTCREATESTRUCT struct
        hWindowMenu     dd ?
        idFirstChild    dd ?
CLIENTCREATESTRUCT ends
```

hWindowMenu : دستگیره‌ای است به ساب منویی که لیست نامهای پنجره‌های فرزند MDI به آن اضافه خواهند شد. این ویژگی نیاز به کمی توضیح بیشتر دارد. برای مثال اگر به برنامه MDI ایی مانند مایکروسافت ورد مراجعه نمایید، منویی به نام Window در آن تعریف شده است که لیست پنجره های باز در آنرا نمایش می دهد. این موارد به صورت درونی توسط ویندوز مدیریت می‌شود.در اینجا تنها باید دستگیره ساب منو را مشخص نمود و مابقی کارها توسط ویندوز صورت می گیرد.لازم به ذکر است که این ساب منو هر نامی می‌تواند داشته باشد.اگر نیازی به لیست کردن پنجره‌ها نباشد تنها کافی است hWindowMenu را به NULL مقداردهی نمائید.برای بدست آوردن دستگیره ساب منو از تابع GetSubMenu استفاده می‌شود.

idFirstChild : ID اولین پنجره فرزند MDI است. برای مثال اگر این فیلد را با ۱۰۰ مقدار دهی نمایید، ID اولین پنجره فرزند مساوی ۱۰۰ ، ID دومین پنجره فرزند مساوی ۱۰۱ خواهد بود و الی آخر.هنگامیکه پنجره ای از طریق منوی window انتخاب شود، ID آن از طریق پیغام WM_COMMAND، به پنجره چهارچوب ارسال می گردد. بطور متداول این پیغام مدیریت نشده به تابع DefFrameProc ارسال می گردد.از کلمه مدیریت نشده در اینجا بدین جهت استفاده گردید که آیتم‌های منو window توسط برنامه ما ایجاد نشده است. بنابراین برنامه از ID آنها بی خبر بوده و روشی را برای پردازش آنها پیش روی ندارد. در برنامه های MDI جهت پردازش این پیغام باید به صورت زیر عمل کرد:

www.mohandesyar.com

```
.elseif uMsg==WM_COMMAND
    .if lParam==0                    ; this message is generated from a menu
       mov eax,wParam
       .if ax==IDM_CASCADE
            .....
       .elseif ax==IDM_TILEVERT
            .....
       .else
            invoke DefFrameProc, hwndFrame, hwndClient, uMsg,wParam, lParam
            ret
       .endif
```

بطور معمول از پیغامهای رسیده از حالتهای مدیریت نشده ، صرفنظر می‌شود. اما در حالتهای MDI ، اگر از این پیغامها صرفنظر شود ، دیگر امکان پردازش پیغامهای ساب منوی window وجود نخواهد داشت. این پیغامها باید به DefFrameProc ارسال شوند تا بصورت صحیح بتوان آنها را مدیریت کرد.

تذکر:

در مورد مقدار idFirstChild باید دقت کرد. این مقدار نباید مساوی صفر قرار گیرد. درغیراینصورت آیتم‌های منو رفتار صحیحی را از خود نشان نخواهد داد.

با پر کردن اعضای ساختار CLIENTCREATESTRUCT ، فراخوانی تابع CreateWindowEx با بکارگیری نام کلاس از پیش تعریف شده MDICLIENT و ارسال آدرس ساختار CLIENTCREATESTRUCT به lParam ، پنجره کلاینت ایجاد می‌شود. همچنین در پارامتر hWndParent ، باید دستگیره پنجره چهارچوب قرار گیرد تا رابطه‌ی پدر و فرزندی بین پنجره چهارچوب و پنجره کلاینت به ویندوز ارسال گردد. سبک‌های پنجره ای که باید استفاده شوند موارد زیر هستند:

WS_CHILD ,WS_VISIBLE and WS_CLIPCHILDREN

اگر حالت WS_VISIBLE فراموش شود، حتی در صورت ایجاد پنجره‌های فرزند MDI ، چیزی نمایش داده نخواهد شد.

مراحل ایجاد پنجره کلاینت به شرح زیر هستند:

۱. بدست آوردن دستگیره ساب منویی که نیاز است لیست پنجره‌های MDI به آن اضافه شوند.
۲. مقدار دهی اعضای ساختار CLIENTCREATESTRUCT ، با دستگیره منو فوق و ID اولین پنجره فرزند MDI .
۳. فراخوانی تابع CreateWindowEx با نام کلاس MDICLIENT ، با ارسال آدرس CLIENTCREATESTRUCT به lParam .

www.mohandesyar.com

ایجاد پنجره فرزند MDI :

تا اینجا هر دو پنجره چهارچوب و پنجره کلاینت ایجاد شده اند. برای ایجاد پنجره های فرزند MDI دو روش وجود دارد:

- ارسال پیغام WM_MDICREATE به پنجره کلاینت به همراه ارسال آدرس ساختار MDICREATESTRUCT در wParam .این روش، ساده ترین روش ایجاد پنجره های فرزند MDI بوده و متداول است.

```
.data?
  mdicreate MDICREATESTRUCT <>
  ....
.code
  .....
  [fill the members of mdicreate]
  ......
invoke SendMessage, hwndClient, WM_MDICREATE,addr mdicreate,0
```

در اینجا تابع SendMessage ، دستگیره پنجره تازه ایجاد شده را در صورت موفقیت بازخواهد گرداند. نیازی به ذخیره سازی این دستگیره نبوده و در زمان نیاز با روش‌های دیگری می‌توان آنرا بدست آورد. ساختار MDICREATESTRUCT به صورت زیر تعریف می‌شود:

```
MDICREATESTRUCT STRUCT
szClass   DWORD ?
szTitle    DWORD ?
hOwner    DWORD ?
x           DWORD ?
y           DWORD ?
lx          DWORD ?
ly          DWORD ?
style       DWORD ?
lParam    DWORD ?
MDICREATESTRUCT ENDS
```

szClass : آدرس کلاسی که بعنوان قالب برای پنجره های فرزند مورد استفاده قرار می گیرد.

www.mohandesyar.com

szTitle : آدرس متنی که مایل هستید بعنوان عنوان پنجره فرزند ظاهر شود.

hOwner : دستگیره وهله برنامه

x,y,lx,ly : مختصات بالا سمت چپ و طول و عرض پنجره فرزند

style : سبک پنجره فرزند. اگر پنجره کلاینت با سبک MDIS_ALLCHILDSTYLES ایجاد شده باشد، از هر نوع سبک پنجره‌ای می‌توان استفاده کرد.

lParam : مقدار ۳۲ بیتی که توسط برنامه تعریف می‌شود.این روشی است برای به اشتراک گذاشتن مقادیر بین پنجره های فرزند MDI . اگر نیازی به استفاده از آن نمی‌بینید ، آنرا با نال مقدار دهی کنید.

- همچنین می‌توان تابع CreateMDIWindow را نیز فراخوانی کرد. این تابع به صورت زیر تعریف می‌شود:

```
CreateMDIWindow proto lpClassName:DWORD
                      lpWindowName:DWORD
                      dwStyle:DWORD
                      x:DWORD
                      y:DWORD
                      nWidth:DWORD
                      nHeight:DWORD
                      hWndParent:DWORD
                      hInstance:DWORD
                      lParam:DWORD
```

اگر به پارامترهای این تابع دقت نمائید ، درخواهید یافت که آنها با اعضای ساختار MDICREATESTRUCT ، بجز hWndParent ، یکی هستند. ساختار MDICREATESTRUCT دارای فیلد hWndParent نیست، زیرا توسط SendMessage به پنجره کلاینت مربوطه ارسال می‌شود.

حال شاید بپرسید از کدام روش باید استفاده کرد و تفاوت این روشها در چیست؟

روش WM_MDICREATE ، تنها می تواند پنجره فرزندی را در همان ترد فراخوانی کننده آن ایجاد نماید. برای مثال اگر برنامه دو ترد داشته باشد و اولین ترد، پنجره چهارچوب را ایجاد کرده باشد، اگر دومین ترد قصد ایجاد پنجره فرزند را داشته باشد ، باید از تابع CreateMDIChild استفاده کند و ارسال پیغام WM_MDICREATE به اولین ترد کار نخواهد کرد.اگر برنامه single thread باشد از هر دو روش می‌توان استفاده کرد.

نیاز به توضیحات بیشتری در مورد رویه پنجره مربوط به پنجره‌های فرزند MDI وجود دارد. همانند حالت پنجره چهارچوب، نباید از تابع DefWindowProc جهت پردازش پیغامهای مدیریت نشده استفاده کرد و باید از تابع DefMDIChildProc استفاده نمود.پارامترهای این تابع با پارامترهای DefWindowProc یکسان هستند.

www.mohandesyar.com

همچنین بجز پیغام WM_MDICREATE ، پیغامهای دیگری نیز مربوط به پنجره های فرزند MDI هستند که در ادامه توضیح داده خواهند شد:

WM_MDIACTIVATE :این پیغام از طرف برنامه به پنجره کلاینت جهت فعال کردن پنجره فرزند انتخابی فرستاده می‌شود.زمانیکه پنجره کلاینت این پیغام را دریافت کند، پنجره فرزند انتخاب شده را فعال کرده و پیغام WM_MDIACTIVATE را به پنجره‌های در حال فعال و غیرفعال شدن می‌فرستد.این پیغام دو منظور را دنبال می کند. فعال کردن پنجره فرزند انتخابی و اعلان فعال ویا غیرفعال شدن پنجره فرزند، توسط آن پنجره. برای مثال در این حالت می‌توان منوهای پنجره چهارچوب را اصلاح کرد.

WM_MDIICONARRANGE , WM_MDITILE , WM_MDICASCADE : این پیغامها نحوه آرایش پنجره‌های فرزند را مدیریت می کنند.برای مثال با ارسال پیغام WM_MDICASCADE به پنجره کلاینت، پنجره‌های فرزند به صورت آبشاری مرتب می‌شوند.

WM_MDIDESTROY : این پیغام به پنجره کلاینت جهت تخریب یک پنجره فرزند ارسال می‌شود.برای تخریب یک پنجره فرزند نباید از DestroyWindow استفاده کرد. زیرا اگر پنجره فرزند در حالت حداکثر اندازه خود قرار داشته باشد، فراخوانی این تابع سبب می‌شود عنوان پنجره چهارچوب بازیابی نشود.

WM_MDIGETACTIVE : از این پیغام برای دریافت دستگیره پنجره فرزندی که فعال است استفاده می گردد.

WM_MDIRESTORE , WM_MDIMAXIMIZE :برای حداکثر کردن اندازه یک پنجره فرزند و یا به حالت اول درآوردن اندازه آن ، استفاده می‌شوند.همواره از این پیغامها برای تغییر اندازه پنجره‌های فرزند بجای فراخوانی ShowWindow به همراه SW_MAXIMIZE استفاده کنید. در غیراینصورت پیغامهای به حالت اول درآوردن پنجره مجددا کار نخواهند کرد.

WM_MDINEXT : این پیغام برای فعال کردن پنجره بعدی و یا قبلی وابسته به مقادیر wParam and lParam ارسال می‌شود.

WM_MDIREFRESHMENU : این پیغام برای به روز درآوردن منوهای پنجره چهارچوب ارسال می‌شود. این پیغام باید به همراه تابع DrawMenuBar بکار گرفته شود.

WM_MDISETMENU : از این پیغام برای تعویض کل و یا بخشی از منوی پنجره چهارچوب، به پنجره کلاینت ارسال می‌شود. در اینجا باید از این پیغام بجای SetMenu استفاده کرد. پس از آن باید از تابع DrawMenuBar جهت به روز درآوردن منو استفاده کرد. عموما از این پیغام زمانی استفاده می‌شود که پنجره فرزند MDI دارای منو بوده و در هنگام فعال شدن آن نیاز است تا منوی آن بجای منوی پنجره چهارچوب نمایش داده شود.

www.mohandesyar.com

به صورت خلاصه، روش ایجاد یک برنامه MDI به شرح زیر است:

۱. رجیستر کردن کلاس پنجره درهر دو حالت پنجره چهارچوب و پنجره فرزند.
۲. ایجاد پنجره چهارچوب توسط CreateWindowEx .
۳. با فراخوانی تابع TranslateMDISysAccel ، در حلقه پیغامها، فشردن شدن کلیدها قابل پردازش خواهند شد.
۴. درون رویه پنجره با فراخوانی DefFrameProc ، تمامی پیغامهای مدیریت نشده قابل پردازش خواهند شد.
۵. ایجاد پنجره کلاینت با فراخوانی تابع CreateWindowEx به همراه نام کلاس استاندارد MDICLIENT و ارسال آدرس CLIENTCREATESTRUCT در lParam . عموما این فراخوانی در هنگام پردازش پیغام WM_CREATE ، صورت می گیرد.
۶. ایجاد پنجره‌های فرزند MDI با ارسال پیغام WM_MDICREATE به پنجره کلاینت و یا با فراخوانی CreateMDIWindow .
۷. درون رویه پنجره فرزند MDI ، تمامی پیغامهای مدیریت نشده باید به DefMDIChildProc ارسال شوند.
۸. استفاده از نگارش MDI پیغامها در صورت وجود . برای مثال بجای فراخوانی تابع DestroyWindow ، باید از WM_MDIDESTROY استفاده شود.

کد مثال ۲۹ :

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

.const
IDR_MAINMENU    equ 101
IDR_CHILDMENU   equ 102
IDM_EXIT                equ 40001
IDM_TILEHORZ    equ 40002
IDM_TILEVERT    equ 40003
IDM_CASCADE     equ 40004
IDM_NEW                 equ 40005
IDM_CLOSE       equ 40006
```


