

# Creating GUIs in pure assembly

Dialogs design, getting your hands dirty

written by: Xacker

بسم الله الرحمن الرحيم

## - مقدمة:

هذا المقال على عكس المقال السابق له سيقوم بتوضيح الخطوات اللازمة لبناء او كتابة واجهة مرئية لبرامج 32bit من خلال كتابة الكود اللازم لتشكيل تلك النوافذ وذلك بالاعتماد بشكل كامل على دوال API.

## - لمن هذا المقال؟

هذا المقال مخصص لمن لم يقرأ مقالات Iczelion من قبل, او قام بقراءتها ولم يفهم الاجزاء الخاصة بتشكيل تلك الواجهات (النوافذ).

## - لماذا مقالات Iczelion؟

مقالات Iczelion هي بداية جيدة جدا لمن يقوم بتعلم الاسبلي بنفسه من خلال الاطلاع على المصادر الجاهزة وتنقيحها او كتابة اجزاء مماثلة, يمكن اعتبار هذه المقالات مدخل الى البرمجة المرئية بواسطة الاسبلي الا انها من ناحية اخرى لا تناقش النواحي الفيزيائية الخاصة بالحاسب والتي قمنا باذن الله بتغطيتها في مقالات سابقة. ام انك تريد الانتقال الى HLA مباشرة؟

## - الادوات اللازمة للعمل:

سنحتاج في هذا المقال الى البرامج التالية:

RadAsm -1

Masm32 -2 (لعمل بواسطة RadAsm)

3- موسيقى so close.xml لتهدئة الاعصاب.

## - التحضير للبدء:

المقال يفترض انك قمت باعداد Masm32 و RadAsm للعمل معه, هذا الافتراض ينبع



من انه "عيب يعني بدك تتعلم اسبلي وما بتعرف تستخدم برنامج!" وقد تم النقاش في هذا الامر عدة مرات في القسم لذا ابحت عن هذا الموضوع.

## - الخطوات اللازمة لتشكيل اي نافذة:

مقتبس من مقالة Iczelion الثالثة (A simple window):

- 1- الحصول على مقبض البرنامج (ضروري)
- 2- الحصول على محتويات محث الاوامر (غير ضروري الا في الحالة التي يحتاج فيها برنامجك لمعالجة اوامر مسبقة للتنفيذ, كان يكون تطبيق console على سبيل المثال)
- 3- تسجيل Class النافذة (ضروري الا في حالة النوافذ المعرفة من قبل النظام سلفا)
- 4- انشاء النافذة (ضروري)
- 5- اظهار النافذة (ضروري, الا في الحالة التي لا تريد فيها اظهار النافذة بشكل مباشر)

- 6- تحديث شاشة العرض لظهور النافذة.
- 7- الدخول في حلقة غير منتهية في انتظار اي رسالة موجهة من البرنامج.
- 8- عند استقبال اي رسالة معالجتها وفق الكود المكتوب.
- 9- انتهاء البرنامج عندما يتم اغلاق النافذة الرئيسية.

والان ننتقل الى كتابة اجزاء البرنامج وشرحها بشئ من التفصيل باذن الله.

### - التوجيهات, التصريحات والمقاطع:

كما ناقشنا في مقالات سابقة فان اي مجمع يحتاج الى بضع معلومات في البداية كي يتابع عمله دون مشاكل, الاسطر هذه ستكون بالشكل التالي:

```
.386
.model flat, stdcall
option casemap: none

include SimpleWindow.inc

.code
start:
```

وكما راينا سابقا فان ملف SimpleWindow.inc سيحوي على المسارات اللازمة الى المكتبات الضرورية للمجمع اضافة الى مقطعي .data و .data?. واي معلومات اخرى ضرورية.. طبعا لاحظ ان المسارات ليست x:\masm32\... لانك قمت باعداد RadAsm في البداية وحددت دليل هذه المكاتب سلفا لذا يكفي ذكر الاسم وسيتكفل RadAsm باكمالها (اكمال تلقائي) وتحديد المسار اللازم اثناء عملية الربط والتجميع.

```
include windows.inc
include kernel32.inc
include user32.inc
includelib kernel32.lib
includelib user32.lib

.data?

.data
```

هناك بعض التصريحات التي سنقوم باضافتها فيما بعد وسيتم التنبيه اليها عندها.

