

Working with Resources for 32bit programs

Introduction to Dialogs design

written by: Xacker

بسم الله الرحمن الرحيم

- مقدمة:

الغاية من هذا المقال هو الاطلاع على كيفية استثمار البرامج التي تسمح بانشاء ملفات resource مع تطبيقات مكتوبة بالاسمبلي لتبسيط عملية تشكيل النوافذ وواجهات البرامج الرئيسية والفرعية.

كما نعلم فان نجاح التطبيق يعتمد ولو ب 20% على سهولة استخدام واجهته الرئيسية وتوفر الميزات التي تسهل التعامل معه لكافة شرائح المستخدمين. لذا كان من الضروري للمبرمجين الاهتمام بشكل الواجهة المرئية GUI (Graphical User Interface) وانشاء برمجيات ذات واجهات سهلة وواضحة الاستخدام.

لكن نعلم ايضا كم انه من السهولة انشاء مثل هذه الواجهات بواسطة لغات برمجة اخرى مثل VB او C واخواتها وانه من الصعوبة انشاء تطبيقات ذات واجهات متقدمة بلغات مثل Assembly, وهو ما سيدور محور نقاشنا عنه في هذا المقال.

- لمن هذا المقال؟

هذا المقال مخصص فقط لمن لم يقم بكتابة برنامج 32bit بواجهة مرئية باستخدام ملف res من قبل.

- الادوات اللازمة للعمل:

سنحتاج في هذا المقال الى البرامج التالية:

1- RadAsm

2- Masm32 (للعمل بواسطة RadAsm)

- التحضير للبدء:

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



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

- البدء بتشكيل ال Resource:

في البداية نقوم بفتح RadAsm, ننقر Ctrl+Shift+N لانشاء مشروع جديد, نختار نوع المشروع Win32 App واي اسم للمشروع, مثلا ResDlg ثم Next ونختار DialogApp (لانشاء واجهة افتراضية فورية للتطبيق) ثم Next واخيرا Finish.

الان سيشكل لنا RadAsm اربع ملفات هي: ResDlg.asm, ResDlg.inc, ResDlg.dlg, ResDlg.rc.

الملفات الجديدة في حال تابعت المقالات السابقة هي ResDlg.dlg, ResDlg.rc وهما ملفين احدهما خاص بمعلومات خاصة بالواجهة والاخر هو ملف الواجهة الافتراضي الذي

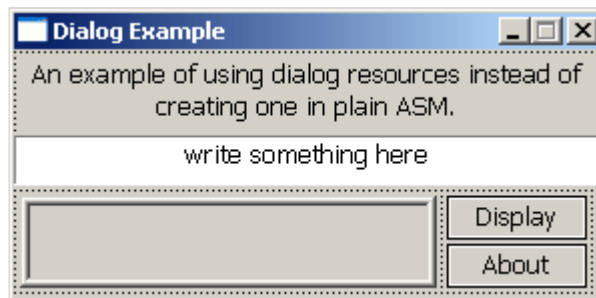
سنعامل معه. نلاحظ كيف ان RadAsm يوفر لنا امكانية انشاء الواجهات للتطبيقات بسهولة بسبب توفر ال templates الجاهزة معه. في الواقع انشاء ال resource بواسطة برامج اخرى مثل ++MSVC ليس بالامر الصعب ولا يختلف ابدا عن العملية هنا الا في عملية الربط بين ال resource وبين ملف ال obj بعد ان يتم كتابة البرنامج بواسطة Masm32 (منفردا دون RadAsm). سيكون الامر اضاعة للوقت لو اردنا نقاش هذا الامر بصورة منفصلة لذا يمكن لو احببت السؤال عنه في القسم في نفس الموضوع او لربما ساقوم بنقاشه في نهاية المقال لو توفر لي الوقت الكافي..

بعد ان قمنا الان بتشكيل الواجهة المبدئية للبرنامج سنقوم باضافة وتعديل بعض الامور المعروفة في الواجهات كالأزرار ومناطق الادخال النصية وغيرها من العناصر المعروفة.