www.mohandesyar.com

```
.data
ClassName           db "MDIASMClass",0
MDIClientName       db "MDICLIENT",0
MDIChildClassName db "Win32asmMDIChild",0
MDIChildTitle       db "MDI Child",0
AppName             db "Win32asm MDI Demo",0
ClosePromptMessage          db "Are you sure you want to close this window?",0

.data?
hInstance           dd ?
hMainMenu           dd ?
hwndClient          dd ?
hChildMenu          dd ?
mdicreate           MDICREATESTRUCT <>
hwndFrame           dd ?

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

WinMain proc hInst:HINSTANCE,hPrevInst:HINSTANCE,CmdLine:LPSTR,CmdShow:DWORD
        LOCAL wc:WNDCLASSEX
        LOCAL msg:MSG
        ;=================================================
        ; Register the frame window class
        ;=================================================
        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 hInstance
        pop wc.hInstance
        mov wc.hbrBackground,COLOR_APPWORKSPACE
        mov wc.lpszMenuName,IDR_MAINMENU
        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
        ;=================================================
        ; Register the MDI child window class
        ;=================================================
        mov wc.lpfnWndProc,offset ChildProc
        mov wc.hbrBackground,COLOR_WINDOW+1
        mov wc.lpszClassName,offset MDIChildClassName
        invoke RegisterClassEx,addr wc
        invoke CreateWindowEx,NULL,ADDR ClassName,ADDR AppName,\
                        WS_OVERLAPPEDWINDOW or WS_CLIPCHILDREN,CW_USEDEFAULT,\
                        CW_USEDEFAULT,CW_USEDEFAULT,CW_USEDEFAULT,NULL,0,\
                        hInst,NULL
        mov hwndFrame,eax
```


www.mohandesyar.com

```
	invoke LoadMenu,hInstance, IDR_CHILDMENU
	mov hChildMenu,eax
	invoke ShowWindow,hwndFrame,SW_SHOWNORMAL
	invoke UpdateWindow, hwndFrame
	.while TRUE
		invoke GetMessage,ADDR msg,NULL,0,0
		.break .if (!eax)
		invoke TranslateMDISysAccel,hwndClient,addr msg
		.if !eax
			invoke TranslateMessage, ADDR msg
			invoke DispatchMessage, ADDR msg
		.endif
	.endw
	invoke DestroyMenu, hChildMenu
	mov eax,msg.wParam
	ret
WinMain endp

WndProc proc hWnd:HWND, uMsg:UINT, wParam:WPARAM, lParam:LPARAM
	LOCAL ClientStruct:CLIENTCREATESTRUCT
	.if uMsg==WM_CREATE
		invoke GetMenu,hWnd
		mov hMainMenu,eax
		invoke GetSubMenu,hMainMenu,1
		mov ClientStruct.hWindowMenu,eax
		mov ClientStruct.idFirstChild,100
		INVOKE CreateWindowEx,NULL,ADDR MDIClientName,NULL,\
				WS_CHILD or WS_VISIBLE or
		 WS_CLIPCHILDREN,CW_USEDEFAULT,\
			CW_USEDEFAULT,CW_USEDEFAULT,CW_USEDEFAULT,hWnd,NULL,\
				hInstance,addr ClientStruct
		mov hwndClient,eax
		;=====================================
		; Initialize the MDICREATESTRUCT
		;=====================================
		mov mdicreate.szClass,offset MDIChildClassName
		mov mdicreate.szTitle,offset MDIChildTitle
		push hInstance
		pop mdicreate.hOwner
		mov mdicreate.x,CW_USEDEFAULT
		mov mdicreate.y,CW_USEDEFAULT
		mov mdicreate.lx,CW_USEDEFAULT
		mov mdicreate.ly,CW_USEDEFAULT
	.elseif uMsg==WM_COMMAND
		.if lParam==0
			mov eax,wParam
			.if ax==IDM_EXIT
				invoke SendMessage,hWnd,WM_CLOSE,0,0
			.elseif ax==IDM_TILEHORZ
	invoke SendMessage,hwndClient,WM_MDITILE,MDITILE_HORIZONTAL,0
			.elseif ax==IDM_TILEVERT
	invoke SendMessage,hwndClient,WM_MDITILE,MDITILE_VERTICAL,0
			.elseif ax==IDM_CASCADE
	invoke SendMessage,hwndClient,WM_MDICASCADE,MDITILE_SKIPDISABLED,0
			.elseif ax==IDM_NEW
	invoke SendMessage,hwndClient,WM_MDICREATE,0,addr mdicreate
			.elseif ax==IDM_CLOSE
	invoke SendMessage,hwndClient,WM_MDIGETACTIVE,0,0
```


www.mohandesyar.com

```
		invoke SendMessage,eax,WM_CLOSE,0,0
				.else
		invoke DefFrameProc,hWnd,hwndClient,uMsg,wParam,lParam
						ret
				.endif
			.endif
		.elseif uMsg==WM_DESTROY
			invoke PostQuitMessage,NULL
		.else
			invoke DefFrameProc,hWnd,hwndClient,uMsg,wParam,lParam
			ret
		.endif
		xor eax,eax
		ret
WndProc endp

ChildProc proc hChild:DWORD,uMsg:DWORD,wParam:DWORD,lParam:DWORD
		.if uMsg==WM_MDIACTIVATE
			mov eax,lParam
			.if eax==hChild
				invoke GetSubMenu,hChildMenu,1
				mov edx,eax
			invoke SendMessage,hwndClient,WM_MDISETMENU,hChildMenu,edx
			.else
				invoke GetSubMenu,hMainMenu,1
				mov edx,eax
				invoke SendMessage,hwndClient,WM_MDISETMENU,hMainMenu,edx
			.endif
			invoke DrawMenuBar,hwndFrame
		.elseif uMsg==WM_CLOSE
		invoke MessageBox,hChild,addr ClosePromptMessage,addr AppName,MB_YESNO
			.if eax==IDYES
			invoke SendMessage,hwndClient,WM_MDIDESTROY,hChild,0
			.endif
		.else
			invoke DefMDIChildProc,hChild,uMsg,wParam,lParam
			ret
		.endif
		xor eax,eax
		ret
ChildProc endp
end start
```

بررسی کد فوق:

اولین کاری که برنامه انجام می‌دهد رجیستر کردن کلاس پنجره چهارچوب و پنجره کلاینت است. سپس با فراخوانی CreateWindowEx ، پنجره چهارچوب ایجاد می‌شود.در ادامه هنگام پردازش پیغام WM_CREATE ، پنجره کلاینت ایجاد خواهد شد.

```
		LOCAL ClientStruct:CLIENTCREATESTRUCT
		.if uMsg==WM_CREATE
			invoke GetMenu,hWnd
			mov hMainMenu,eax
```


www.mohandesyar.com

```
invoke GetSubMenu,hMainMenu,1
mov ClientStruct.hWindowMenu,eax
mov ClientStruct.idFirstChild,100
invoke CreateWindowEx,NULL,ADDR MDIClientName,NULL,\
        WS_CHILD or WS_VISIBLE or WS_CLIPCHILDREN,CW_USEDEFAULT,\
        CW_USEDEFAULT,CW_USEDEFAULT,CW_USEDEFAULT,hWnd,NULL,\
        hInstance,addr ClientStruct
mov hwndClient,eax
```

با فراخوانی GetMenu ، دستگیره منو پنجره چهارچوب جهت استفاده در GetSubMenu ، دریافت می‌شود.لازم به ذکر است که در اینجا مقدار یک به GetSubMenu ارسال شده است زیرا می‌خواهیم ساب منوی window ، بعنون دومین ساب منو ظاهر شود.

