
Subject: Re: compile a routine wich include a commun
Posted by [Paul Van Delst\[1\]](#) on Mon, 23 Jan 2006 22:24:32 GMT
[View Forum Message](#) <> [Reply to Message](#)

David Fanning wrote:

> To avoid pointers!? Are you a Luddite? Pointers
> are the coolest thing *in* IDL. Global, sticky, variables
> that act *exactly* like any other IDL variables. Fantastic!
> I think almost everyone would agree it is one thing RSI got
> *exactly* right.

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

E.g.

```
IDL> x=indgen(4,4)
IDL> print, x
    0    1    2    3
    4    5    6    7
    8    9   10   11
   12   13   14   15
IDL> p=ptr_new(x[1:2,1:2])
IDL> print, *p
    5    6
    9   10
IDL> *p=*p+100
IDL> print, *p
   105   106
   109   110
IDL> print, x
    0    1    2    3
    4    5    6    7
    8    9   10   11
   12   13   14   15
```

If pointers in IDL worked right (and by "right", I mean how *I* intuitively expect them to work <take grain of salt here>) I would expect the "print, x" command to output the following:

```
IDL> print, x
    0    1    2    3
    4   105   106    7
    8   109   110   11
   12   13   14   15
```

In my pointer-naivete, it seems to me that p should point to the memory that x occupies. But that's not what happens. I expect it to work like the following Fortran95 program:

```

program testptr
  integer,parameter::n=4
  integer,target::x(0:n-1,0:n-1)
  integer,pointer::p(:,:)
  x=reshape((/(i-1,i=1,n*n)/),(/n,n/))
  write(*,('Print x:'))
  write(*,'(4i5)')x
  p=>x(1:2,1:2)
  write(*,('Print p:'))
  write(*,'(2i5)')p
  p=p+100
  write(*,('Print p added to:'))
  write(*,'(2i5)')p
  write(*,('Print x:'))
  write(*,'(4i5)')x
end program testptr

```

Inx:scratch : lf95 testptr.f90

Encountered 0 errors, 0 warnings in file testptr.f90.

Inx:scratch : a.out

Print x:

```

 0  1  2  3
 4  5  6  7
 8  9 10 11
12 13 14 15

```

Print p:

```

 5  6
 9 10

```

Print p added to:

```

105 106
109 110

```

Print x:

```

 0  1  2  3
 4 105 106  7
 8 109 110 11
12 13 14 15

```

Maybe PTR_NEW() should be renamed to something else? Like, um, HEAPVAR_NEW() ?

I'm sure all of this has something to do with the pass-by-reference/pass-by-value nature of certain things in IDL.

paulv

--

Paul van Delst