السؤال الذي قد يتبادر الى الذهن الان كيف يمكن للبرنامج ان يميز بين النقر على احد زرین مثلا موجودين في الواجهة؟ اليس هناك طريقة لذلك؟ في الواقع الاجابة هي نعم، فكل عنصر في الواجهة بما فيها الواجهة نفسها تملك ما يسمى بال ID او الهوية لو اردنا تعريب هذا. هذا ال ID هو عبارة عن **رقم** يتميز به العنصر عن غيره من عناصر الواجهة. الامر مشابه للغات البرمجة الاخرى كالفيجوال مثلا، ففيها يتم تعريف كل عنصر بخاصة تسمى Name هي المسؤولة عن التمييز بين العناصر في الواجهة، في الاسمبلي الامر مختلف قليلا، فكل عنصر له Name يمكن ان يتم التمييز بواسطته، وللعنصر ايضا ID يمكن ان يتم التمييز بواسطته بالاعتماد على ال Name. قد تبدو الامور غير مفهومة لكن خلال كتابة البرنامج سيتوضح هذا الامر حيث ساقوم بالاعتماد على رقم ال ID وبالتالي سيتم شمل المفهومين ID و Name في الشرح.

الواجهة التي سنقوم بتشكيلها لهذا المقال بسيطة، تتضمن زرین About و Read وتتضمن منطقة نصية للادخال.. و label ولا اعتقد ان هذا الامر بحاجة الى توضيح!

يمكنك البدء باضافة هذه العناصر بواسطة تفعيل العنصر ومن ثم انشاء ابعاده على الواجهة، يجب ان تبدو الواجهة بعد الانتهاء بالشكل التالي



طبعا يمكنك تغيير العديد من الخصائص الخاصة بكل عنصر من خلال نافذة Properties كما في اي لغة برمجة اخرى.

الان وقد انتهينا من تشكيل الواجهة (يمكنك ايجاد ملف ال dlg في الحزمة المرفقة ولاختصار الوقت وتقليل المساحة اللازمة لهذا العمل يمكن الاطلاع على قائمة بخصائص كل عنصر من ذلك الملف بفتحه باي محرر resource).

- ربط العمليات مع بعضها البعض:

الآن نعود الى طرح السؤال السابق بصورة اخرى، كيف يمكننا الان تحديد عمل كل زر او عنصر من خلال كود البرنامج؟

للاجابة على هذا السؤال لنقم بالاطلاع على الكود الموجود في ملف ResDlg.asm للوقوف على حقيقة ما يجري.

```
invoke GetModuleHandle,NULL
mov     hInstance,eax

invoke InitCommonControls
invoke DialogBoxParam,hInstance,IDD_DIALOG1,NULL,addr DlgProc,NULL
invoke ExitProcess,0
```

DlgProc proc hWin:HWND,uMsg:UINT,wParam:WPARAM,lParam:LPARAM

```
mov     eax,uMsg
.if     eax==WM_INITDIALOG

.elseif eax==WM_COMMAND

.elseif eax==WM_CLOSE
    invoke EndDialog,hWin,0
.else
    mov     eax,FALSE
    ret
.endif
mov     eax,TRUE
ret
```

DlgProc endp

سنلاحظ من خلال الكود السابق التالي, عند تنفيذ البرنامج فانه يقوم بتنفيذ اول عدة دوال الى ان يصل الى الدالة DialogBoxParam, هذه الدالة باختصار شديد تقوم بتشكيل واجهة برنامج من خلال ملف واجهة resource.

