Subject: Re: Arrays in structures; workarounds?
Posted by John-David T. Smith on Thu, 04 May 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Craig Markwardt wrote:
>
> Ed Santiago <esm@lanl.gov> writes:
>
>>> Does anybody know a way to work around this? [...] The
>>> only thing I can come up with is to parse the result of HELP,
>>> OUTPUT=out, but that seems like the crappiest solution ever.
>>
>> Yep. Crappy indeed... but I couldn't find an alternative, either.
>>
>> Below is a copy of "esmsize", a function I wrote last year when I
>> *absolutely needed* to obtain true dimensions within a struct.
>>
>> Hope it helps,
> ...
>
> Yes, that's what I was afraid I might have to do. Unfortunately I'd
> still like it to work for IDL 4, for which HELP, OUTPUT doesn't work.
>
> Here was one trick I found to determine the size of a structure tag,
> if it has *at least* two elements. Try this:

```
>    IDL> zz = {x:reform(dblarr(2,2,1),2,2,1)}
>    IDL> help, zz([0,0]).x
>    <Expression>    DOUBLE    = Array[2, 2, 1, 2]
```

>
> In this case, I index the structure with the [0,0] list while at the
> same time extracting the X tag. You get the correct dimensions, with
> an extra "2" tacked on the end, which you can then hack off. You need
> to handle the case of X being 8 dimensional (!), but it works.
> Unfortunately this *doesn't* work if X has only element. Arghh!
>
> A comment on your procedure. I believe that you are treating the
> output of HELP too simply. When tag names are long enough, help will
> wrap the type description to the next line. Consider this:

```
>    IDL> zz = {sdlfkjsdlkfjsdklfjslkfjsldkfjsdlkfjsljfsldkfjsdf:1}
>    IDL> help, /struct, zz
>    ** Structure <40045788>, 1 tags, length=2, refs=1:
>       SDLFKJSDLKFJSDKLFJSLKFJSLDKFJSDLKFJSLJFSLDKFJSDF
>                     INT       =        1
```

>
> This will affect both the first tag, and any of the following ones.
> Maybe it's better to do a search for the tag you want.

The problem here is reliance on *trailing* shallow dimensions. The IDL manual quotes us:

"As with other subscript operations, trailing degenerate dimensions (those with a size of 1) are eliminated."

While I can't agree with IDL's mixed notion of a variable's dimensionality between help and direct structure member access, I think IDL has always been clear about this point.

e.g.

```
IDL> x=reform(dblarr(2,2,1),2,2,1)
IDL> print,size(x)
           3           2           2           1           5           4
IDL> y=x
IDL> print,size(y)
           2           2           2           5           4
```

Whenever it gets a chance to, IDL snips off those trailing dimensions, not just on structure access. You may disagree with the policy, which was designed to accomodate image subscripting without too much dimensional fussing, but it is documented. Living with this behavior when you'd like to preserve then number of dimensions usually involves ensuring that subscripting extraction happens on leading dimensions. Making use of this behavior usually involves ensuring it happens on the trailing dimensions.

The one place you are justified in complaining is the truncation of a single element vector to a scalar: this is a bug (or at least an inconsistency), and affects structure field access only:

```
IDL> a=[1]
IDL> print,size(a)
           1           1           2           1
IDL> b=a
IDL> print,size(b)
           1           1           2           1
IDL> b={a:a}
IDL> help,b,/st
** Structure <823abb4>, 1 tags, length=2, refs=1:
   A               INT       Array[1]
IDL> help,b.a
<Expression>    INT       =        1
```

The statement from the documentation quoted above is wrong. The is one time a final unit length dimension is not degenerate --- for a vector of length 1.

Good Luck,

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 |*|