سپس اعضای ساختار CLIENTCREATESTRUCT مقدار دهی می‌شوند.

در ادامه، اعضای ساختار MDICLIENTSTRUCT ، مقدار دهی خواهند شد. (انجام اینکار در هنگام پردازش پیغام WM_CREATE معمول است اما اجباری نیست.)

```
mov mdicreate.szClass,offset MDIChildClassName
mov mdicreate.szTitle,offset MDIChildTitle
push hInstance
pop mdicreate.hOwner
mov mdicreate.x,CW_USEDEFAULT
mov mdicreate.y,CW_USEDEFAULT
mov mdicreate.lx,CW_USEDEFAULT
mov mdicreate.ly,CW_USEDEFAULT
```

پس از ایجاد پنجره‌های چهارچوب و کلاینت، تابع LoadMenu را جهت بارگذاری منوی پنجره فرزند از ریسورس برنامه ، فراخوانی می‌کنیم. ما به این دستگیره جهت جایگزینی منوی پنجره چهارچوب با منوی پنجره فرزند ، نیاز داریم.

پیش از پایان کار برنامه، فراخوانی DestroyMenu را فراموش نکنید.هر چند ویندوز بطور معمول اینکار پاک سازی را انجام خواهد داد اما در این حالت بدلیل اینکه منوی ایجاد شده به هیچ پنجره‌ای وابسته نیست باید کار تخریب آنرا به صورت دستی انجام داد.

```
invoke LoadMenu,hInstance, IDR_CHILDMENU
mov hChildMenu,eax
.....
invoke DestroyMenu, hChildMenu
```

درون حلقه پیغامها، تابع TranslateMDISysAccel را فراخوانی خواهیم کرد.

```
.while TRUE
        invoke GetMessage,ADDR msg,NULL,0,0
        .break .if (!eax)
        invoke TranslateMDISysAccel,hwndClient,addr msg
        .if !eax
                invoke TranslateMessage, ADDR msg
                invoke DispatchMessage, ADDR msg
```


www.mohandesyar.com

```
            .endif
        .endw
```

اگر TranslateMDISysAccel صفر برنگرداند به این معنا است که ویندوز این پیغام را مدیریت کرده و نیازی نیست تا کار بیشتری بر روی آن انجام شود. اگر صفر برگرداند، یعنی این پیغام از نوع MDI نبوده و مطابق معمول پردازش می‌شود.

```
WndProc proc hWnd:HWND, uMsg:UINT, wParam:WPARAM, lParam:LPARAM
        .....
        .else
                invoke DefFrameProc,hWnd,hwndClient,uMsg,wParam,lParam
                ret
        .endif
        xor eax,eax
        ret
WndProc endp
```

لازم به ذکر است که درون رویه پنجره مربوط به پنجره چهارچوب، با فراخوانی DefFrameProc ، پیغامهایی را که مایل به پردازش آنها نیستیم، مدیریت می‌نمائیم.

قسمت عمده رویه پنجره را پردازش WM_COMMAND به خود اختصاص می‌دهد. هنگامیکه کاربر از منوی پنجره اصلی برنامه گزینه New را انتخاب می‌کند، پنجره فرزند MDI جدید را ایجاد خواهیم کرد.

```
        .elseif ax==IDM_NEW
                invoke SendMessage,hwndClient,WM_MDICREATE,0,addr mdicreate
```

با ارسال پیغام WM_MDICREATE به پنجره کلاینت به همراه آدرس ساختار MDICREATESTRUCT در lParam ، پنجره‌های فرزند MDI ایجاد می‌شوند.

```
ChildProc proc hChild:DWORD,uMsg:DWORD,wParam:DWORD,lParam:DWORD
        .if uMsg==WM_MDIACTIVATE
                mov eax,lParam
                .if eax==hChild
                        invoke GetSubMenu,hChildMenu,1
                        mov edx,eax
                        invoke SendMessage,hwndClient,WM_MDISETMENU,hChildMenu,edx
                .else
                        invoke GetSubMenu,hMainMenu,1
                        mov edx,eax
                        invoke SendMessage,hwndClient,WM_MDISETMENU,hMainMenu,edx
                .endif
                invoke DrawMenuBar,hwndFrame
```

هنگامیکه یک پنجره فرزند MDI ایجاد می‌شود، فعال یا غیرفعال بودن خود را توسط پیغام WM_MDIACTIVATE بررسی خواهد کرد.اینکار با مقایسه مقدار lParam که حاوی دستگیره پنجره فرزند فعال است با دستگیره پنجره، انجام می‌شود. اگر شرط صادق باشد، پنجره فعال بوده و منوی آن جایگزین منو پنجره چهارچوب خواهد شد.

www.mohandesyar.com

از آنجائیکه منوی اصلی جایگزین خواهد شد، نیاز است تا مجددا به ویندوز در مورد دستگیره ساب منو window گزارش داده شود و این کار را با فراخوانی GetSubMenu جهت دریافت دستگیره ساب منو انجام خواهیم داد. سپس جهت حصول نتیجه مورد نظر، پیغام WM_MDISETMENU را به پنجره کلاینت ارسال خواهیم کرد. مقدار wParam مربوط به WM_MDISETMENU ، حاوی دستگیره منویی است که جایگزین منوی اصلی خواهد شد. lParam حاوی دستگیره ساب منویی است که جهت نمایش لیست پنجره‌ها از آن استفاده خواهد شد.

بلافاصله پس از فراخوانی WM_MDISETMENU ، DrawMenuBar را برای به روز رسانی منو، فراخوانی می‌نمائیم. درغیراینصورت چیزی نمایش داده نخواهد شد.

```
.else
        invoke DefMDIChildProc,hChild,uMsg,wParam,lParam
        ret
.endif
```

درون رویه پنجره فرزند MDI ، باید تمامی پیغامهای مدیریت نشده را به DefMDIChildProc فرستاد.

```
.elseif ax==IDM_TILEHORZ
        invoke SendMessage,hwndClient,WM_MDITILE,MDITILE_HORIZONTAL,0
.elseif ax==IDM_TILEVERT
        invoke SendMessage,hwndClient,WM_MDITILE,MDITILE_VERTICAL,0
.elseif ax==IDM_CASCADE
        invoke SendMessage,hwndClient,WM_MDICASCADE,MDITILE_SKIPDISABLED,0
```

هنگامیکه کاربر یکی از آیتم‌های ساب منوی window را انتخاب کند، پیغام متناظر آن به پنجره کلاینت ارسال می‌شود.در مورد پیغامهای WM_CASCADE و امثال آن پیشتر توضیح داده شد.

```
.elseif ax==IDM_CLOSE
        invoke SendMessage,hwndClient,WM_MDIGETACTIVE,0,0
        invoke SendMessage,eax,WM_CLOSE,0,0
```

اگر کاربر گزینه close را انتخاب کند، باید دستگیره پنجره‌ای را که هم اکنون فعال است بدست آورد. اینکار با ارسال پیغام WM_MDIGETACTIVE به پنجره کلاینت قابل انجام است. خروجی قرار گرفته در eax ، دستگیره پنجره MDI فعال است.سپس پیغام WM_CLOSE به آن پنجره ارسال خواهد شد.

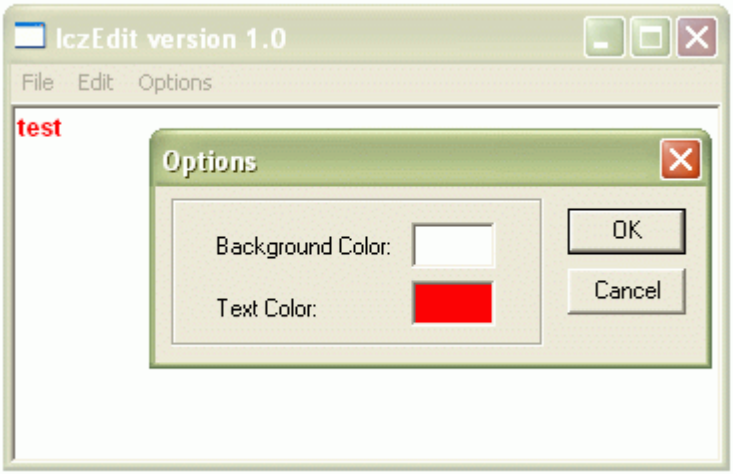
```
.elseif uMsg==WM_CLOSE
        invoke MessageBox,hChild,addr ClosePromptMessage,addr AppName,MB_YESNO
        .if eax==IDYES
                invoke SendMessage,hwndClient,WM_MDIDESTROY,hChild,0
        .endif
```


www.mohandesyar.com

درون رویه پنجره فرزند MDI ، هنگامیکه پیغام WM_CLOSE دریافت شود، به کاربر پیغامی جهت تایید و یا لغو بسته شدن پنجره نمایش داده خواهد شد. اگر پاسخ مثبت بود، پیغام WM_MDIDESTROY به پنجره کلاینت ارسال نخواهد شد. این پیغام سبب بسته شدن پنجره فرزند و بازیابی عنوان پنجره چهارچوب می گردد.

www.mohandesyar.com

مثال ۳۰- اصول مقدماتی کار با RichEdit Control



کنترل richedit دارای ویژگی‌های متعددی نسبت به یک کنترل تکست باکس ساده ، مانند امکان استفاده از چندین نوع قلم، جستجو جهت یافتن متن ، توانایی استفاده از OLE-embedded objects و غیره است. از آنجائیکه ویژگی های این کنترل نسبتا زیاد است، آنرا درون dll مجزایی قرار داده اند. بنابراین برای استفاده از آن نمی توان از **InitCommonControls** کمک گرفت و باید از **LoadLibrary** برای بارگذاری آن استفاده کرد. تا به امروز حداقل سه نگارش از آن به همراه نگارش‌های مختلف ویندوز ارائه شده است که لیست آنها در ذیل ارائه می‌شود:

DLL Name	RichEdit version	Richedit Class Name
Riched32.dll	1.0	RICHEDIT
RichEd20.dll	2.0	RICHEDIT20A
RichEd20.dll	3.0	RICHEDIT20A

همانطور که ملاحظه می نمائید دو نگارش ۲ و ۳ آن دارای نام کلاس و نام فایل یکسانی هستند.

```
.data
   RichEditDLL db "RichEd20.dll",0
.....
.data?
hRichEditDLL dd ?
.code
invoke LoadLibrary,addr RichEditDLL
mov hRichEditDLL,eax
......
invoke FreeLibrary,hRichEditDLL
```


www.mohandesyar.com

هنگامیکه این dll بارگذاری شود ، کلاس پنجره RichEdit را رجیستر خواهد کرد.