### - عودة الى الخطوات الاساسية في تشكيل النافذة:

نعود الان الى الخطوات التي ذكرها Iczelion في تشكيل النافذة ونبدأ باول خطوة.

- الحصول على مقبض البرنامج: المقبض = handle وهو الذي نستطيع بواسطته التخاطب مع برنامجنا بالتحديد, يتم الحصول على هذا المقبض من خلال دالة [GetModuleHandle](#) والتي استخدمناها في بداية كل برنامج 32bit قمنا بكتابته من قبل.

للحصول على مقبض البرنامج نقوم باستخدام السطر:

```
invoke GetModuleHandle,0 ; get the handle
```

```
mov hInstance,eax ; save it in hInstance
```

ويتم هذا الامر في بداية مقطع `code`. بعد `start` مباشرة, لكن, هل تذكر ما ذكرته في المقال السابق, ان اردت استخدام 0 بدلا من NULL فانا بحاجة الى استخدام التوجيه `equ` كي اقوم بالتصريح عن المساواة بين NULL و 0. لذا نتوجه الى ملف SimpleWindow.inc ونقوم باضافة السطر:

```
.const
NULL equ 0

.data?
hInstance dd ?
```

حيث يكون التوجيه `const`. خاصا بالثوابت, ايضا لا تنسى ان تقوم باضافة `hInstance` تحت `.data?`.

• الخطوة الثانية: الحصول على محتويات محث الاوامر, غير ضرورية في هذا المقال لكن للتذكير يتم استخدام الدالة `GetCommandLine` لهذا الغرض.

• تسجيل class النافذة: تسبق عملية تسجيل class النافذة واحدة من اهم العمليات وهي بناء الكتلة الخاصة بالنافذة مع معلوماتها.. فكما راينا في المقال الماضي اننا قمنا ببناء كتلة `DlgProc` الخاصة بمعالجة الرسائل الموجهة من البرنامج, بنفس الطريقة لكن باختلاف واضح سنقوم بكتابة كتلة تحوي على جميع المعلومات الخاصة بالنافذة قبل تسجيلها.

سنقوم بهذا الامر عبر كتلة سنطلق عليها اسم `WinMain` والتي هي عبارة عن اجرائية او وظيفة مصرح عنها من قبل النظام وتملك اربع بارامترات, `hInstance` الذي هو عبارة عن مقبض البرنامج الذي قمنا بالحصول عليه من الخطوة الاولى, `hPrevInstance` والذي نقوم باعطائه القيمة 0 دوما (لمزيد من المعلومات عنه راجع MSDN), `lpCmdLine` والذي يشير الى اي عبارة موجودة في محث الاوامر لمعالجتها والذي ايضا نقوم باعطائه القيمة 0 لاننا لسنا بحاجة اليه, `nCmdShow` والذي يحدد الحالة التي ستظهر بها النافذة وسنناقش هذا بعد قليل. اذن يكون شكل هذه الكتلة كالتالي:

```
WinMain proc
hInst:HINSTANCE,hPrevInst:HINSTANCE,CmdLine:LPSTR,CmdShow:DWORD
```

ونقوم بتوجيه القيم الخاصة بهذه البارامترات عبر استدعاء `WinMain` بالشكل التالي:

```
invoke WinMain,hInstance,0,0,SW_SHOWDEFAULT
```

الان عندما يقوم البرنامج بالوصول الى هذه الدالة للقيام بتنفيذها فانه يقوم باستدعاء الكتلة `WinMain` والتي قمنا بالتصريح عنها ليقوم بتنفيذ الاجرائية المكتوبة فيها.

