Subject: Re: Structure field concatenation
Posted by Martin Schultz on Thu, 07 Sep 2000 07:46:22 GMT
View Forum Message <> Reply to Message

Now, if an american is getting pedantic on this issue, I would really like to be thorough here ;-)

Let's state the "problem" again:
We already have a pointer variable, and we want to reassign a new value to it. But, as typical for "real life" situations, we don't know for sure if the pointer that we have is valid or not.

This leads to two possible solutions:

dwf:
```
  IF NOT Ptr_Valid(myptr) THEN Ptr_New(myptr)
  *myptr = thingy
```

mgs:
```
  IF Ptr_Valid(myptr) THEN Ptr_Free, myptr
  myptr = Ptr_New(thingy, /No_Copy)
```

Both solutions should lead to the same result and with approximately the same speed (at least if I add the No_Copy keyword to my solution). So, the difference is that thingy is still accessible in David's solution while it becomes undefined in mine (and David's solution saves a couple of keystrokes which you can then use for documentation).

Well, if there were no other hidden differences, I would declare victory to David. BUT, there is in fact a difference!! If you want to point to an undefined variable (well, who wants to do this anyway), David's solution breaks, whereas my way happily creates a new pointer pointing to an undefined variable. Thus, the fail-safe dwf solution would be:

```
  IF NOT Ptr_Valid(myptr) THEN Ptr_New(myptr)
  IF N_Elements(thingy) GT 0 THEN *myptr = thingy $
  ELSE ...  ; either stop, print a warning, create a new pointer, or
...
```

And this gives me leeway for more documentation ;-)

Cheers,
Martin

"J.D. Smith" wrote:

>
> David Fanning wrote:
>>
>> Martin Schultz (martin.schultz@dkrz.de) writes:
>>
>>> The only reason to do this that I could accept without further
>>> quirking is if you tell me there is a lot of penalty if you manually
>>> deallocate and reallocate the memory instead of letting IDL do it.
>>> Haven't tested, but I would doubt that it makes a big difference.
>>
>> I really don't think there is any difference at all.
>> If it makes you feel safer, by all means free pointers
>> yourself. I just wanted to make the point once again that
>> IDL really does have some nice features. This aspect
>> of pointer memory management is one of them. :-)
>
> I just wanted to point out that this technically isn't pointer memory management
> at all. Rather, it's simply the same old variable memory management we know and
> love:
>
> IDL> a=
> IDL> a=1
>
> ...no memory loss there! The only difference is that *heap* variables are being
> handled in the pointer case. So Martin, if you're happy with this, you should
> be happy with David's method. Of course you might always use "delvar,a", but
> somehow I doubt it.
>
> I'm being pedantic only to prevent readers (especially the Java-enabled among
> them) getting confused about what kind of memory management IDL really
> provides. The best way to stay clear on the issue is to think about pointers as
> what they are: references to IDL variables which are exactly the same as any
> other variable except for being hidden ("on the heap") and persistent ("not
> cleaned up by function/procedure exits"). They do *not* point directly to
> memory (just as variables like "a" above don't directly map to memory --
> thankfully for us).
>
> JD
>
> --
> J.D. Smith /*\ WORK: (607) 255-6263
> Cornell University Dept. of Astronomy */ (607) 255-5842
> 304 Space Sciences Bldg. /*\ FAX: (607) 255-5875
> Ithaca, NY 14853 */

--
[[[[[[[[[
[[Dr. Martin Schultz Max-Planck-Institut fuer Meteorologie [[

[[Bundesstr. 55, 20146 Hamburg [[
[[phone: +49 40 41173-308 [[
[[fax: +49 40 41173-298 [[
[[martin.schultz@dkrz.de [[
[[[[[[[[[[[[[[[[