Subject: Re: Structures
Posted by Paul Van Delst[1] on Wed, 02 Mar 2005 15:33:46 GMT
View Forum Message <> Reply to Message

m_schellens@hotmail.com wrote:
> IDL> s={s,s:indgen(42)}
> IDL> ss=[s,s,s,s]
> IDL> help,ss[*].s
> <Expression> INT = Array[42, 4]
>
> Thats why they need to be fixed size.

Why's that? Doesn't it depend on what you want to do/how flexible you want the code to be?

```
IDL> s={s,s:ptr_new()}
IDL> ss=[s,s,s,s]
IDL> help, ss[*].s
<Expression>  POINTER  = Array[4]
IDL> ss[0].s=ptr_new(indgen(731))
IDL> ss[1].s=ptr_new(indgen(20))
IDL> ss[2].s=ptr_new(indgen(5))
IDL> ss[3].s=ptr_new(indgen(7))
IDL> help, *(ss[0].s)
<PtrHeapVar1>  INT  = Array[731]
IDL> help, *(ss[2].s)
<PtrHeapVar3>  INT  = Array[5]
```

Granted, the way you access all the bits and pieces has become more complex due to the pointer dereferencing, but you wouldn't do all this stuff on the command line anyway. (Right? Like objects. :o)

And, if I was doing it the way you suggested, I wouldn't access the data in ss[*].s as a rank-2 array, e.g.

```
IDL> help, (ss.s)[20,3]
<Expression>  INT  =  20
```

to get the 21st element from the 4th structure. I would reference it as

```
IDL> help, ss[3].s[20]
<Expression>  INT  =  20
```

My personal preference only of course. Doing it the first way seems sneaky

paulv

p.s. As an aside - in this context, this is another reason I like Fortran90/95 pointers. The dereferencing is implicit (i.e. no "*" out the front) so your accessing code doesn't

have to change when you modify your structure to use allocatable pointers rather than fixed size arrays. E.g. in f95, ss(3)%s(20) gets the same info whether or not the "s" component is a fixed size array or a pointer in it's definition.

--
Paul van Delst
CIMSS @ NOAA/NCEP/EMC