نعود الان الى تسجيل ال class الخاص بالنافذة, للقيام بهذا الامر يتم استدعاء `RegisterClass` او `RegisterClassEx` والتي تملك بارامتر واحد يشير الى كتلة `WNDCLASSEX` (او `WNDCLASS`) والتي يجب ملؤها بالمعلومات الضرورية للنافذة قبل تسجيلها. اذن سنقوم بحجز جزء من المكس لهذه الكتلة من خلال توجيه

LOCAL والذي يقوم بحجز جزء من المكس للمتغير المطلوب وفق حجمه المصرح عنه. بما ان هذه الكتلة مصرح عنها في النظام فلسنا بحاجة الى تحديد الحجم فنقوم مباشرة باستخدام السطر:

```
LOCAL wc:WNDCLASSEX
```

بعد تصريح الكتلة WinMain مباشرة.  
التوجيه LOCAL يقوم باخذ قيمتين, الاولى هي اسم المتغير (في هذه الحالة wc), والقيمة الثانية هي نوع المتغير (WNDCLASSEX).

الان ناتي الى ملئ الكتلة التي قمنا بتعريفها wc بالمعلومات اللازمة الخاصة بـ WNDCLASSEX, ان هذه الكتلة تملك المتغيرات التالية والتي سنحتاج الى ملئها بالقيم اللازمة.

```
typedef struct {
    UINT      cbSize;
    UINT      style;
    WNDPROC    lpfnWndProc;
    int       cbClsExtra;
    int       cbWndExtra;
    HINSTANCE hInstance;
    HICON      hIcon;
    HCURSOR    hCursor;
    HBRUSH     hbrBackground;
    LPCTSTR    lpstrMenuName;
    LPCTSTR    lpstrClassName;
    HICON      hIconSm;
} WNDCLASSEX, *PWNDCLASSEX;
```

كيف نقوم اذن بمعرفة القيم الضرورية وحفظها في المتغيرات الموافقة؟  
الامر بسيط في الواقع, طالما انك قمت بحجز الكتلة wc وتعريفها بانها تابعة لكتلة WNDCLASSEX فاذن يمكنك استخدام الشكل

```
mov wc.cbSize, sizeof WNDCLASSEX
```

على سبيل المثال لوضع القيمة الخاصة بحجم هذه الكتلة في المتغير cbSize عبر استخدام ماكرو (حسبما اذكر) هو sizeof مصرح عنه سلفا ضمن المكاتب.  
بنفس الطريقة يتم وضع القيم الاخرى, لن اقوم بشرحها لان كل ما يحتاجه الامر لفهمها هو الاطلاع على كتلة WNDCLASSEX ضمن MSDN ولا اعتقد ان في ذلك مشكلة.

```
mov wc.style, CS_HREDRAW or CS_VREDRAW
mov wc.lpfnWndProc, OFFSET WndProc
mov wc.cbClsExtra, 0
mov wc.cbWndExtra, 0
push hInstance
pop wc.hInstance
```

```

mov     wc.hbrBackground,COLOR_BTNFACE+1
mov     wc.lpszMenuName,0
mov     wc.lpszClassName,OFFSET ClassName
invoke  LoadIcon,0,IDI_APPLICATION
mov     wc.hIcon,eax
mov     wc.hIconSm,eax
invoke  LoadCursor,0,IDC_ARROW
mov     wc.hCursor,eax
invoke  RegisterClassEx,addr wc

```

لاحظ فقط اننا بحاجة الى اضافة متغير اخر هو `ClassName` والذي سيحوي على الاسم المراد اعطاؤه للـ `class`.

```

.data
ClassName db "SimpleWindow",0

```

يجب ان تدرك ان هذه الكتلة تقريبا ثابتة في جميع البرامج التي قد تريد انشاءها لاحقا لذا يمكنك بعد ان تقوم بفهمها بالاستغناء عن كتابتها مرة تلو الاخرى وتكتفي بنسخها ولصقها مباشرة مع اجراء التعديلات التي تراها مناسبة لبرنامجك الجديد.

• انشاء النافذة: في هذه الخطوة نقوم باستدعاء الدالة `CreateWindowEx` لانشاء النافذة والحصول على مقبضها.

```

invoke  CreateWindowEx,\
        WS_EX_STATICEDGE,\
        addr ClassName,\
        addr WindowName,\
        WS_OVERLAPPEDWINDOW,\
        CW_USEDEFAULT,\
        CW_USEDEFAULT,\
        250,\ ; width of the window
        150,\ ; height of the window
        0,0,hInst,0

```

السطر طويل, لذلك يمكنك ان تقوم بتجزئته باستخدام `\` عند نهاية البارامتر الذي تريد انهاء السطر به.