لاحظ كيف ان البارامتر الثاني وهو lpTemplateName يشير الى اسم ال Dialog الذي قمنا بانشاءه. وان البارامتر الرابع lpDialogFunc يشير الى كتلة DlgProc في الاسفل. لكن ماذا يعني هذا؟

عندما يريد البرنامج تنفيذ هذه الدالة لتشكيل الواجهة فانه يقوم بارسال رسالة WM_INITIALIZE الى كتلة DlgProc والتي هي عبارة عن وظيفة معرفة تقوم بالتحكم وتنفيذ الرسائل الموجهة الى البرنامج من خلال النقر مثلا على زر ما, القيام بالكتابة ضمن مربع نصي.. الخ كل هذه الاحداث تقوم بتفعيل رسالة MSG ترسل الى هذه الوظيفة كي يتم تحليلها وتنفيذها وفقا لما هو مكتوب في الكود.

لاحظ ان هذه الوظيفة تملك اربع متغيرات, hWin (طبعا يمكن تسمية اي اسم اخر) هي التي تقوم بحفظ مقبض واجهة التطبيق عند الانشاء. uMsg متغير يحمل رقم الرسالة المفعلة (او المنشطة), wParam و lParam متغيران مخصصان لحمل اي معلومات ضرورية للرسالة فهناك رسائل تحتاج الى معلومات اضافية يمكن الحصول عليها من خلال هذين المتغيرين... سيتضح هذا بعد قليل.

من الكود السابق نلاحظ انه عندما ينتقل التحكم الى تلك الوظيفة (او الكتلة) فان البرنامج (او المبرمج p:) يقوم بنقل القيمة الخاصة بالرسالة الى المسجل eax. يقوم المبرمج بهذا العمل لتسهيل عملية التحقق من نوع الرسالة المرسله, حيث يتم استخدام كتلة IF ... END IF للتحقق من القيمة الموجودة في eax ومطابقتها مع عدة انواع للرسائل التي من المتوقع ان يقوم البرنامج بتفعيلها والمراد التعامل معها وفق صورة معينة يحددها الكود المكتوب.

في برنامجنا فنحن نعلم ان هناك رسالة واحدة.. او بالاحرى اثنتين نريد ان نتعامل معهم بصورة معينة, هاتين الرسالتين هما WM_CLOSE و WM_COMMAND حيث يتم ارسال الاولى عند النقر على زر معين, ويتم ارسال الثانية عندما يتم النقر على زر x لاجلاق النافذة.

سيقوم برنامجنا بامر بسيط, لربما قد شاهدته لو قرأت مقالات Iczelion, سيقوم الزر Display عند النقر عليه بعرض ما هو موجود في ال textbox في رسالة على الشاشة, اما الزر About فسيقوم باسناد القيمة Copyright AT2000 © 2006 الى label فارغ في الصورة السابقة وسيتم اسناد القيمة " " اي صفر الى نفس ال label بعد مرور 3 ثوان.

يلزمنا لتنفيذ هذا الامر عدة دوال لن نتناول شرحها بالتفصيل في هذا المقال كما فعلنا مع دوال سابقا, يمكن مراجعة مكتبة MSDN لو امتلكتها او السؤال عنها في نفس الموضوع ان كنت تريد المزيد من المعلومات عنها, وبأي حال يمكن الحصول على حزمة كافية مساعدة لفهم هذه الدوال من موقع Iczelion باسم win32api.

