Subject: Re: Arrays of Structures
Posted by JD Smith on Fri, 09 Feb 2007 20:51:50 GMT
View Forum Message <> Reply to Message

On Fri, 09 Feb 2007 02:13:56 -0800, Mick Brooks wrote:

> If I try, I get the following:
>
> IDL> struct=replicate({a:lindgen(100,100,100)},100)
> IDL> print,(m=memory(/HIGHWATER))/1024/1024.,"MB"
> 385.836MB
> IDL> val=struct[10].a[4]
> IDL> print,((n=memory(/HIGHWATER))-m)/1024/1024.,"MB Extra"
> -3.81445MB Extra
> IDL> val3=struct.a[10,4]
> IDL> print,(memory(/HIGHWATER)-n)/1024/1024.,"MB Extra"
> 0.00000MB Extra
> IDL> HELP, val, val3
> VAL LONG = 4
> VAL3 LONG = Array[100]
> IDL> PRINT, val3
> 410 [+ another 99 of the same]
>
> The problem case doesn't use any extra memory (great!), but it gives a
> different result (boo!).

I probably should have used a better example, that was confusing. The reason val3 "works" is because "a", the field being de-referenced, already has 3 dimensions. So it grabs the [10,4] element of all a's, and then proceeds to thread that across all 100 structures in the structure array, creating a new array in the process. Here's a better example illustrating this issue:

```
IDL> val4=(struct.a)[10,4,1,1]
IDL> val5=struct[10].a[4,1,1]
IDL> print,val4,val5
      10410       10104
IDL> val6=struct.a[10,4,1,1]
% Subscript range values of the form low:high must be >= 0, < size, with low
  <= high: <No name>.
```

Here's the real issue: since "struct.a" doesn't exist as an array anywhere in memory, but instead is a composite entity, it must be formed anew by:

a) allocating a new array of enough memory to hold all of
   n_elements(struct) x size(a,/dimensions) values.
b) going through each structure in the structure array, and copying its

copy of "a" into the new array.

I agree the precedence table is misleading on this point. I do think there is a reasonable argument that IDL should understand struct.a[10,4,1,1] and not need to first create a 325MB array to de-reference it. Here's an easy rule of thumb though: in order to avoid the "new array creation" process described above, just attach all indices as close as possible to the quantity they are indexing. "10" goes with struct (we want the 10th struct). [4,1,1] goes with "a" (we want element [4,1,1] of a).

JD