تفاوت نگارش های مختلف این dll ها به شرح زیر است:

Feature	Version 1.0	Version 2.0	Version 3.0
selection bar	x	x	x
unicode editing		x	x
character/paragraph formatting	x	x	x
search for text	forward	forward/backward	forward/backward
OLE embedding	x	x	x
Drag and drop editing	x	x	x
Undo/Redo	single-level	multi-level	multi-level
automatic URL recognition		x	x
Accelerator key support		x	x
Windowless operation		x	x
Line break	CRLF	CR only	CR only (can emulate version 1.0)
Zoom			x
Paragraph numbering			x
simple table			x
normal and heading styles			x
underline coloring			x
hidden text			x
font binding			x

جدول فوق به هیچ عنوان کامل نبوده و صرفا موارد مهم متذکر شده‌اند.

ایجاد richedit control :

پس از بارگذاری dll ، از تابع CreateWindowEx جهت ایجاد کنترل استفاده می‌شود. از تمام حالتها در اینجا می توان بهره جست بجز موارد زیر:

ES_LOWERCASE, ES_UPPERCASE and ES_OEMCONVERT

www.mohandesyar.com

نحوه کار :

```
.const
        RichEditID equ 300
.data
        RichEditDLL db "RichEd20.dll",0
        RichEditClass db "RichEdit20A",0
        ...
.data?
        hRichEditDLL dd ?
        hwndRichEdit dd ?
.code
        .....
        invoke LoadLibrary,addr RichEditDLL
        mov hRichEditDLL,eax
        invoke CreateWindowEx,0,addr RichEditClass,\
        WS_VISIBLE or ES_MULTILINE or WS_CHILD or WS_VSCROLL or WS_HSCROLL, \
        CW_USEDEFAULT,CW_USEDEFAULT,CW_USEDEFAULT,CW_USEDEFAULT,hWnd,RichEditID,hInsta
nce,0
mov hwndRichEdit,eax
```

تنظیم رنگ متن و زمینه :

برای تنظیم رنگ زمینه می‌توان از پیغام EM_SETBKGNDCOLOR که به صورت زیر تعریف می‌شود استفاده کرد.

wParam : گزینه رنگ. اگر مقدار آن مساوی با صفر باشد یعنی ویندوز از مقدار **lParam** بعنوان رنگ زمینه استفاده خواهد کرد. اگر این مقدار غیرصفر باشد، ویندوز از رنگ پیش فرض سیستمی برای این منظور کمک خواهد گرفت. بنابراین جهت تغییر رنگ باید از مقداری مساوی صفر استفاده نمود.

lParam : در صورتیکه wParam مساوی صفر باشد، بیانگر ساختار COLORREF است. برای مثال اگر نیاز است رنگ زمینه آبی خالص باشد می‌توان از کد زیر استفاده کرد:

```
invoke   SendMessage,hwndRichEdit,EM_SETBKGNDCOLOR,0,0FF0000h
```

برای تنظیم رنگ متن از پیغام EM_SETCHARFORMAT استفاده می‌شود.این پیغام بر روی متن انتخابی و یا کل متن قرار گرفته در کنترل اعمال می‌شود. این پیغام به صورت زیر تعریف می‌شود:

wParam : گزینه های تعیین قالب:

SCF_ALL : بر روی کل متن قرار گرفته در کنترل مؤثر خواهد بود.

SCF_SELECTION : تنها بر روی متن انتخاب شده اثر می گذارد.

SCF_WORD or SCF_SELECTION : بر روی یک حرف انتخاب شده تاثیر می گذارد.این دو پرچم باید با هم بکار گرفته شوند.

www.mohandesyar.com

lParam : اشاره‌گری است به ساختارهای CHARFORMAT or CHARFORMAT2 . تنها در نگارش‌های ۲ و یا بالاتر ساختار CHARFORMAT2 مهیا است. این ساختار به صورت زیر تعریف می‌شود:

```
CHARFORMATA STRUCT
        cbSize DWORD ?
        dwMask DWORD ?
        dwEffects DWORD    ?
        yHeight DWORD ?
        yOffset DWORD ?
        crTextColor COLORREF ?
        bCharSet BYTE ?
        bPitchAndFamily    BYTE ?
        szFaceName BYTE LF_FACESIZE dup(?)
        _wPad2 WORD ?
CHARFORMATA ENDS
```

Field Name	Description
cbSize	اندازه ساختار است. از این ساختار در جهت مشخص کردن نگارش کنترل مورد استفاده و ساختارهای CHARFORMAT2 or CHARFORMAT استفاده می‌شود.
dwMask	پرچم های بیتی بیانگر اعضای معتبر ساختار. مقادیر مجاز برای آن :

CFM_BOLD	مقدار CFE_BOLD عضو dwEffects معتبر خواهد بود.
CFM_CHARSET	عضو bCharSet معتبر است.
CFM_COLOR	عضو crTextColor و مقدار CFE_AUTOCOLOR عضو dwEffects معتبر هستند.
CFM_FACE	عضو szFaceName معتبر است.
CFM_ITALIC	مقدار CFE_ITALIC عضو dwEffects معتبر است.
CFM_OFFSET	عضو yOffset معتبر است.
CFM_PROTECTED	مقدار CFE_PROTECTED عضو dwEffects معتبر است.
CFM_SIZE	عضو yHeight معتبر است.
CFM_STRIKEOUT	مقدار CFE_STRIKEOUT عضو dwEffects معتبر است.
CFM_UNDERLINE	مقدار CFE_UNDERLINE عضو dwEffects معتبر است.

www.mohandesyar.com

dwEffects	پرچم هایی که جلوه های نمایشی متن را مشخص می کنند:	
	CFE_AUTOCOLOR	نمایانگر رنگ متن سیستمی است.
	CFE_BOLD	
	CFE_ITALIC	
	CFE_STRIKEOUT	
	CFE_UNDERLINE	
	CFE_PROTECTED	کاراکترهای محافظت شده. در این حالت هرگونه سعی در جهت تغییر آنها پیغام EN_PROTECTED را به
yHeight	ارتفاع کاراکترها بر حسب twips . 1/1440 of an inch or 1/20 of a printer's point	
yOffset	آفست یک کاراکتر از خط مبنا.	
crTextColor	رنگ متن. اگر از پرچم CFE_AUTOCOLOR استفاده شود از این عضو صرفنظر خواهد شد.	
bCharSet	Character set value	
bPitchAndFamily	Font family and pitch	
szFaceName	آرایه حروف مختوم به نال مشخص کننده نام قلم.	
_wPad2	Padding	

ساختار **CHARFORMAT2** ، جلوه های متنی بیشتری مانند font weight, spacing,text background color, kerning و غیره را دربردارد. اگر نیازی به این جلوه‌ها ندارید به سادگی از **CHARFORMAT** استفاده کنید.

جهت اعمال قالب ویژه به متن باید بازه دلخواهی از حروف را انتخاب کرد. این کنترل به هر حرف یک آی دی انتساب می دهد. اولین حرف موجود در کنترل با آی دی صفر مشخص شده و دومین حرف با یک و الی آخر. برای مشخص نمودن بازه حروف مورد نظر باید آی دی اولین و آخرین حرف آن را مشخص نمود. جهت اعمال قالب بر روی متن به کمک **EM_SETCHARFORMAT** ، حداقل سه راه وجود دارد:

۱. اعمال به تمامی حروف موجود در کنترل (**SCF_ALL**).
۲. اعمال به حروف انتخابی (**SCF_SELECTION**).
۳. اعمال به کل کلمه در حالتی که یک یا چند حرف آن انتخاب شده باشد (**SCF_WORD** or **SCF_SELECTION**).

www.mohandesyar.com

برای استفاده از **EM_SETCHARFORMAT** ، باید اعضای **CHARFORMAT** و یا **CHARFORMAT2** مقدار دهی شوند.برای مثال (تنظیم رنگ متن) :

```
.data?
        cf CHARFORMAT <>
....
.code
        mov cf.cbSize,sizeof cf
        mov cf.dwMask,CFM_COLOR
        mov cf.crTextColor,0FF0000h
invoke SendMessage,hwndRichEdit,EM_SETCHARFORMAT,SCF_ALL,addr cf
```

لازم به ذکر است که اگر متنی درون کنترل وجود نداشته باشد، اجرای دستورات فوق سبب خواهد شد تا متنی که در کنترل قرار خواهد گرفت این قالب را داشته باشد.

ذخیره سازی و دریافت متن از کنترل

پیشتر در مورد پیغامهای **WM_GETTEXT/WM_SETTEXT** جهت دریافت و ذخیره سازی متن در کنترل ها بحث شد. این پیغامها هنوز درباره این کنترل نیز صادق هستند اما برای حجم‌های بالای متن کارآیی لازم را ندارند. تکست باکس معمولی توانایی دریافت 64K متن را دارا است اما کنترل richedit می‌تواند حجم بالاتری را نیز پردازش کند. در اینجا برای پردازش حجم متنی مانند ۱۰ مگابایت و یا بالاتر از روشی به نام text streaming استفاده شده است. آدرس تابعی callback به کنترل ارسال شده ، کنترل آنرا فراخوانی کرده و آدرس بافر را به آن اعلام می‌کند.تابع callback ، بافر را با داده های مورد نظر پر کرده و آنرا به کنترل ارسال می‌نماید. سپس منتظر فراخوانی بعدی خواهد ماند تا عملیات تکمیل گردد. این روش هم برای خواندن اطلاعات از کنترل و همچنین جهت مقدار دهی آن مورد استفاده قرار می‌گیرد (**EM_STREAMIN** and **EM_STREAMOUT**).

هر دوی **EM_STREAMIN** and **EM_STREAMOUT** از نحو یکسانی برخوردار هستند:

wParam : گزینه‌های تعیین قالب

SF_RTF : قالب داده ها (RTF) rich-text format است.

SF_TEXT : قالب داده ها متن ساده است.

SFF_PLAINRTF : تنها واژه های کلیدی مشترک در بین تمامی زبانها دریافت خواهند شد.

