Subject: Re: Proper pointer cleanup question
Posted by Paul Van Delst[1] on Tue, 08 Apr 2003 18:10:54 GMT
View Forum Message <> Reply to Message

"M. Katz" wrote:
>
>>>> 2) a and all of its dependent pointers:
>>>> for i=0,n_elements((*a).p)-1 do $
>>>> ptr_free, ((*a).p)(i)
>>>> ptr_free, a
>>>
>>> #2 is the go. All the others leave you with dangling references and memory leaks. My
>>> personal mantra is that when it comes to pointers, be very explicit in their garbage
>>> collection i.e. don't assume freeing a pointer also frees any "child" pointers like the
>>> components "p" in your example. (I actually don't know of any languages that *do* do that,
>>> but I'm barely bilingual. :o)
>>
>
> Thanks! I'll write myself a full reverse-ordered cleanup routine.
> I suppose this explicit cleanup is just as important for objects as
> well:
> Object pointer fields should be explicitly freed in the Cleanup
> method.
>
> Question 1) But what about simple scalar pointers?
> a = ptr_new(fltarr(10,10))
>
> If I set
> a = 0
> Will I have stranded my fltarr(),

Yes

> or is IDL smart enough to deallocate
> it properly?

No. To quote the IDL help, you can "reclaim" lost heap variables using the CAST keyword to PTR_VALID() but I wouldn't recommend using that on a regular basis (I've used it on the command line after fat-fingering, but not in a routine)

> Question 2) Then how about this scenario
> a = ptr_new(fltarr(10,10))
> b = ptr_new(dblarr(5,5))
>
> a = b ;--- Does this strand the original a array?
> *a = *b ;--- How about this?

The first one would (I think....I don't like to see pointers used with "=" signs. It

confuses me. I prefer the Fortran operator => or the C one ->). The second one, I'm not sure. In either case, I would explicitly destroy what I do not need anymore before reusing the variable/pointer name.

paulv

--
Paul van Delst
CIMSS @ NOAA/NCEP/EMC
Ph: (301)763-8000 x7748
Fax:(301)763-8545