الكود سيكون كالتالي وساتناول شرح الاجزاء التي خصص لها هذا المقال بالاصل في نهايته..

```
invoke GetModuleHandle,NULL
mov hInstance,eax

invoke InitCommonControls
invoke DialogBoxParam,hInstance,IDD_DIALOG1,NULL,addr DlgProc,NULL
invoke ExitProcess,0

DlgProc proc hWin:HWND,uMsg:UINT,wParam:WPARAM,lParam:LPARAM

    mov     eax,uMsg
    .if eax==WM_INITDIALOG
        ; nothing goes here for initializing
    .elseif eax==WM_COMMAND
        mov     eax,wParam
        .if eax==IDC_BTN1
            invoke GetDlgItem,hWin,IDC_EDIT1
            ; eax holds the editbox handle
            mov     CtrlHwnd,eax
            invoke GetWindowTextLength,CtrlHwnd
            ; eax holds the length of the text inserted
            cmp     eax,0
```

```

        je          notext
        inc         eax
        ; we use a different value for nMaxCount depending on
how
        ; many char we enter.. we dont want it to be vulnerable
        ; for buffer overflows :p
        invoke      GetDlgItemText,hWin,IDC_EDIT1,addr
szBuffer,eax
        ; now the text is in szBuffer
        invoke      MessageBox,hWin,addr szBuffer,addr
cool,MB_OK
        notext:
        .elseif eax==IDC_BTN2
        invoke      SetTimer,hWin,NULL,3000,NULL
        invoke      SetDlgItemText,hWin,IDC_STC2,addr
copyright
        .endif
        .elseif eax==WM_TIMER
        invoke      SetDlgItemText,hWin,IDC_STC2,addr blank
        .elseif eax==WM_CLOSE
        invoke      EndDialog,hWin,0
        .else
        mov         eax,FALSE
        ret
        .endif
        mov         eax,TRUE
        ret
DlgProc endp

```

الان ما نريد مناقشته هو كيفية تحديد العنصر المسؤول عن ارسال الرسالة, كما ذكرنا اعلاه فاننا نقوم في البداية بتحديد نوع الرسالة المرسله بمقارنتها بمجموعة من الرسائل التي نريد ان نوجه البرنامج لعمل ما وفقا لها. بعد ان نقوم بتحديد الرسالة ولنفرض انها WM_COMMAND فاننا الان بحاجة الى تحديد الزر الذي قام بحث او تفعيل هذه الرسالة, هذا يتم عمله من خلال التحقق من القيمة الموجودة في المتغير wParam والذي سيحمل الـ ID الخاص بذلك الزر!

اذن نقوم بوضع قيمة المتغير في eax مرة اخرى لسهولة التعامل معه, ومن ثم نقوم باجراء مقارنة بين القيمة في eax وبين (اما) اسم العنصر (او) الـ ID الخاص به.

بما معناه, كان بإمكانني استبدال القيمة IDC_BTN1 على سبيل المثال برقم الـ ID وهو 1003 .. ايضا ضمن بقية اجزاء الكود كما في الدالة SetDlgItemText على سبيل المثال, كان بإمكانني استبدال IDC_STC2 بالقيمة 1005 لكن استخدام الرقم ملزم بامر واحد وهو اني بحاجة الى تعريف هذا الرقم على انه الاسم الخاص بالعنصر الفلاني.. انظر الى محتويات الملف ResDlg.inc وسترى الاسطر التالية:

```

.const
IDD_DIALOG1      equ 101
IDC_EDIT1        equ 1002

```

IDC_BTN1	equ 1003
IDC_BTN2	equ 1004
IDC_STC2	equ 1005

كما نعلم فان equ تقوم بتعريف المصدر ولتكن IDD_DIALOG1 على انها ممثلة لاستخدام 101 (الوجهة).. كما لو اردت الاستغناء عن كتابة NULL على سبيل المثال المساوية للصفر فاقوم بكتابة:

NULL	equ 0
------	-------

- اخيرا:

بهذا نكون قد وصلنا الى نهاية المقال, نسال الله ان نكون قد وفقنا في ايصال المعلومة الواضحة الصحيحة.

لفهم هذا المقال بشكل جيد يجب على القارئ ان يقوم بتطبيقه بنفسه خطوة بخطوة مع تنقيح البرنامج في حال وجود اي خطأ للوقوف على سببه والتصحيح.

لم نقم في هذا المقال بشرح كل تفصيله فيه لبحث القارئ على البحث بنفسه عن الاجوبة للسئلة التي خطرت له وذلك لاكتساب خبرة اكثر في البحث عن الاجابة وفي قراءة المزيد من المواضيع اثناء البحث.

- اسئلة؟

<http://www.arabteam2000-forum.com/>
best wishes