Subject: Re: When you can't concatenate structures like you expect....
Posted by J.D. Smith on Mon, 24 May 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Paul van Delst wrote:
>
> Kenneth P. Bowman wrote:
>
>> In article <3745B3BB.815FDE87@ssec.wisc.edu>, Paul van Delst
>> <paul.vandelst@ssec.wisc.edu> wrote:
>>
>> As Pavel pointed out in another post, anonymous structures have a "hidden"
>> name that must match in order to be able to concatenate. The trick is to
>> copy your original structure and update the fields, rather than creating a
>> new structure.
>
> Yep I know - I explained this and provided an example in my original post.
>
> I guess what I really trying to say in my long-winded, circuitous way is that
> I do not understand why IDL treats structures differently from any other data
> type. I understand why one would have a restriction on changing the
> definition of the various "tags" within a structure but other than that, why
> should structures or arrays of structures be any different from other
> variable types or arrays? From my point of view, it appears to be an
> implementation issue within IDL - why should I care what the "hidden"
> structure name is? If I cared about structure names I'd used a named
> structure, right?
>
> As Liam Gumley sagely pointed out to me this morning, I am really asking the
> wrong question. Instead of asking "why doesn't this work like I expect it
> should?" the question should be "what do I need to do to get the job done?"

Structures are treated differently because they are different! There
are no straitforward a priori rules which guarantee two structures are
compatible. Contrast this to arrays, for example, where size and type
determine compatibility quite simply.

Internal names are needed because otherwise I'd be able to do:

IDL> a=[{foo:5},{foo:'a'}]

... unless, of course, IDL was willing to traverse each structure in
full and check all tag/type pairs to ensure a match, which seems to be
what you're proposing. This is very different from the simple type
checking done for all the other variable types. And then what happens
if I say:

IDL> a={foo:{foo:1}}

Which is correct? The outer or the inner, or both? Or suppose I'd really like to do:

IDL> a={foo:1} & b={foo:'testing'}

This simple functionality permitted by anonymous structures would not be allowed.

Basically, what you're advocating is that *all* structures be named structures, where the "name" is not specified by the user, but instead based upon the tag names plus the full recursive type info, to be gathered either at the time of creation (full-fledged named structs), or each time a concatenation is performed (special case named structs). This is not impossible. Presumably it could be done with a suitably chosen hash function. However, it eliminates the benefits of anonymous structures, and makes structure concatenation stricter and slower.

My suggestion is to bite the bullet and use a named struct, or stick to your relaxed assignment.

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 |*|