Subject: Re: dynamic memory in dll
Posted by Randall Skelton on Mon, 03 Jun 2002 22:02:48 GMT
View Forum Message <> Reply to Message

I am hardly an expert, but I'll take a stab at answering this one...

(1) Direct access to pointer and object reference heap variables (types IDL_TYP_PTR and IDL_TYP_OBJREF, respectively) is not allowed as of IDL 5.5. I have asked for this interface in the past and hope that by the release of IDL 6, we will all be able to encapsulate our data and return IDL objects from C/C++ ;)

(2) If you can write call_external functions then you can do DLMs. I strongly suggest you buy Ronn Kling's book as he lays out the interface and gives example code that will have you passing any non-heap variable between IDL and C in 2 days or less!

(3) If you must use call_external read the External Development Guide (EDG) under chapter 9 and read the protos given in external.h. However, given point number 1, I doubt you will be able to do what you are suggesting. As I have recently learned, IDL is constantly managing memory so working and expecting to be able to resize variables from C/C++ under these conditions is unrealistic. Allocating a heap variable with 'ALLOCATE_HEAP' in IDL is really just giving you a handle to a heap variable with an undefined IDL type. This variable is not actually dynamically sizable in the way you imagine, i.e. to re-size the heap variable, you must rely on IDL:

```
IDL> .reset
IDL> a = ptr_new(/Allocate_heap) & help, a & help, /heap
IDL> *a = findgen(100)
IDL> *a = [*a, findgen(100)
```

This is not really any different than simply writing:

```
IDL> .reset
IDL> a = a & help a
IDL> a = findgen(100)
IDL> a = [a, findgen(100)]
```

except that you are using pointer notation to refer to the IDL variable in the first case. At the end of the day, however, the performance will the same because in each case you are relying on the IDL interpreter to reallocate memory for the given IDL variable.

> call external (name, function, a, b, c,d ,...)
> a now contains something (not fixed size)

I am not entirely sure I understand what you mean here by 'not fixed size?' or why you need/want such a thing (feel free to post an example). You most certainly can create any non-heap variable type from within C and return it to IDL. This means IDL short, long, float, double, complex, dcomplex, string, scalars and/or arrays can be returned. Likewise, with creating and returning IDL structures or arrays of structures. The interfaces for all these types are given in the EDG.

Cheers,
Randall
