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

Subject: Re: dynamic memory in dll
Posted by ronn on Tue, 04 Jun 2002 01:04:44 GMT

View Forum Message <> Reply to Message

in article 3cfb8997.0@news.ruca.ua.ac.be, Gert Van de Wouwer at Gert.VandeWouwer@NOSPAMua.ac.be wrote on 6/3/02 11:21 AM:

> Hi,
>
> I want to use C++ code through a dll with call_external. Is it possible to
> allocate IDL memory for a variable that is alive after the dll unloads? I
> mean:
>

Hello Gert,

Do you really want to pass in a pointer or do you just want to create an array in the dll and pass it back to IDL? i.e.

;a does not exist yet
call external (name, function, a, b, c,d ,...)
a now contains something (not fixed size)

I avoid call_external at all costs. It is so much easier to write your own dlm/dll even if you have to call another dll. If you choose this route then you want to use the IDL_MakeTempArray in your dll and pass back the IDL_VPTR it creates. That way you can use it in your IDL code just like a normal variable.

-Ronn

--
Ronn Kling
KRS, inc.
email: ronn@rlkling.com
"Application Development with IDL"ï¿½ programming book updated for IDL5.5!

Subject: Re: dynamic memory in dll
Posted by Gert on Wed, 05 Jun 2002 20:46:33 GMT
View Forum Message <> Reply to Message

Thanks for the input Randall and Ronn.

Understandebly, noone's a fan of call_external. I used it for my previous project, but now I think I'll use linkimage. DLM's are next - i'm moving up... (I'll have to get Ronn's book).
Unfortunately, I have to get something finished in 2 weeks - no time to investigate too much further.

Now, could you comment on the code i put below?

This code seems to work with

```
a=float(indgen(5))
b=testdlm1(a)
print, b
```

I have 2 questions
1) Can I change the dimension a in the C-code
2) how can I use IDL_cvtflt if a is an int array? if i use float* pflTemp = (float*)IDL_CvtFlt(1, idlIn1) then pflTemp apparantly is a new temporary variable that has nothing to do with a - a is thus unchanged when returning to IDL.

...or am i just asking to much here?

thanks for the replies,

Gert

C-Code

```
IDL_VPTR TestDLM1(int argc, IDL_VPTR argv[])
{
int i;

//get first input
IDL_VPTR idlIn1 = argv[0];
IDL_ENSURE_SIMPLE(idlIn1);
IDL_ENSURE_ARRAY(idlIn1);
if (idlIn1->type != IDL_TYP_FLOAT)
   IDL_Message(IDL_M_GENERIC, IDL_MSG_LONGJMP, "Error! float array
expected");

float* pflIn = (float *)idlIn1->value.arr->data;
int dN = idlIn1->value.arr->n_elts;

for(i=0;i<dN;i++) pflIn[i] = pflIn[i] * (float)2.0;


//make a return vector
IDL_VPTR dst;
float* pflTemp =
 (float*)IDL_MakeTempVector(IDL_TYP_FLOAT,10,IDL_ARR_INI_INDE X, &dst);
pflTemp[0]=100.0;
pflTemp[3]=200.0;

return(dst);
}
```

*****end of code **********

"ronn kling" <ronn@rlkling.com> wrote in message
news:B9218AAC.545B%ronn@rlkling.com...
> in article 3cfb8997.0@news.ruca.ua.ac.be, Gert Van de Wouwer at
> Gert.VandeWouwer@NOSPAMua.ac.be wrote on 6/3/02 11:21 AM:
>
>> Hi,
>>
>> I want to use C++ code through a dll with call_external. Is it possible
to
>> allocate IDL memory for a variable that is alive after the dll unloads?
I
>> mean:
>>
> Hello Gert,
>
> Do you really want to pass in a pointer or do you just want to create an
> array in the dll and pass it back to IDL? i.e.
>
> ;a does not exist yet
> call external (name, function, a, b, c,d ,...)
> a now contains something (not fixed size)

>
> I avoid call_external at all costs. It is so much easier to write your
own
> dlm/dll even if you have to call another dll. If you choose this route
then
> you want to use the IDL_MakeTempArray in your dll and pass back the
IDL_VPTR
> it creates. That way you can use it in your IDL code just like a normal
> variable.
>
> -Ronn
>
>
> --
> Ronn Kling
> KRS, inc.
> email: ronn@rlkling.com
> "Application Development with IDL" programming book updated for IDL5.5!
> "Calling C from IDL, Using DLM's to extend your IDL code"
> http://www.rlkling.com/
>
>
>