Subject: Re: Object Data and pointer assignments
Posted by davidf on Thu, 09 Mar 2000 08:00:00 GMT
View Forum Message <> Reply to Message

Ben Tupper (tupper@seadas.bigelow.org) writes:

> I am in the middle of wrtting my first object from scratch. Scratch is
> a good word since I'm doing a lot of that on my head. I'm hoping to get
> some advice on organization of data. I need 4 pieces of data (one 2d
> arrays and two structures that vary in size according to the size of the
> arrays) plus six keywords that I need to get/set. Currently, I have
> defined each of the 3 bits of data as null pointers in the BLAH__DEFINE
> procedure.
>
> In the BLAH::INIT function, the user passes one of the two arrays as an
> argument. At that point I reassign one of the pointers to...
>
>   Self.InArray = Ptr_New(InArray).
>
> I think I understand why I can reassign the structure field when going
> from a null pointer to a filled pointer. On second thought, I don't
> understand it but I can accept that it works. It's the next step I need
> help on.

The reason you need to use an actual pointer (Ptr_New) here,
is that you *don't* have a pointer from the BLAH__DEFINE
module. What you have done in that module is said that the
definition of the InArray field *will be* a pointer. In other
words, the BLAH__DEFINE module only *defines* the object and
its fields, it doesn't assign anything to the self object. This
is what must be done by the INIT method.

> I would like to change the contents of this field later to some other
> value (a differently sized array.) Here's where the ice under me gets
> very very thin and my eyes get misty. In the BLAH::SETPROPERTY method,
> I don't know if I should free this pointer before reassigning (and does
> that leave the structure field undefined?), or if I should simply
> overwrite it as I did in the INIT function. If I reassign the filed
> to a new pointer, what happens to the previously occupied heap space?
> Have I sprung a leak?

To reassign the pointer to something else (after it has been
defined by the INIT method), you simple de-reference the pointer:

```
  *self.InArray = newStruct
```

You don't leak any memory because IDL is managing this
whole process for you. (Remember, these pointers are

not real pointers in the C sense. They are really
glorified variables in the IDL sense.) This is the
bestest feature of IDL pointers. :-)

If you overwrite the pointer like this:

```
   self.InArray = Ptr_New(newStruct)
```

you *will* leak memory because now you destroyed the
only reference to that pointer area of memory. You could
do this:

```
   Ptr_Free, self.InArray
   self.InArray = Ptr_New(newStruct)
```

But what is the point, if IDL can do it all for you?

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155