Subject: Re: pointers--avoiding a memory leak
Posted by David Fanning on Wed, 19 May 2004 19:34:57 GMT
View Forum Message <> Reply to Message

M. Katz writes:

> Here's a simple Pointers 101 question for the pointer gurus.
>
> Suppose you have a structure with a pointer field
> s = {a:10, b:ptr_new(10)}
>
> Somewhere down the line you want to update the value of *s.b making it
> equal to the value contained in a another pointer, say *q = 20. After
> the assignment, you'll no longer need the q pointer.
>
> So which is a better strategy?
>
> #1)
> ptr_free, s.b
> s.b = q
>
> #2)
> *s.b = *q
> ptr_free, q
>
> #3)
> s.b = q ;--- what becomes of the old s.b in this case?
>
> I can see how #1 is memory-efficient because only the pointer is
> passed. I can see that #2 is memory inefficient because the values are
> swapped. This could be slower if the value is a large array. I can see
> how #3 might result in a memory leak, since the old s.b value could be
> stranded in memory with no pointer pointing to it. Am I right about
> these? What else should I be thinking about in the above situation?

I think you pretty much understand the situation. You definitely
leak memory in #3. I quibble a little bit with you conclusion that
#2 is memory inefficient, since I think internally C pointers are
moving around, not actual data. But other than that, I think you
can start handling the pointer guru questions from now on. :-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/