Subject: Re: compile a routine wich inlude a commun
Posted by Paul Van Delst[1] on Tue, 24 Jan 2006 14:42:18 GMT
View Forum Message <> Reply to Message

Peter Albert wrote:
> Just my 2 cents: imho, the problem with the given example
>
> p=ptr_new(x[1:2,1:2])
>
> is that x[1:2, 1:2] on the right hand side of the equation is, if I
> remember correctly, actually a new temporary variable. So p is pointing
> to a temporary variable which has nothing in commion with the variable
> x, apart from the fact that its initial values equal the appropriate
> ones of x. After that, any operation on p is totally disconnected to x.
> On the other hand, Davids example used
>
> p = ptr_new(x)
>
> and here, p is actually pointing to the memory space occupied by x, so
> modifying *p actually does modify x.

No, it does not. Whether or not you point to the "complete" variable, or just a part of it, a new heap "variable" is created.

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

```
       8       9      10      11
      12      13      14      15
```

> And yes, for this little example the effect is the same when using
> temporary() two times, as Paul suggested. But this is not the, umh,
> only point of pointers. I can hardly imagine how something like Davids
> highly appreciated linked list object would work using temporary()
> instead of pointers.

I agree, but you create a link list object once and that's pretty much it. If you've done your job right, you don't have to do much else except use the object. On the other hand, aliasing is something that comes in handy in day-to-day usage. My Fortran95 linked list code hasn't been touched pretty much since I wrote it. But I use aliasing a lot to, for example, concatenate complicated and large data structures. Aliasing let's me manipulate the structures more naturally with minimum copying/creation of temporary vars.

> But well, looking at Pauls original wish of using pointers to alias
> subsets of an array; given the fact that x[...] actually creates a new
> temporary variable makes me feel that this is actually not possible in
> IDL.

That's what I'm thinking also. There's probably some fundamental design issue that prevents it from being a simple thing to do. Sort of like passing structure components as arguments and expecting them to be writable in the called procedure - the pass-by-reference/value issue previosuly mentioned.

> Of course, there might be a way using histogram ...

That's possible :o)  But, then, they would be histopointers. Or maybe pointygrams? :o)

paulv

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