Subject: IDL V5 and Pointers
Posted by martin on Wed, 23 Jul 1997 07:00:00 GMT

I just started using pointers in IDL. Does anyone know how to get create a
pointer that points to an existing variable, without duplicating the
variable and without making it undefined?

For example:

x=findgen(1000,1000,1000) ; a big array
.
.
.
lots of code
.
.
.
.
ptr=ptr_new()
ptr=&x   ; this syntax is derived from C and is not valid in IDL.
              ; Does anyone know how to do the equivalent in IDL?

Thanks,

Charles Martin

Subject: Re: IDL V5 and Pointers
Posted by David Foster on Fri, 25 Jul 1997 07:00:00 GMT

Charles Martin wrote:
>
> I just started using pointers in IDL. Does anyone know how to get create a
> pointer that points to an existing variable, without duplicating the
> variable and without making it undefined?
>
> For example:
>
> x=findgen(1000,1000,1000) ; a big array
> .
> .
> .
> lots of code
> .
> .
> .

```
> .
> ptr=ptr_new()
> ptr=&x        ; this syntax is derived from C and is not valid in IDL.
>               ; Does anyone know how to do the equivalent in IDL?
```

Charles -

This facility does not exist in IDL, and I believe it is because
of the rather unstable nature of variables and their addresses in
IDL. Hopefully someone will explain this more completely, as I
would like to hear it myself.

You can always create your pointer like this:

```
 ptr = ptr_new(x)   ; Your big array, x is STILL defined!
```

Creating your pointer does not undefine x. If you don't want two
copies you can do:

```
 ptr = ptr_new( findgen(1000,1000,1000) )
```

Dave
--

  ~~~~~~ ~~~~~~
   David S. Foster   Univ. of California, San Diego
    Programmer/Analyst   Brain Image Analysis Laboratory
   foster@bial1.ucsd.edu Department of Psychiatry
    (619) 622-5892   8950 Via La Jolla Drive, Suite 2200
                 La Jolla, CA 92037
  ~~~~~~ ~~~~~~

   "I have this theory that if we're told we're bad,
      then that's the only idea we'll ever have.
     But maybe if we are surrounded in beauty,
       someday we will become what we see."   - Jewel Kilcher