Subject: Re: Structure field concatenation
Posted by Liam E. Gumley on Wed, 06 Sep 2000 14:26:54 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
>
> Amara Graps (Amara.Graps@mpi-hd.removethis.mpg.de) writes:
>
>> I appreciate your answer, but then I am back to the same error
>> I inquired about a couple of weeks ago, i.e.:
>>
>> If I do this:
>> thisstruc = {orbit:'',freq:ptr_new()}
>> instead of this:
>> thisstruc = {orbit:'',freq:ptr_new(/allocate_heap)}
>>
>> I get this error when I start to create an array of structures
>> and fill it:
>>
>> periodcube = replicate(thisstruc,1)
>> periodcube(0).orbit = 'G2'
>> *periodcube(0).freq=DINDGEN(100) ;first pointer array is len 100
>>
>> % Unable to dereference NULL pointer: <POINTER (<NullPointer>)>.
>
> Exactly. A NULL pointer is an *invalid* pointer. Hence,
> it cannot be dereferenced. Only valid pointers can be
> dereferenced. A pointer to an undefined variable *is*
> a valid pointer and can be dereferenced, but if you
> replicate the same pointer in a bunch of structures
> all the pointers are to the same variable. It is
> indeed an oscillating universe. :-)
>
> The solution, I think, is to check to see (if you have
> no other way of knowing in your code) if you have
> a valid pointer reference before trying to fill the
> field with data. Something like this:
>
> thisstruc = {orbit:'',freq:ptr_new()}
> structs = Replicate(thisStruc, 10)
> IF Ptr_Valid(structs[5].freq) THEN $
> *structs[5].freq = FLTARR(100) ELSE $
> structs[5].freg = Ptr_New(FLATARR(100))

Or you could create the valid pointers first:

;- Create template for one record which contains a single NULL pointer
thisstruc = {orbit:'',freq:ptr_new()}

```
;- Make an array which has the same NULL pointer in 10 places
structs = replicate(thisstruc, 10)

;- Replace the null pointer with an array of 10 valid pointers
structs.freq = ptrarr(10, /allocate_heap)

;- Store your data
*(structs[0].freq) = findgen(25)
*(structs[1].freq) = dist(256)
```

Cheers,
Liam.
http://cimss.ssec.wisc.edu/~gumley
