Subject: Re: compile a routine wich inlude a commun
Posted by David Fanning on Mon, 23 Jan 2006 23:16:15 GMT
View Forum Message <> Reply to Message

Paul Van Delst writes:

> Well, they're called pointers but they're not, really. You can't actually "point" to
> anything - just make copies.

Well, they aren't fingers, if that's what you mean. :-)

> But, not being a pointer expert, let me ask the question:
> How *do* you use a pointer in IDL to, uh, well, "point" to an already created variable? Or
> just parts of an already created array?

They are variables that live in a different place
than local variables. Outside the room, if you like.
A pointer can't point to the memory locations of your local
variable because there aren't enough technical support
engineers in the world to sort out why your programs
are suddenly crashing right and left.

Goodness, *all* variables work like this, as far as I
can tell.

```
a = 5
b = a
b = 3
print, a
```

And I don't think you really want to see a 3 printed out!
Chaos (and I have a high tolerance, myself) would reign supreme.

But, there is no reason you can do what you want to do.

```
; Create a local variable.
x = Indgen(4,4)

; Move the variable out of the room.
ptr = Ptr_New(x, /No_Copy)
help, x
   X   UNDEFINED = <Undefined>

; Change a subset of the data
(*ptr)[1:2, 1:2] = (*ptr)[1:2, 1:2] * 100

; Move the variable back into the room
x = Temporary(*ptr)
```

```
Help, *ptr
  <PtrHeapVar2>   UNDEFINED = <Undefined>

Print, x
     0      1      2      3
     4    500    600      7
     8    900   1000     11
    12     13     14     15
```

The advantage of pointers as variables, is that they don't go away when you exit the local program unit, so they can be used by any program unit made aware of them. That's pretty powerful.

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