SFF_SELECTION : هدف نهایی عملیات را متن انتخاب شده تعیین می‌کند. اگر عملیات دریافت داده‌ها باشد، متن دریافتی با متن انتخابی جایگزین خواهد شد. اگر

www.mohandesyar.com

عملیات ذخیره سازی داده‌ها باشد، متن انتخابی ذخیره خواهد شد. اگر این پرچم انتخاب نشود، عملیات بر تمامی متن موجود در کنترل اثرگذار خواهد شد.

lParam: اشاره‌گری است به ساختار **EDITSTREAM** که به صورت زیر تعریف می‌شود:

```
EDITSTREAM STRUCT
        dwCookie DWORD    ?
        dwError DWORD ?
        pfnCallback DWORD ?
EDITSTREAM ENDS
```

dwCookie: مقداری است تعریف شده توسط برنامه که به تابع callback تعیین شده در عضو pfnCallback ارسال می‌شود.برای مثال دستگیره فایل جهت پردازش ارسال خواهد شد.

dwError: نتیجه عملیات ورودی و یا خروجی را مشخص می‌کند و صفر به معنای عدم وجود خطا است. مقدار غیرصفر، می‌تواند مقدار خروجی تابع EditStreamCallback باشد ویا بیانگر کدی است که مشخص کننده نوع خطا است.

pfnCallback: اشاره‌گری است به تابع EditStreamCallback ، که تابعی است تعریف شده توسط برنامه و انتقال داده‌ها را کنترل می‌کند. کنترل، این تابع callback را متناوبا فراخوانی کرده و هربار قسمتی از داده‌ها را انتقال می‌دهد.این تابع به صورت زیر تعریف می‌شود:

```
EditStreamCallback    proto     dwCookie:DWORD,
                                pBuffer:DWORD,
                                NumBytes:DWORD,
                                pBytesTransferred:DWORD
```

باید در برنامه تابعی را به صورت فوق تعریف نمود و سپس آدرس آنرا به **EM_STREAMIN** or **EM_STREAMOUT** توسط **EDITSTREAM** ارسال کرد.

جهت عملیات stream-in:

dwCookie: متغیر تعریف شده توسط برنامه که به همراه **EM_STREAMIN** توسط ساختار **EDITSTREAM** ارسال می‌شود.تقریبا همیشه دستگیره فایل مورد نظر در اینجا ارسال می‌گردد.

pBuffer: اشاره‌گری است به بافری تهیه شده توسط کنترل، که متنی را از تابع callback دریافت می‌کند.

NumBytes: حداکثر تعداد بایتی است که می‌توان در بافر (pBuffer) در این فراخوانی نوشت.

pBytesTransferred: عموما این مقدار انتقال داده شد با مقدار **NumBytes** یکی است. استثناء تنها زمانی است که داده های ارسالی از حجم بافر در نظر گرفته شده کمتر باشند.

www.mohandesyar.com

جهت عملیات stream-out :

dwCookie : همانند حالت stream-in است.

pBuffer : اشاره گری است به بافری که توسط کنترل مهیا شده و مقدار دهی می گردد.برای بدست آوردن مقدار آن باید مقدار **NumBytes** بررسی شود.

NumBytes : اندازه داده های قرار گرفته در بافری که توسط pBuffer به آن اشاره می‌شود.

pBytesTransferred : اشاره گری است به مقداری که بیانگر تعداد بایتهای دریافتی از بافر است.

اگر تابع callback صفر برگرداند به معنای موفقیت عملیات بوده و کنترل، فراخوانی تابع callback را ادامه خواهد داد و اینکار تا زمانی که داده وجود داشته باشد ادامه خواهد یافت. موفقیت و یا شکست عملیات در عضو **dwError** ساختار **EDITSTREAM** قرار می گیرد و این مورد را با SendMessage می‌توان بررسی کرد.

کد مثال ۳۰ :

```
.386
.model flat,stdcall
option casemap:none
include \masm32\include\windows.inc
include \masm32\include\user32.inc
include \masm32\include\comdlg32.inc
include \masm32\include\gdi32.inc
include \masm32\include\kernel32.inc
includelib \masm32\lib\gdi32.lib
includelib \masm32\lib\comdlg32.lib
includelib \masm32\lib\user32.lib
includelib \masm32\lib\kernel32.lib

WinMain proto :DWORD,:DWORD,:DWORD,:DWORD

.const
IDR_MAINMENU            equ 101
IDM_OPEN                equ  40001
IDM_SAVE                equ 40002
IDM_CLOSE               equ 40003
IDM_SAVEAS              equ 40004
IDM_EXIT                equ 40005
IDM_COPY                equ  40006
IDM_CUT                 equ  40007
IDM_PASTE               equ 40008
IDM_DELETE              equ 40009
IDM_SELECTALL           equ 40010
IDM_OPTION                      equ 40011
IDM_UNDO                        equ 40012
IDM_REDO                        equ 40013
IDD_OPTIONDLG           equ 101
```


www.mohandesyar.com

```
IDC_BACKCOLORBOX            equ 1000
IDC_TEXTCOLORBOX            equ 1001

RichEditID                          equ 300

.data
ClassName db "IczEditClass",0
AppName  db "IczEdit version 1.0",0
RichEditDLL db "riched20.dll",0
RichEditClass db "RichEdit20A",0
NoRichEdit db "Cannot find riched20.dll",0
ASMFilterString                 db "ASM Source code (*.asm)",0,"*.asm",0
                                    db "All Files (*.*)",0,"*.*",0,0
OpenFileFail db "Cannot open the file",0
WannaSave db "The data in the control is modified. Want to save it?",0
FileOpened dd FALSE
BackgroundColor dd 0FFFFFFh             ; default to white
TextColor dd 0                  ; default to black

.data?
hInstance dd ?
hRichEdit dd ?
hwndRichEdit dd ?
FileName db 256 dup(?)
AlternateFileName db 256 dup(?)
CustomColors dd 16 dup(?)

.code
start:
        invoke GetModuleHandle, NULL
        mov    hInstance,eax
        invoke LoadLibrary,addr RichEditDLL
        .if eax!=0
                mov hRichEdit,eax
                invoke WinMain, hInstance,0,0, SW_SHOWDEFAULT
                invoke FreeLibrary,hRichEdit
        .else
                invoke MessageBox,0,addr NoRichEdit,addr AppName,MB_OK or MB_ICONERROR
        .endif
        invoke ExitProcess,eax

WinMain proc hInst:DWORD,hPrevInst:DWORD,CmdLine:DWORD,CmdShow:DWORD
        LOCAL wc:WNDCLASSEX
        LOCAL msg:MSG
        LOCAL hwnd:DWORD
        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,IDR_MAINMENU
        mov   wc.lpszClassName,OFFSET ClassName
        invoke LoadIcon,NULL,IDI_APPLICATION
        mov   wc.hIcon,eax
        mov   wc.hIconSm,eax
```


www.mohandesyar.com

```
        invoke LoadCursor,NULL,IDC_ARROW
        mov   wc.hCursor,eax
        invoke RegisterClassEx, addr wc
        INVOKE CreateWindowEx,NULL,ADDR ClassName,ADDR AppName,\
      WS_OVERLAPPEDWINDOW,CW_USEDEFAULT,\
      CW_USEDEFAULT,CW_USEDEFAULT,CW_USEDEFAULT,NULL,NULL,\
      hInst,NULL
        mov   hwnd,eax
        invoke ShowWindow, hwnd,SW_SHOWNORMAL
        invoke UpdateWindow, hwnd
        .while TRUE
                invoke GetMessage, ADDR msg,0,0,0
                .break .if (!eax)
                invoke TranslateMessage, ADDR msg
                invoke DispatchMessage, ADDR msg
        .endw
        mov   eax,msg.wParam
        ret
WinMain endp

StreamInProc proc hFile:DWORD,pBuffer:DWORD, NumBytes:DWORD, pBytesRead:DWORD
        invoke ReadFile,hFile,pBuffer,NumBytes,pBytesRead,0
        xor eax,1
        ret
StreamInProc endp

StreamOutProc proc hFile:DWORD,pBuffer:DWORD, NumBytes:DWORD, pBytesWritten:DWORD
        invoke WriteFile,hFile,pBuffer,NumBytes,pBytesWritten,0
        xor eax,1
        ret
StreamOutProc endp

CheckModifyState proc hWnd:DWORD
        invoke SendMessage,hwndRichEdit,EM_GETMODIFY,0,0
        .if eax!=0
                invoke MessageBox,hWnd,addr WannaSave,addr AppName,MB_YESNOCANCEL
                .if eax==IDYES
                        invoke SendMessage,hWnd,WM_COMMAND,IDM_SAVE,0
                .elseif eax==IDCANCEL
                        mov eax,FALSE
                        ret
                .endif
        .endif
        mov eax,TRUE
        ret
CheckModifyState endp

SetColor proc
        LOCAL cfm:CHARFORMAT
        invoke SendMessage,hwndRichEdit,EM_SETBKGNDCOLOR,0,BackgroundColor
        invoke RtlZeroMemory,addr cfm,sizeof cfm
        mov cfm.cbSize,sizeof cfm
        mov cfm.dwMask,CFM_COLOR
        push TextColor
        pop cfm.crTextColor
        invoke SendMessage,hwndRichEdit,EM_SETCHARFORMAT,SCF_ALL,addr cfm
        ret
SetColor endp
```


www.mohandesyar.com

```
OptionProc proc hWnd:DWORD, uMsg:DWORD, wParam:DWORD, lParam:DWORD
	LOCAL clr:CHOOSECOLOR
	.if uMsg==WM_INITDIALOG
	.elseif uMsg==WM_COMMAND
		mov eax,wParam
		shr eax,16
		.if ax==BN_CLICKED
			mov eax,wParam
			.if ax==IDCANCEL
				invoke SendMessage,hWnd,WM_CLOSE,0,0
			.elseif ax==IDC_BACKCOLORBOX
				invoke RtlZeroMemory,addr clr,sizeof clr
				mov clr.lStructSize,sizeof clr
				push hWnd
				pop clr.hwndOwner
				push hInstance
				pop clr.hInstance
				push BackgroundColor
				pop clr.rgbResult
				mov clr.lpCustColors,offset CustomColors
				mov clr.Flags,CC_ANYCOLOR or CC_RGBINIT
				invoke ChooseColor,addr clr
				.if eax!=0
					push clr.rgbResult
					pop BackgroundColor
					invoke GetDlgItem,hWnd,IDC_BACKCOLORBOX
					invoke InvalidateRect,eax,0,TRUE
				.endif
			.elseif ax==IDC_TEXTCOLORBOX
				invoke RtlZeroMemory,addr clr,sizeof clr
				mov clr.lStructSize,sizeof clr
				push hWnd
				pop clr.hwndOwner
				push hInstance
				pop clr.hInstance
				push TextColor
				pop clr.rgbResult
				mov clr.lpCustColors,offset CustomColors
				mov clr.Flags,CC_ANYCOLOR or CC_RGBINIT
				invoke ChooseColor,addr clr
				.if eax!=0
					push clr.rgbResult
					pop TextColor
					invoke GetDlgItem,hWnd,IDC_TEXTCOLORBOX
					invoke InvalidateRect,eax,0,TRUE
				.endif
			.elseif ax==IDOK

	;=====================================================
	; Save the modify state of the richedit control because changing the text color changes the
	; modify state of the richedit control.

	;=====================================================
				invoke SendMessage,hwndRichEdit,EM_GETMODIFY,0,0
				push eax
				invoke SetColor
				pop eax
```


