Subject: Re: Q: pointers inside of an array of structures?
Posted by davidf on Wed, 16 Aug 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Amara Graps (Amara.Graps@mpi-hd.removethis.mpg.de) writes:

> But what about varying data lengths inside of the array of
> structures, for example, if my freq arrays are each a different
> length:
>
> IDL> thisstruc = {orbit:'', freq:ptr_new()}
> IDL> periodcube = replicate(thisstruc,1)
>
> ;Put values into the structure
> IDL> periodcube(0).orbit='G2'
> IDL> *periodcube(0).freq=DINDGEN(50)
> % Unable to dereference NULL pointer: <POINTER (<NullPointer>)>.
> % Execution halted at: $MAIN$
>
> What am I doing wrong?

Uh, trying to dereference a NULL pointer, it looks like. :-)

You can create a NULL pointer like this:

```
IDL> ptr = Ptr_New()
```

You can tell this is a NULL pointer because it is invalid:

```
IDL> Print, Ptr_Valid(ptr)
   0
```

And, of course, it is illegal to dereference an invalid pointer:

```
IDL> *ptr = 5
% Unable to dereference NULL pointer: <POINTER  (<NullPointer>)>.
```

What I think you want in this case is a pointer to an undefined variable. Another way to say this is that you want to allocate some memory on the heap, but you don't want that memory to actually point to anything yet. One way to do that is like this:

```
IDL> goodPtr = Ptr_New(/Allocate_Heap)
```

Now, even though this doesn't point to anything, it *is* a valid pointer:

```
IDL> Print, Ptr_Valid(goodPtr)
```

1

This means it can be dereferenced:

```
IDL> *goodPtr = 5
```

The only caveat is that you have to be just as careful using this dereferenced pointer in an expression as you do using an undefined variable.

So, you have a couple of choices in your program. One choice (this one immediately solves your problem) would be creating your initial structure like this:

```
thisstruc = {orbit:'', freq:ptr_new(/allocate_heap)}
periodcube = replicate(thisstruc,1)
*periodcube(0).freq=DINDGEN(50)
```

The other choice would be to check if you have a valid pointer before you dereference it:

```
thisstruc = {orbit:'', freq:ptr_new()}
periodcube = replicate(thisstruc,1)
IF Ptr_Valid(periodcube(0).freq) THEN *periodcube(0).freq=DINDGEN(50) $
   ELSE periodcube(0).freq = Ptr_New(DINDGEN(50))
```

Now you are free to put whatever you like in each structure's pointer. In other words, the frequency field can not be just variable length arrays, it can be completely different data!

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155