Subject: Re: Object Data and pointer assignments
Posted by John-David T. Smith on Thu, 09 Mar 2000 08:00:00 GMT
View Forum Message <> Reply to Message

Ben Tupper wrote:
>
> David Fanning wrote:
>
>> 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. :-)
>>
>
> Thanks for the tips. It's probably a good thing that I don't know much about
> C (no bad habits, eh?)
>
>>
>> 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:
>>
>>
>
> So, if I am following your instruction correctly, I should only see ...
>
> self.InArray = Ptr_New(newStruct)
>
> once in my code in the INIT function. Thereafter (in SetProperty for
> example) it is simply derefence....
>
> *self.inarray = newStruct
>

Just to be clear... you are free to free self.inarray, and point it somewhere else, at any time. This can be useful if you have a list which is either empty (NULL pointer a.k.a. a dangling reference), or not (pointer to a list of finite size). If the list changes size, and becomes empty again, you can simply free it, which indicates its emptiness. If it then grows again, simply use ptr_new() to get another heap variable for it. So, while it might be easiest in some cases only to call ptr_new() once, in other cases it is useful to let a single member variable like self.inarray point to different heap variables over its life.

Good Luck,

JD

--

J.D. Smith |*| WORK: (607) 255-5842
Cornell University Dept. of Astronomy |*| (607) 255-6263
304 Space Sciences Bldg. |*| FAX: (607) 255-5875
Ithaca, NY 14853 |*|