Subject: Is a dynamically sized pointer array object component possible?
Posted by Paul Van Delst[1] on Thu, 21 May 2009 21:35:34 GMT
View Forum Message <> Reply to Message

Hello,

I'm trying to create an object with a component that is a pointer array... but I don't know the size of the array ahead of time. In structure-speak, I'm doing this:

```
IDL> x={blah,y:ptr_new()}
IDL> x.y=ptr_new(ptrarr(4))
IDL> (*x.y)[0] = ptr_new(dindgen(20))
IDL> help, *(*x.y)[0]
<PtrHeapVar1>   DOUBLE    = Array[20]
```

Now, if I knew what size of array I needed in advance (in this example, 4), I could do the following:

```
IDL> x={blah,y:ptrarr(4)}
IDL> x.y[0] = ptr_new(dindgen(20))
IDL> help, *x.y[0]
<PtrHeapVar1>   DOUBLE    = Array[20]
```

The latter example is preferable since
  a) it more closely reflect the data, and
  b) the dereferencing is clearer.

I tried to do this:

```
IDL> x={blah,y:ptr_new()}
IDL> x.y = ptrarr(4)
% Expression must be a scalar in this context: <POINTER   Array[4]>.
% Execution halted at: $MAIN$
```

I (mostly) knew it wouldn't work, but is there a way to do this? Having a pointer to a pointer array I find..... disconcerting.

In the final application I would have the following procedure,

```
PRO blah__define
  void = { blah, y:some_fancy_definition?.... }
END
```

and then do something like,

```
PRO blah::allocate, n
  self.y = PTRARR(N_ELEMENTS(n))    ; This causes the heartache.
  FOR i = 0, N_ELEMENTS(n)-1 DO BEGIN
```

```
    self.y[i] = PTR_NEW(DBLARR(n[i]))
  ENDFOR
END
```

to be called thusly:

```
x = obj_new('blah')
x->allocate([2,5,9,25])
```

Is it doable? Am I missing another simple fix (ala the FORMAT_AXIS_VALUES function from a previous thread :o) I would like to avoid the double dereferencing if possible.

Hopefully I've explained myself. Thanks for making it this far.

cheers,

paulv