www.mohandesyar.com

```
                                        invoke SendMessage,hwndRichEdit,EM_SETMODIFY,eax,0
                                        invoke EndDialog,hWnd,0
                                .endif
                        .endif
                .elseif uMsg==WM_CTLCOLORSTATIC
                        invoke GetDlgItem,hWnd,IDC_BACKCOLORBOX
                        .if eax==lParam
                                invoke CreateSolidBrush,BackgroundColor
                                ret
                        .else
                                invoke GetDlgItem,hWnd,IDC_TEXTCOLORBOX
                                .if eax==lParam
                                        invoke CreateSolidBrush,TextColor
                                        ret
                                .endif
                        .endif
                        mov eax,FALSE
                        ret
                .elseif uMsg==WM_CLOSE
                        invoke EndDialog,hWnd,0
                .else
                        mov eax,FALSE
                        ret
                .endif
                mov eax,TRUE
                ret
OptionProc endp

WndProc proc hWnd:DWORD, uMsg:DWORD, wParam:DWORD, lParam:DWORD
        LOCAL chrg:CHARRANGE
        LOCAL ofn:OPENFILENAME
        LOCAL buffer[256]:BYTE
        LOCAL editstream:EDITSTREAM
        LOCAL hFile:DWORD
        .if uMsg==WM_CREATE
                invoke CreateWindowEx,WS_EX_CLIENTEDGE,addr RichEditClass,0,WS_CHILD or
WS_VISIBLE or ES_MULTILINE or WS_VSCROLL or WS_HSCROLL or ES_NOHIDESEL,\
                                        CW_USEDEFAULT,CW_USEDEFAULT, CW_USEDEFAULT,
CW_USEDEFAULT,hWnd,RichEditID,hInstance,0
                mov hwndRichEdit,eax

        ;=====================================================
                ; Set the text limit. The default is 64K
        ;=====================================================
                invoke SendMessage,hwndRichEdit,EM_LIMITTEXT,-1,0

        ;=====================================================
                ; Set the default text/background color

        ;=====================================================
                invoke SetColor
                invoke SendMessage,hwndRichEdit,EM_SETMODIFY,FALSE,0
                invoke SendMessage,hwndRichEdit,EM_EMPTYUNDOBUFFER,0,0
        .elseif uMsg==WM_INITMENUPOPUP
                mov eax,lParam
                .if ax==0               ; file menu
                        .if FileOpened==TRUE            ; a file is already opened
                                invoke EnableMenuItem,wParam,IDM_OPEN,MF_GRAYED
```


www.mohandesyar.com

```
                        invoke EnableMenuItem,wParam,IDM_CLOSE,MF_ENABLED
                        invoke EnableMenuItem,wParam,IDM_SAVE,MF_ENABLED
                        invoke EnableMenuItem,wParam,IDM_SAVEAS,MF_ENABLED
                .else
                        invoke EnableMenuItem,wParam,IDM_OPEN,MF_ENABLED
                        invoke EnableMenuItem,wParam,IDM_CLOSE,MF_GRAYED
                        invoke EnableMenuItem,wParam,IDM_SAVE,MF_GRAYED
                        invoke EnableMenuItem,wParam,IDM_SAVEAS,MF_GRAYED
                .endif
        .elseif ax==1           ; edit menu

;=========================================================
; Check whether there is some text in the clipboard. If so, we enable the paste menuitem

;=========================================================
                invoke SendMessage,hwndRichEdit,EM_CANPASTE,CF_TEXT,0
                .if eax==0                      ; no text in the clipboard
                        invoke EnableMenuItem,wParam,IDM_PASTE,MF_GRAYED
                .else
                        invoke EnableMenuItem,wParam,IDM_PASTE,MF_ENABLED
                .endif

;=========================================================
                ; check whether the undo queue is empty

;=========================================================
                invoke SendMessage,hwndRichEdit,EM_CANUNDO,0,0
                .if eax==0
                        invoke EnableMenuItem,wParam,IDM_UNDO,MF_GRAYED
                .else
                        invoke EnableMenuItem,wParam,IDM_UNDO,MF_ENABLED
                .endif

;=========================================================
                ; check whether the redo queue is empty

;=========================================================
                invoke SendMessage,hwndRichEdit,EM_CANREDO,0,0
                .if eax==0
                        invoke EnableMenuItem,wParam,IDM_REDO,MF_GRAYED
                .else
                        invoke EnableMenuItem,wParam,IDM_REDO,MF_ENABLED
                .endif

;=========================================================
                ; check whether there is a current selection in the richedit control.
                ; If there is, we enable the cut/copy/delete menuitem

;=========================================================
                invoke SendMessage,hwndRichEdit,EM_EXGETSEL,0,addr chrg
                mov eax,chrg.cpMin
                .if eax==chrg.cpMax                     ; no current selection
                        invoke EnableMenuItem,wParam,IDM_COPY,MF_GRAYED
                        invoke EnableMenuItem,wParam,IDM_CUT,MF_GRAYED
                        invoke EnableMenuItem,wParam,IDM_DELETE,MF_GRAYED
                .else
                        invoke EnableMenuItem,wParam,IDM_COPY,MF_ENABLED
                        invoke EnableMenuItem,wParam,IDM_CUT,MF_ENABLED
```


www.mohandesyar.com

```
                                        invoke EnableMenuItem,wParam,IDM_DELETE,MF_ENABLED
                                .endif
                        .endif
                .elseif uMsg==WM_COMMAND
                        .if lParam==0                   ; menu commands
                                mov eax,wParam
                                .if ax==IDM_OPEN
                                        invoke RtlZeroMemory,addr ofn,sizeof ofn
                                        mov ofn.lStructSize,sizeof ofn
                                        push hWnd
                                        pop ofn.hwndOwner
                                        push hInstance
                                        pop ofn.hInstance
                                        mov ofn.lpstrFilter,offset ASMFilterString
                                        mov ofn.lpstrFile,offset FileName
                                        mov byte ptr [FileName],0
                                        mov ofn.nMaxFile,sizeof FileName
                                        mov ofn.Flags,\
                                    OFN_FILEMUSTEXIST or OFN_HIDEREADONLY or OFN_PATHMUSTEXIST
                                        invoke GetOpenFileName,addr ofn
                                        .if eax!=0
                                                invoke CreateFile,addr
FileName,GENERIC_READ,FILE_SHARE_READ,NULL,OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,0
                                                .if eax!=INVALID_HANDLE_VALUE
                                                        mov hFile,eax

                ;=====================================================
                        ; stream the text into the richedit control

                ;=====================================================
                                mov editstream.dwCookie,eax
                                mov editstream.pfnCallback,offset StreamInProc
                                invoke SendMessage,hwndRichEdit,EM_STREAMIN,SF_TEXT,addr editstream

                ;=====================================================
                                ; Initialize the modify state to false

                ;=====================================================
                        invoke SendMessage,hwndRichEdit,EM_SETMODIFY,FALSE,0
                                                        invoke CloseHandle,hFile
                                                        mov FileOpened,TRUE
                                                .else
                        invoke MessageBox,hWnd,addr OpenFileFail,addr AppName,MB_OK or MB_ICONERROR
                                                .endif
                                        .endif
                                .elseif ax==IDM_CLOSE
                                        invoke CheckModifyState,hWnd
                                        .if eax==TRUE
                                                invoke SetWindowText,hwndRichEdit,0
                                                mov FileOpened,FALSE
                                        .endif
                                .elseif ax==IDM_SAVE
                        invoke CreateFile,\
                                     addr FileName,GENERIC_WRITE,FILE_SHARE_READ,\
                                    NULL,CREATE_ALWAYS,FILE_ATTRIBUTE_NORMAL,0
                                        .if eax!=INVALID_HANDLE_VALUE
@@:
                                                mov hFile,eax
```


www.mohandesyar.com

```
;=====================================================
                                    ; stream the text to the file

;=====================================================
                                    mov editstream.dwCookie,eax
                                    mov editstream.pfnCallback,offset StreamOutProc
 invoke SendMessage,hwndRichEdit,EM_STREAMOUT,SF_TEXT,addr editstream

;=====================================================
                                    ; Initialize the modify state to false

;=====================================================
                            invoke SendMessage,hwndRichEdit,EM_SETMODIFY,FALSE,0
                                    invoke CloseHandle,hFile
                            .else
            invoke MessageBox,hWnd,addr OpenFileFail,addr AppName,MB_OK or MB_ICONERROR
                            .endif
                    .elseif ax==IDM_COPY
                            invoke SendMessage,hwndRichEdit,WM_COPY,0,0
                    .elseif ax==IDM_CUT
                            invoke SendMessage,hwndRichEdit,WM_CUT,0,0
                    .elseif ax==IDM_PASTE
                            invoke SendMessage,hwndRichEdit,WM_PASTE,0,0
                    .elseif ax==IDM_DELETE
                            invoke SendMessage,hwndRichEdit,EM_REPLACESEL,TRUE,0
                    .elseif ax==IDM_SELECTALL
                            mov chrg.cpMin,0
                            mov chrg.cpMax,-1
                            invoke SendMessage,hwndRichEdit,EM_EXSETSEL,0,addr chrg
                    .elseif ax==IDM_UNDO
                            invoke SendMessage,hwndRichEdit,EM_UNDO,0,0
                    .elseif ax==IDM_REDO
                            invoke SendMessage,hwndRichEdit,EM_REDO,0,0
                    .elseif ax==IDM_OPTION
                            invoke DialogBoxParam,hInstance,IDD_OPTIONDLG,hWnd,addr OptionProc,0
                    .elseif ax==IDM_SAVEAS
                            invoke RtlZeroMemory,addr ofn,sizeof ofn
                            mov ofn.lStructSize,sizeof ofn
                            push hWnd
                            pop ofn.hwndOwner
                            push hInstance
                            pop ofn.hInstance
                            mov ofn.lpstrFilter,offset ASMFilterString
                            mov ofn.lpstrFile,offset AlternateFileName
                            mov byte ptr [AlternateFileName],0
                            mov ofn.nMaxFile,sizeof AlternateFileName
            mov ofn.Flags,OFN_FILEMUSTEXIST or OFN_HIDEREADONLY or OFN_PATHMUSTEXIST
                            invoke GetSaveFileName,addr ofn
                            .if eax!=0
            invoke CreateFile,addr AlternateFileName,GENERIC_WRITE,FILE_SHARE_READ,\
                                NULL,CREATE_ALWAYS,FILE_ATTRIBUTE_NORMAL,0
                                    .if eax!=INVALID_HANDLE_VALUE
                                            jmp @B
                                    .endif
                            .endif
                    .elseif ax==IDM_EXIT
                            invoke SendMessage,hWnd,WM_CLOSE,0,0
```


