Subject: Re: HOW CAN I CALL AN EXTERNAL C ROUTINE FROM IDL ???
Posted by Steve Hartmann on Thu, 29 Jun 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Terenzio Pucci wrote:

> I tried also examples that are contained in the
> "rsi\idl52\external\call_external\c" directory, but they don't work.
> They refer to a file call_library.dll that doesn't exist.
> "Call_library.def" and "Call_library.opt" only exist.
> Then i tried to build a dll, compiling one of the examples, under
> visual c++ 5.0.
>

xxx - Some example code removed - xxx

>
>
> CAN ANYONE HELP ME ???
>
> I NEED AN ILLUSTRATED EXAMPLE.
>
> THANKS

I agree that the documentation on this is not very complete. I spent quite a bit of time trying to get this to work, but now I find it really easy to add new functions when I need them. To help myself and others in our lab, I created a small 'how to' document explaining how to create a DLL using MS Visual C++ that can be called from IDL using Call_External. This works for my Windows NT machine, but I think the procedure for a Mac is very similar.

I hope this helps,
Steve Hartmann

How to create a Windows NT DLL for use with IDL's Call_External:
 -- ---

1. Create a Win32 DLL project using VC++.
2. Create desired member functions for this project.
 a. It must have a DLLMain function.
3. The DLL must also include one of two additional items.
 a. A .DEF file listing the DLL name and export member functions, OR
 b. Member function defined using the _declspec(dllexport) command.
4. Compile and Link the project to create the DLL.
5. Use this DLL with Call_External.

--- -------------

METHOD#1 -- Including a .def file in the project.

Here is an example of a .def file (include this with the project):

```
LIBRARY  FunctionName
DESCRIPTION  'Optional description of function'

CODE   PRELOAD MOVEABLE DISCARDABLE {Optional}
DATA   PRELOAD MOVEABLE SINGLE {Optional}

EXPORTS  MemberFunction1 @1
   MemberFunction2 @2
   MemberFunction3 @3
   MemberFunction4 @4
```

Note: the @number field is optional and denotes the ordinal value.

Member functions are called as in method #2, without the DLLExport keyword.

--- -------------

METHOD#2 -- Define member functions using _declspec(dllexport)

Here is an example of the header file definition of the member functions:

```
//myDLL.h

//Define a pointer to a float type.
#ifndef LPFLOAT
#define LPFLOAT  FLOAT FAR*
#endif

#define DllExport _declspec(dllexport)

//Function prototypes.
BOOL   WINAPI DllMain(HINSTANCE hInst, ULONG ulReason, LPVOID lpReserved);
long   DllExport MemberFunction1(LONG lArgc, LPVOID lpvArgv);
long   DllExport MemberFunction2(LONG lArgc, LPVOID lpvArgv);
int    DllExport MemberFunction3(LONG lArgc, LPVOID lpvArgv);
LPSTR  DllExport MemberFunction4(LONG lArgc, LPVOID lpvArgv);
```

Here are examples of the member functions included in myDLL.c:

1. Example 1

```
LONG DllExport  MemberFunction1(LONG lArgc, LPVOID lpvArgv)
```

```
{
 MessageBox(NULL, (LPSTR)"The call to TestOne() was successful.",
  (LPSTR)"Reply From dlltst32.dll", MB_OK | MB_ICONINFORMATION);
 return(1);
}
 -------------------------------------
```

2. Example 2

```
LONG DllExport MemberFunction2(LONG lArgc, LPVOID lpvArgv)
{
 LPLONG  lplArray = NULL;
 LPLONG* lplplArgv = NULL;
 LONG   lLen = 0;
 LONG   lSum = 0;
 CHAR  szMsg[256];

 lplplArgv = (LPLONG*)lpvArgv;
 lplArray = lplplArgv[0];
 lLen = *lplplArgv[1];

 // Determine the sum of the array elements.
 while (lLen-- > 0) {
  lSum += *lplArray++;
 }

 return(lSum);
}
```

 -- -----------

Both methods require a DllMain function in the DLL, similar to this:

```
#include "myDLL.h"

//Windows DLL entry point.
BOOL WINAPI DllMain(HINSTANCE hInst, ULONG ulReason, LPVOID lpReserved)
{
 return(TRUE);
}
```

 -- -----------
 -- -----------

How to use the DLL in IDL:

1. Call without passing parameters:

```
lResult = CALL_EXTERNAL('myDLL.dll', 'MemberFunction1')
```

2. Call and pass a LONG array by reference, and a LONG scalar by reference.
lResult = CALL_EXTERNAL('myDLL.dll', 'MemberFunction2', LINDGEN(x), xL)