Subject: Re: When you can't concatenate structures like you expect....
Posted by Pavel Romashkin on Fri, 21 May 1999 07:00:00 GMT

What you describe is exactly what you expect from the way IDL creates
structures. It says in the manual that every named structure is unique,
and this is also valid for anonimous structures, which are different from
named in only that IDL names them, not you. But if you create two
anonimous structures with same fields, they will have different internal
names and will not be identical, whether this is for good or bad.
I found this feature too and it seems that nice workarounds can be found to
accomplish what you want without needing this sort of concatenation. If
you think about it, you are going to store *exactly* identical data in
exactly the same fields... then what is the point of creating two
different structures, unless you do want them to be different somehow?
Cheers,
Pavel

Subject: Re: When you can't concatenate structures like you expect....
Posted by bowman on Sat, 22 May 1999 07:00:00 GMT

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.

That is:

IDL> a = {i: 0L, j:0L}
IDL> b = {i: 1L, j:1L}
IDL> a = [a, b]
% Conflicting data structures: B,concatenation.
% Execution halted at: $MAIN$

won't work, because

IDL> HELP, a, b, /STRUCT
** Structure <1007dc08>, 2 tags, length=8, refs=1:
   I   LONG   0
   J   LONG   0
** Structure <1007da08>, 2 tags, length=8, refs=1:
   I   LONG   1
   J   LONG   1

shows that the "names" (in angle brackets) are different.

but

```
IDL> a = {i: 0L, j:0L}
IDL> b = a
IDL> b.i = 1L
IDL> b.j = 1L
IDL> a = [a, b]
IDL> HELP, a, b, /STRUCT
** Structure <1007d908>, 2 tags, length=8, refs=2:
   I            LONG            0
   J            LONG            0
** Structure <1007d908>, 2 tags, length=8, refs=2:
   I            LONG            1
   J            LONG            1
```

does work.

Thanks to David Fanning for this one (and many others).

Ken

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 |*|

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

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?"

cheers,

paulv

--
Paul van Delst
Space Science and Engineering Center | Ph/Fax: (608) 265-5357, 262-5974
University of Wisconsin-Madison | Email: paul.vandelst@ssec.wisc.edu
1225 W. Dayton St., Madison WI 53706 | Web: http://airs2.ssec.wisc.edu/~paulv