Subject: Re: Arrays of Structures
Posted by Paul Van Delst[1] on Fri, 09 Feb 2007 16:01:24 GMT
View Forum Message <> Reply to Message

JD Smith wrote:
> On Thu, 08 Feb 2007 14:02:08 -0500, Paul van Delst wrote:
>
>> Mick Brooks wrote:
>
>> You know, I'm confused now too. Check out the precedence in the IDL help:
>
>> So, you see that the the structure field dereference operator, ".", and the array
>> subscript operator, "[]" , have the same precedence. Operators with equal precedence are
>> evaluated from left to right.
>>
>> So,
>>
>> structs.a[23]
>>
>> means FIRST dereference the structure field (structs.a), THEN index the array ([23]). The
>> way I see it,
>>
>> structs.a[23]
>> and
>> (structs.a)[23]
>>
>> should be equivalent.
>
> The problem is that structs.a is not an array until the ".a" part has been
> applied. What if structs had been;
>
> structs={a:indgen(25)}
>
> what should structs.a[23] do then, and how should IDL know the difference.
> Until IDL knows what shape "structs.a" will have, it cannot make any
> informed decision about how to index it.

How is it different from:

```
IDL> x=findgen(10,20)
IDL> help, x
X               FLOAT     = Array[10, 20]
IDL> help, x[23]
<Expression>    FLOAT     =       23.0000
IDL> help, (x)[23]
<Expression>    FLOAT     =       23.0000
```

?

Or
IDL> x=findgen(10,20,30)
IDL> help,x[1000]
<Expression>   FLOAT   =   1000.00
IDL> help,x[0,0,5]
<Expression>   FLOAT   =   1000.00

?

The reference [23] is being applied to an array. The rank of that array doesn't matter since IDL allows you to reference multi-rank arrays with a single index (treating it as a "flat" array).

If we have

IDL> structs=replicate({a:indgen(25)},10)
IDL> help, structs
STRUCTS   STRUCT   = -> <Anonymous> Array[10]
IDL> help, structs.a
<Expression>   INT   = Array[25, 10]

Then, via the precedence rules,

IDL> help, structs.a[23]

should be equivalent to

IDL> help, structs[0].a[23]

> You might object, saying that
> IDL should just evaluate that to begin with always, but think how
> expensive that is. Creating "structs.a" would cause a large temporary
> array to be created, only to finally index a single element.

Yes, I agree that is bad but it is an implementation detail. The precedence rules seem quite clear that this should work. Please correct me if my interpretation of the rules is wrong.

Maybe it was disallowed in general so they (RSI) wouldn't have to special case structures containing pointers,

IDL> structs=replicate({a:ptr_new(/allocate_heap)},10)
IDL> *structs[0].a=findgen(20)
IDL> *structs[1].a=indgen(74)
IDL> *structs[2].a=dindgen(3)
IDL> *structs[4].a=sindgen(13)
IDL> help, *structs.a[4]

% Subscript range values of the form low:high must be >= 0, < size, with low <= high: <No name>.
% Execution halted at: $MAIN$

Although, again, the rules indicate that the pointer dereference operator "*" has a lower precedence that either "." or "[]" so I would contend that even the above

IDL> help, *structs.a[4]
% Subscript range values of the form low:high must be >= 0, < size, with low <= high: <No name>.
% Execution halted at: $MAIN$

is valid and that parentheses should not be required,

IDL> help, *(structs.a)[4]
<PtrHeapVar1>   STRING   = Array[13]

I also think the case of

IDL> help, *structs.a[23]

is also well defined.... it's invalid (at least it is until IDL allows arrays composed of different different objects).

I reckon these sorts of special cases are why the more general case is disallowed. (But what the hell do I know! :o)

> Compare the
> memory usage of the following:
>
> IDL> struct=replicate({a:lindgen(100,100,100)},100)
> IDL> print,(m=memory(/HIGHWATER))/1024/1024.,"MB"
>   382.472MB
> IDL> val=struct[10].a[4]
> IDL> print,(memory(/HIGHWATER)-m)/1024/1024.,"MB extra"
>   0.00000MB extra
> IDL> val2=(struct.a)[10,4]
> IDL> print,(memory(/HIGHWATER)-m)/1024/1024.,"MB extra"
>   381.470MB extra
>
> So the latter method first creates a temporary variable of size
> 100,100,100,100, and then pulls a single element out of it. Not
> exactly good form.

No, I agree, not good. But the precedence rules for operators not being followed isn't too crash hot either. :o)

cheers,

paulv

--
Paul van Delst Ride lots.
CIMSS @ NOAA/NCEP/EMC Eddy Merckx