www.mohandesyar.com

```
                        .endif
                .endif
        .elseif uMsg==WM_CLOSE
                invoke CheckModifyState,hWnd
                .if eax==TRUE
                        invoke DestroyWindow,hWnd
                .endif
        .elseif uMsg==WM_SIZE
                mov eax,lParam
                mov edx,eax
                and eax,0FFFFh
                shr edx,16
                invoke MoveWindow,hwndRichEdit,0,0,eax,edx,TRUE
        .elseif uMsg==WM_DESTROY
                invoke PostQuitMessage,NULL
        .else
                invoke DefWindowProc,hWnd,uMsg,wParam,lParam
                ret
        .endif
        xor eax,eax
        ret
WndProc endp
end start

;=====================================================================
; The resource file
;=====================================================================
#include "resource.h"
#define IDR_MAINMENU                101
#define IDD_OPTIONDLG               101
#define IDC_BACKCOLORBOX              1000
#define IDC_TEXTCOLORBOX              1001
#define IDM_OPEN                    40001
#define IDM_SAVE                    40002
#define IDM_CLOSE                    40003
#define IDM_SAVEAS                   40004
#define IDM_EXIT                    40005
#define IDM_COPY                     40006
#define IDM_CUT                     40007
#define IDM_PASTE                    40008
#define IDM_DELETE                   40009
#define IDM_SELECTALL                 40010
#define IDM_OPTION                   40011
#define IDM_UNDO                    40012
#define IDM_REDO                    40013

IDR_MAINMENU MENU DISCARDABLE
BEGIN
   POPUP "&File"
   BEGIN
      MENUITEM "&Open",                  IDM_OPEN
      MENUITEM "&Close",                 IDM_CLOSE
      MENUITEM "&Save",                  IDM_SAVE
      MENUITEM "Save &As",                IDM_SAVEAS
      MENUITEM SEPARATOR
      MENUITEM "E&xit",                  IDM_EXIT
   END
   POPUP "&Edit"
```


www.mohandesyar.com

```
    BEGIN
        MENUITEM "&Undo",                IDM_UNDO
        MENUITEM "&Redo",                IDM_REDO
        MENUITEM "&Copy",                IDM_COPY
        MENUITEM "C&ut",                 IDM_CUT
        MENUITEM "&Paste",               IDM_PASTE
        MENUITEM SEPARATOR
        MENUITEM "&Delete",              IDM_DELETE
        MENUITEM SEPARATOR
        MENUITEM "Select &All",          IDM_SELECTALL
    END
    MENUITEM "Options",                  IDM_OPTION
END


IDD_OPTIONDLG DIALOG DISCARDABLE  0, 0, 183, 54
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU | DS_CENTER
CAPTION "Options"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,137,7,39,14
    PUSHBUTTON      "Cancel",IDCANCEL,137,25,39,14
    GROUPBOX        "",IDC_STATIC,5,0,124,49
    LTEXT           "Background Color:",IDC_STATIC,20,14,60,8
    LTEXT           "",IDC_BACKCOLORBOX,85,11,28,14,SS_NOTIFY | WS_BORDER
    LTEXT           "Text Color:",IDC_STATIC,20,33,35,8
    LTEXT           "",IDC_TEXTCOLORBOX,85,29,28,14,SS_NOTIFY | WS_BORDER
END
```

بررسی کد فوق:

در ابتدا، برنامه dll مربوط به richedit را به نام riched20.dll بارگذاری می کند. در صورت شکست برنامه خاتمه خواهد یافت.

```
invoke LoadLibrary,addr RichEditDLL
.if eax!=0
        mov hRichEdit,eax
        invoke WinMain,hInstance,0,0, SW_SHOWDEFAULT
        invoke FreeLibrary,hRichEdit
.else
        invoke MessageBox,0,addr NoRichEdit,addr AppName,MB_OK or MB_ICONERROR
.endif
invoke ExitProcess,eax
```

پس از بارگذاری موفقیت آمیز dll ، پنجره اصلی برنامه ایجاد خواهد شد. درحین پردازش پیغام **WM_CREATE** ، کنترل richedit ایجاد خواهد شد.

```
invoke CreateWindowEx,WS_EX_CLIENTEDGE,addr RichEditClass,\
0,WS_CHILD or WS_VISIBLE or ES_MULTILINE or WS_VSCROLL or WS_HSCROLL or ES_NOHIDESEL,\
CW_USEDEFAULT,CW_USEDEFAULT, CW_USEDEFAULT,\
W_USEDEFAULT,hWnd,RichEditID,hInstance,0
mov hwndRichEdit,eax
```


www.mohandesyar.com

لازم به ذکر است که اگر سبک **ES_MULTILINE** مشخص نشود، کنترل تک خطی خواهد بود.

```
invoke  SendMessage,hwndRichEdit,EM_LIMITTEXT,-1,0
```

پس از ایجاد کنترل، باید حداکثر اندازه متن قابل دریافت مشخص شود. به صورت پیش فرض این حد همان ۶۴ کیلوبایت متداول برای کنترل تکست باکس معمولی است که باید توسعه داده شود. در اینجا با قرار دادن منفی یک، حجم 0FFFFFFFFh در نظر گرفته شده است.در ادامه :

```
invoke  SetColor
```

در اینجا رنگ متن و زمینه تنظیم خواهند شد و از تابعی تعریف شده به نام SetColor برای این منظور استفاده گردید.

```
SetColor proc
        LOCAL cfm:CHARFORMAT
invoke SendMessage,hwndRichEdit,EM_SETBKGNDCOLOR,0,BackgroundColor
```

برای تعیین رنگ زمینه به سادگی از پیغام **EM_SETBKGNDCOLOR** استفاده می‌شود. اگر از کنترل چند خطی استفاده می‌شود باید **WM_CTLCOLOREDIT** پردازش شود.رنگ پیش فرض زمینه سفید است.

```
invoke RtlZeroMemory,addr cfm,sizeof cfm
mov cfm.cbSize,sizeof cfm
mov cfm.dwMask,CFM_COLOR
push TextColor
pop cfm.crTextColor
```

پس از تنظیم رنگ پس زمینه ، اعضای ساختار **CHARFORMAT** را جهت تنظیم رنگ متن مقدار دهی خواهیم کرد.از آنجائیکه ما عضو **cbSize** ساختار را با اندازه ساختار فوق مقدار دهی کرده ایم ، کنترل richedit درخواهد یافت که ساختار **CHARFORMAT** به آن ارسال شده است و نه ساختار **CHARFORMAT2** .

dwMask تنها با پرچم **CFM_COLOR** مقدار دهی شده است، زیرا ما تنها قصد داریم رنگ متن را تنظیم کنیم و مقدار رنگ در عضو **crTextColor** قرار خواهد گرفت.

```
invoke SendMessage,hwndRichEdit,EM_SETCHARFORMAT,SCF_ALL,addr cfm
ret
SetColor endp
```

پس از تعیین رنگ، لازم است تا undo buffer خالی شود ، زیرا این عملیات غیرقابل لغو است. با ارسال **EM_EMPTYUNDOBUFFER** ، این امر محقق می‌شود.

```
invoke SendMessage,hwndRichEdit,EM_EMPTYUNDOBUFFER,0,0
```

پس از پرکردن ساختار **CHARFORMAT** ، پیغام **EM_SETCHARFORMAT** را به کنترل ارسال کرده و با مشخص کردن مقدار پرچم **SCF_ALL** در **wParam** ، تعیین می‌نمائیم که اعمال قالب بر روی تمام متن باید صورت گیرد.

www.mohandesyar.com

زمانیکه کنترل richedit ایجاد می‌شود، اندازه و مکان آنرا مشخص نمی‌نمائیم. اینکار در هنگام تغییر اندازه پنجره اصلی برنامه جهت پوشاندن تمام ناحیه کلاینت ، انجام خواهد شد:

```
.elseif uMsg==WM_SIZE
        mov eax,lParam
        mov edx,eax
        and eax,0FFFFh
        shr edx,16

        invoke MoveWindow,hwndRichEdit,0,0,eax,edx,TRUE
```

اندازه جدید ناحیه کلاینت در **lParam** قرار گرفته و از آن برای تغییر اندازه کنترل استفاده خواهیم کرد.

هنگامیکه کاربر بر روی منوی File/Edit کلیک می کند ، پیغام **WM_INITPOPUPMENU** را پردازش خواهیم کرد. بنابراین امکان بررسی و تعیین حالت ساب منوها قبل از نمایش آنها میسر است. برای مثال اگر فایلی هم اکنون در برنامه باز شده است، می توان آیتم open را غیرفعال و سایر آیتم ها را فعال نمود. در حالت منوی فایل، متغیر **FileOpened** را بررسی کرده و اگر مقدار آن TRUE بود، نتیجه گرفته خواهد شد که فایلی هم اکنون در برنامه باز شده است.

```
.elseif uMsg==WM_INITMENUPOPUP
        mov eax,lParam
        .if ax==0               ; file menu
                .if FileOpened==TRUE            ; a file is already opened
                        invoke EnableMenuItem,wParam,IDM_OPEN,MF_GRAYED
                        invoke EnableMenuItem,wParam,IDM_CLOSE,MF_ENABLED
                        invoke EnableMenuItem,wParam,IDM_SAVE,MF_ENABLED
                        invoke EnableMenuItem,wParam,IDM_SAVEAS,MF_ENABLED
                .else
                        invoke EnableMenuItem,wParam,IDM_OPEN,MF_ENABLED
                        invoke EnableMenuItem,wParam,IDM_CLOSE,MF_GRAYED
                        invoke EnableMenuItem,wParam,IDM_SAVE,MF_GRAYED
                        invoke EnableMenuItem,wParam,IDM_SAVEAS,MF_GRAYED
                .endif
```

در حالت منوی ادیت، نیاز است تا وضعیت richedit control/clipboard بررسی شود.

```
                invoke SendMessage,hwndRichEdit,EM_CANPASTE,CF_TEXT,0
                .if eax==0                      ; no text in the clipboard
                        invoke EnableMenuItem,wParam,IDM_PASTE,MF_GRAYED
                .else
                        invoke EnableMenuItem,wParam,IDM_PASTE,MF_ENABLED
                .endif
```

