Subject: C++ and CALL_EXTERNAL
Posted by mark.t.douglas on Thu, 29 May 2008 09:40:01 GMT
View Forum Message <> Reply to Message

After an entire evening wasted trying to get IDL to interface with a DLL I made, I thought I'd jot down the things that I wish I had known at the beginning, in the hope that it will be useful to someone, somewhere, sometime. I was using IDL 6.1 on Windows and Microsoft's Visual C++ Express 2008 compiler; the same procedure will work on other OSes, mutatis mutadis.

OK, here we go. Suppose we have two functions, written in C++, that we wish to use from within IDL. We naively start with the following header:

```
#ifndef NORMALS_H
#define NORMALS_H

namespace Normals
{
 __declspec(dllexport) double InverseCumulative(double x);
 __declspec(dllexport) double Cumulative(double x);
}

#endif
```

After building the DLL and moving it to IDL's working directory, we type the following into IDL:

```
x = call_external("MyLib.dll","Cumulative",double(0.5),/all_value,/
d_value,value=[0])
```

It can't find the function! Why? Because the polymorphism and overloading features of C++ are usually implemented by mangling your nice function names into something that looks like a core dump. Examine your DLL with a program like PEDUMP to figure out what Normals::Cumulative() is now known as; I get ? Cumulative@Normals@@YANN@Z . That line noise encodes precise information about the argument types accepted by the function, believe it or not. Armed with this information, we type the following into IDL:

```
x = call_external("MyLib.dll","?
Cumulative@Normals@@YANN@Z",double(0.5),/all_value,/d_value,value=[0])
```

Alarmingly IDL crashes! Why? Well, because IDL assumes all functions follow the prototype

RETURN_TYPE function_name (int argc, void *argv[])

so Normals::Cumulative() received a totally different argument list to what it was expecting. So different that it went haywire and took down IDL. To overcome this difficulty we have to provide "glue" functions that take IDL's argument list and mangle it appropriately for our functions. It is important to know how IDL's arg list works. The int argc bit is easy enough; it tells you how many arguments you have been passed. The array of pointers may not actually be an array of pointers, however: if you set the /all_values keyword in call_external, or set some elements of its values keyword to non-zero numbers, then the appropriate arguments will be passed by value, and the appropriate elements of argv[] will not be pointers but values!

Now before we were trying to pass a double by value. However a double is (on this computer at least) bigger than a pointer, and so we can't pass that argument by value, because doing so would corrupt some following pointers. So we must pass the double by reference. We are now ready to proceed.

Our glue functions have the prototypes

```
__declspec(dllexport) double __cdecl InverseCumulative_IDL(int argc,
void *argv[]);
__declspec(dllexport) double __cdecl Cumulative_IDL(int argc, void
*argv[]);
```

within the Normals namespace (the __cdecl bit would allow you to use the more powerful LINKIMAGE IDL procedure instead of call_external). Their implementation is

```
__declspec(dllexport) double __cdecl Cumulative_IDL(int argc, void
*argv[])
{
 double *ptr  =  (double*) argv[0];
 return Cumulative(*ptr);
}

__declspec(dllexport) double __cdecl InverseCumulative_IDL(int argc,
void *argv[])
{
 double *ptr = (double*) argv[0];
 return InverseCumulative(*ptr);
}
```

and their mangled names are (from PEDUMP)
?InverseCumulative_IDL@Normals@@YANHQAPAX@Z
?Cumulative_IDL@Normals@@YANHQAPAX@Z

Now call_external has an "auto_glue" faclity that can build the glue functions for you; it is instructive to write them yourself, at least at first, I feel. Also note that our glue functions are ferociously fragile; passing anything other than a single double by reference will cause problems. This isn't so much of a problem if you clearly document the fact that the glue function must be called with precisely correct arguments. Nonetheless we are now in a position to actually use our DLL from within IDL. After building the DLL with the glue functions and moving it to where IDL can see it, we can type e.g.
x = call_external("MyLib.dll","?Cumulative_IDL@Normals@@YANHQAPAX@Z",double(0.3),/d_value,/cdecl,values=[0])

and everything works.