بهذا يتم انشاء النافذة يتم اعادة مقبض النافذة في `eax`, لذا نقوم باخذه وحفظه في متغير جديد نقوم ايضا باضافته باسم `hWnd` مثلا.

```

mov     hWnd,eax

.data?
hInstance dd ?
hWnd      dd ?

.data
ClassName db "SimpleWindow",0

```

```
WindowName db "Simple Window Example",0
```

• اظهار النافذة: يتم هذا عبر استدعاء `ShowWindow` والتي تملك بارامترين, الاول هو مقبض النافذة والذي قمنا بحفظه في `hWnd` والثاني هو المتغير الذي قمنا بحفظ حالة ظهور النافذة فيه عند استدعاء `WinMain` وهو `CmdShow`.

```
invoke ShowWindow,hWnd,CmdShow
```

• تحديث شاشة العرض لاطهار النافذة: يتم هذا بواسطة `UpdateWindow`, والتي تملك بارامتر واحد هو مقبض النافذة `hWnd`.

```
invoke UpdateWindow,hWnd
```

• الدخول في حلقة لاستقبال وفلتر الرسائل ومعالجتها: نقوم بانشاء حلقة لا يتم الخروج منها الا في حالة واحدة فقط باستخدام التعبير `WHILE` حيث نقوم باسترجاع الرسالة في حال وجدت باستدعاء `GetMessage` ومن ثم نقوم بحفظ معلومات الرسالة في متغير محلي اسمه `msg` مثلا نقوم بالحجز له بواسطة `LOCAL` يشير الى كتلة `MSG` المعروفة. لاحظ ان الحلقة لا يتم الخروج منها الا عندما يتم استلام رسالة `WM_QUIT`. اخيرا في الحلقة يتم ارسال الرسالة لمعالجتها من قبل الكتلة المرافقة التي تكلمنا عنها في المقال السابق وهي `DlgProc` والتي سنطلق عليها في هذا المقال الاسم `WndProc`. يتم هذا من خلال استدعاء `DispatchMessage` والتي تقوم بارسال الرسالة الى `queue` الرسائل.

```
LOCAL msg:MSG
.while TRUE
    invoke GetMessage,addr msg,0,0,0
    .break .if (!eax)
    invoke DispatchMessage,addr msg
.endw
mov eax,msg.wParam
```

• معالجة الرسائل المستلمة وفق الكود المكتوب عبر كتلة `WndProc`: لا اعتقد ان هذه الكتلة ما زالت بحاجة الى توضيح فمحتوياتها اصبحت مالوفة من خلال المقال السابق.

```
WndProc proc
hWin:HWND,uMsg:UINT,wParam:WPARAM,lParam:LPARAM

mov     eax,uMsg
.if     eax==WM_DESTROY
    invoke PostQuitMessage,0
.else
    invoke DefWindowProc,hWnd,uMsg,wParam,lParam
.endif

ret

WndProc endp
```

#### - ملحوظة اخيرة:

تحتاج هذه الطريقة في انشاء النوافذ الى تعريف متحول محلي اضافي هو `hwnd` (اسم حسب رغبتك) يشير الى كتلة `HWND` والتي يتم استخدامها في `WndProc`. عدم التصريح عن هذا المتحول او التصريح عنه بصورة مماثلة لـ `hInstance` مثلا سيحدث خطأ في استعادة مقبض النافذة.

#### - كلمة اخيرة:

في المقال القادم باذن الله ساقوم بتغطية الافكار المتعلقة بانشاء العناصر من ازرار ومناطق ادخال نصية وما الى ذلك بنفس الاسلوب عبر تعليمات اسميلي فقط لكي ننتهي من هذا الجزء من تشكيل النوافذ ان شاء الله.

#### - اسئلة؟

<http://www.arabteam2000-forum.com/>  
may the tool be with you!