Subject: Re: IDL 5.0 - call_external under Win95/NT
Posted by mark on Thu, 03 Jul 1997 07:00:00 GMT
View Forum Message <> Reply to Message

On Mon, 30 Jun 1997 00:12:50 GMT, Peter Mason
<peterm@demsyd.syd.dem.CSIRO.AU> wrote:

>
> On Sat, 28 Jun 1997, Karl Krieger wrote:
>> Has anyone been succesful in getting the Dlltst32 example
>> code to run under Win95 or NT4.0?
>
> I haven't compiled RSI's, but I have compiled other DLLs for IDL 5.0
> using the MS Visual C++ 2 compiler.
>
>
>> Their example DLL works just fine, but recompiling with
>> VisualC++ 5.0 whith the original function declarations
>> (as "LONG WINAPI") gives a DLL where the respective
>> functions are not exported (i.e. not visible to call_external).
>
> It sounds like you might have left out explicitly specifying the .DEF
> file in your project. You have to add it to your project as a "source
> file", along with the .C file(s), I think.
>
>
>> I had to change the function declarations to
>> "__declspec(dllexport) LONG" to get the functions visible
>> to IDL. However, while the recompiled DLL works fine with
>> IDL4.0.1, IDL5.0 will immediately crash if one starts
>> the dlltst32.pro procedure.
>
> The way IDL calls DLLs has changed in version 5.
> IDL4 could quite happily work with DLLs in which the functions had been
> exported without .DEF files, e.g., by means of "__declspec(dllexport)" in
> MS C++ or "__export __syscall" in Watcom C. In IDL5, you *have to*
> export by means of a .DEF file and define your routines to use the WINAPI
> calling convention. (I think this .DEF file muckaround introduces an extra
> level of indirection in the way the routines are called, or something - some
> arcane Microsoft thing. Whatever the case, IDL5 appears to rely on it.)
>
> I've complained to RSI about this, and they did express some willing to
> sort it out. (Haven't done so yet, though.)
>
>
> Peter Mason

I'm not sure if my previous reply got posted, so, I will repost. The

way that DLLs are called in 5.0 has not changed, the change is due to the fact that WINAPI is defined as __stdcall and __stdcall decorates the name of the functions when exported via __declspec(dllexport). Thus, the function in dlltst32 called TestOne is exported as _TestOne@8. You can verify this by executing the following command; dumpbin -exports dlltst32.dll

Using the .def files causes the decorations to be removed from the name. Here is a Microsoft KBase article ID: Q140485

"There is no _pascal keyword in the 32-bit editions of Visual C++. Instead the Windef.h header file has PASCAL defined as __stdcall. This creates the correct style calling convention for the function (the called function cleans up the stack) but decorates the function name differently. So, when __declspec(dllexport) is used (in a DLL, for example), the decorated name is exported instead of the desired PASCAL style name, which is undecorated and all uppercase.

MORE INFORMATION

PASCAL name decoration is simply the undecorated symbol name in uppercase letters. __stdcall name decoration prefixes the symbol name with an underscore (_) and appends the symbol with an at sign (@) character followed by the number of bytes in the argument list (the required stack space). So, the function when declared as:

```
int  __stdcall func (int a, double b)
```

is decorated as:

```
_func@12
```

The C calling convention (__cdecl) decorates the name as _func. Whereas the desired PASCAL style name is FUNC.

To get the decorated name set the Generate Mapfile option in the Linker General category setting.

Use of __declspec(dllexport) does the following:

function being exported does not use the C calling convention (for example, __stdcall), it exports the decorated name.

So to simulate PASCAL name decoration and calling conventions, you must have the "Called Function stack clean-up" provided by using __stdcall and the undecorated uppercase name.

Because there is no way to override who does the stack clean up, you must use __stdcall. To undecorate names with __stdcall, you must specify them by using aliases in the EXPORTS section of the .def file. This is shown below for the following function declaration:

```
int  __stdcall MyFunc (int a, double b);
void __stdcall InitCode (void);
```

In the .def file:

```
EXPORTS
  MYFUNC=_MyFunc@12
  INITCODE=_InitCode@0 "
```

==========
Mark E. Leaming
mark@psec.com
==========