جهت بررسی اینکه آیا متنی در clipboard وجود دارد یا خیر از پیغام **EM_CANPASTE** استفاده خواهد شد. اگر متنی وجود داشته باشد، تابع SendMessage مقدار TRUE را بازگشت داده و آیتم paste را فعال خواهد کرد. در غیر اینصورت منوی مربوط به آنرا خاکستری خواهیم کرد:

```
invoke SendMessage,hwndRichEdit,EM_CANUNDO,0,0
```


www.mohandesyar.com

```
.if eax==0
        invoke EnableMenuItem,wParam,IDM_UNDO,MF_GRAYED
.else
        invoke EnableMenuItem,wParam,IDM_UNDO,MF_ENABLED
.endif
```

سپس وضعیت خالی بودن بافر **undo** را با فرستادن **EM_CANUNDO** بررسی خواهیم کرد.اگر خالی نباشد، SendMessage مقدار TRUE را برخواهد گرداند و منوی مربوطه را فعال خواهیم کرد:

```
invoke SendMessage,hwndRichEdit,EM_CANREDO,0,0
.if eax==0
        invoke EnableMenuItem,wParam,IDM_REDO,MF_GRAYED
.else
        invoke EnableMenuItem,wParam,IDM_REDO,MF_ENABLED
.endif
```

همچنین redo buffer را نیز با ارسال پیغام **EM_CANREDO** به richedit بررسی خواهیم کرد. اگر خالی نباشد، تابع SendMessage ، مقدار TRUE را برخواهد گرداند و منوی مربوطه را فعال خواهیم کرد:

```
invoke SendMessage,hwndRichEdit,EM_EXGETSEL,0,addr chrg
mov eax,chrg.cpMin
.if eax==chrg.cpMax                     ; no current selection
        invoke EnableMenuItem,wParam,IDM_COPY,MF_GRAYED
        invoke EnableMenuItem,wParam,IDM_CUT,MF_GRAYED
        invoke EnableMenuItem,wParam,IDM_DELETE,MF_GRAYED
.else
        invoke EnableMenuItem,wParam,IDM_COPY,MF_ENABLED
        invoke EnableMenuItem,wParam,IDM_CUT,MF_ENABLED
        invoke EnableMenuItem,wParam,IDM_DELETE,MF_ENABLED
.endif
```

در آخر با ارسال پیغام **EM_EXGETSEL** ، بررسی خواهیم کرد که آیا متنی انتخاب شده است یا خیر؟ این پیغام از ساختار **CHARRANGE** که به صورت زیر تعریف می‌شود استفاده می‌کند:

```
CHARRANGE  STRUCT
  cpMin  DWORD     ?
  cpMax  DWORD     ?
CHARRANGE  ENDS
```

cpMin : حاوی اندیس حرفی است که پیش از اولین حرف مشخص شده در بازه ، قرار گرفته است.

cpMax : حاوی اندیس حرفی است که پس از آخرین حرف مشخص شده در بازه ، قرار گرفته است.

پس از پایان کار پیغام **EM_EXGETSEL** ، ساختار **CHARRANGE** مقدار دهی خواهد شد. اگر دو عضو آن یکسان بودند بدین معنا است که متنی انتخاب نشده است و منوی cut/copy/delete را غیرفعال خواهیم کرد.

www.mohandesyar.com

زمانیکه کاربر بر روی گزینه open کلیک کند، open file dialog box نمایش داده شده و در صورت انتخاب فایلی توسط کاربر ، محتوای آنرا در کنترل نمایش خواهیم داد.

```
invoke CreateFile,\
                addr FileName,GENERIC_READ,\
                FILE_SHARE_READ,NULL,OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,0
                   .if eax!=INVALID_HANDLE_VALUE
                   mov hFile,eax
                   mov editstream.dwCookie,eax
                   mov editstream.pfnCallback,offset StreamInProc
invoke SendMessage,hwndRichEdit,\
                   EM_STREAMIN,SF_TEXT,addr editstream
```

پس از اینکه فایل بصورت موفقیت آمیزی توسط **CreateFile** گشوده شد، ساختار **EDITSTREAM** را جهت استفاده در ارسال پیغام **EM_STREAMIN** ، مقدار دهی خواهیم کرد. دستگیره فایل باز شده از طریق **dwCookie** و آدرس تابع callback از طریق عضو **pfnCallback** ارسال می گردد.این تابع به صورت زیر پیاده سازی می‌شود:

```
StreamInProc proc hFile:DWORD,pBuffer:DWORD, NumBytes:DWORD, pBytesRead:DWORD
        invoke ReadFile,hFile,pBuffer,NumBytes,pBytesRead,0
        xor eax,1
        ret
StreamInProc endp
```

همانطور که ملاحظه می‌نمائید، پارامترهای تابع callback با تابع **ReadFile** تطابق کامل دارند.همچنین خروجی آن نیز با یک xor شده است. بنابراین اگر خروجی تابع ReadFile موفقیت آمیز باشد، مقدار آن با xor شدن با یک مساوی صفر شده و نهایتا از تابع بازگشت داده می‌شود.

```
invoke SendMessage,hwndRichEdit,EM_SETMODIFY,FALSE,0
invoke CloseHandle,hFile
mov FileOpened,TRUE
```

پایان کار پیغام **EM_STREAMIN** ، به معنای پایان یافتن کار خواندن فایل است. در شرایط واقعی تر باید مقدار **dwError** عضو ساختار **EDITSTREAM** بررسی شود.

کنترل Richedit و همچنین تکست باکس معمولی ، پرچمی را جهت باخبر سازی از تغییر محتوای آنها ارائه می‌دهند. مقدار این پرچم با ارسال پیغام **EM_GETMODIFY** به کنترل بدست می‌آید. درصورتیکه محتوای این کنترل تغییر کرده باشد، خروجی **SendMessage** مساوی TRUE خواهد بود. از آنجائیکه مقدار دهی اولیه این کنترل نیز نوعی تغییر محسوب می شود باید پس از پایان دریافت محتوای فایل و نمایش آن ، پیغام **EM_SETMODIFY** را به همراه wParam==FALSE به کنترل ارسال نمائیم. همچنین بلافاصله فایل را بسته و پرچم **FileOpened** را به TRUE تنظیم خواهیم کرد.

www.mohandesyar.com

زمانیکه کاربر بر روی گزینه های save/saveas کلیک نماید، از پیغام **EM_STREAMOUT** جهت ذخیره سازی محتوای آن استفاده خواهیم کرد. تابع callback آن با WriteFile بخوبی مطابقت دارد.

اعمال متنی مانند cut/copy/paste/redo/undo ، با ارسال پیغامهای **WM_CUT/WM_COPY/WM_PASTE/WM_REDO/WM_UNDO** به کنترل قابل انجام است.

اعمال delete/select all به صورت زیر انجام می‌شود:

```
.elseif ax==IDM_DELETE
        invoke SendMessage,hwndRichEdit,EM_REPLACESEL,TRUE,0
.elseif ax==IDM_SELECTALL
        mov chrg.cpMin,0
        mov chrg.cpMax,-1
        invoke SendMessage,\
                hwndRichEdit,\
                EM_EXSETSEL,0,addr chrg
```

عملیات حذف بر روی متن انتخابی تاثیر گذار است . به همین منظور پیغام **EM_REPLACESEL** به همراه یک رشته مختوم به نال جهت این منظور استفاده می گردد.

عملیات انتخاب تمامی متن با ارسال پیغام **EM_EXSETSEL** انجام می‌شود (به همراه **cpMin**==0 and **cpMax**==-1 که سبب انتخاب تمام متن می گردد).

زمانیکه کاربر از منوی برنامه گزینه انتخاب رنگ را برگزیند ، دیالوگ باکس انتخاب رنگ نمایش داده خواهد شد. هر کدام از مستطیل های رنگی توسط **SS_NOTIFY** ، پنجره والد را برای مثال از کلیک شدن بر روی خود (**BN_CLICKED**) مطلع می سازند.

```
.elseif ax==IDC_BACKCOLORBOX
        invoke RtlZeroMemory,addr clr,sizeof clr
        mov clr.lStructSize,sizeof clr
        push hWnd
        pop clr.hwndOwner
        push hInstance
        pop clr.hInstance
        push BackgroundColor
        pop clr.rgbResult
        mov clr.lpCustColors,offset CustomColors
        mov clr.Flags,CC_ANYCOLOR or CC_RGBINIT
        invoke ChooseColor,addr clr
        .if eax!=0
                push clr.rgbResult
                pop BackgroundColor
                invoke GetDlgItem,hWnd,IDC_BACKCOLORBOX
                invoke InvalidateRect,eax,0,TRUE
        .endif
```


www.mohandesyar.com

زمانیکه کاربر بر روی یکی از مستطیل های انتخاب رنگ کلیک کند، ساختار **CHOOSECOLOR** را مقدار دهی کرده و سپس تابع **ChooseColor** را فراخوانی خواهیم کرد. اگر کاربر رنگی را انتخاب نماید، مقدار colorref ، در عضو **rgbResult** بازگشت داده شده و مقدار آنرا در متغیر **BackgroundColor** ذخیره خواهیم کرد. پیغام **InvalidateRect** از طرف مستطیل های رنگی به پنجره والد ارسال می‌شود.دستگیره مستطیل انتخابی در **lParam** قرار خواهد گرفت.

```
invoke GetDlgItem,hWnd,IDC_BACKCOLORBOX
.if eax==lParam
        invoke CreateSolidBrush,BackgroundColor
        ret
```

سپس برسی را با استفاده از رنگ انتخابی ایجاد و ازآن برای رنگ آمیزی پس زمینه استفاده خواهیم کرد.

www.mohandesyar.com

درخواستی از خواننده:

اگـر در مـتن فـوق خطـایی را مـشاهده فرمودیـد (فنـی یـا غیرفنـی) ، لطفـا آنـرا بـا نویـسنده بـه آدرس vahid_nasiri[at]yahoo.com مطرح بفرمایید.

مآخذ:

1- Iczelion's Win32 Assembly tutorials. Web: http://win32assembly.online.fr/tutorials.html
2- Win32Asm Tutorial by Thomas Bleeker. Web: http://www.madwizard.org
3- Masm32's help files.
4- An Introduction to Assembly Language I/II/III by Darwen.
Web: http://www.codeguru.com/Cpp/Cpp/cpp_mfc/tutorials/article.php/c9411
5- Win32 Assembly parts 1-6 by Chris Hobbs.
Web: http://www.gamedev.net/reference/list.asp?categoryid=20#101
6- Win32 Assembler Coding Tutorial. Web: http://www.deinmeister.de/w32asm1e.htm

www.mohandesyar.com