Subject: Re: taming the shrew, a.k.a. structure
Posted by HILBERMAN on Wed, 01 Aug 2001 17:02:05 GMT
View Forum Message <> Reply to Message

To put it simply, you rock. I have now successfully created a mess: an
array of a structure
that contains another embedded structure. Unfortunately, I'm still not
'pointed' in the right
direction. When I try to apply the pointer tip to 'the mess' I get the
error:
% Conflicting data structures: <POINTER (<NullPointer>)>,MONTH_STRUCT.

Here's how I have things set up right now.
month_struct = {month_struct, name: ptr_new( ), day: ptr_new( ), temp_c:
ptr_new( )}
station = {station, number:0L, month:{month_struct}}
po_basin = replicate (station, howmany)

po_basin.month.name = ptr_new(strarr(2190))
...

Any ideas? Since station references a structure with pointers, do I have to
make a pointer to
station as well--or something similar? I can't say I know a lick about
objects, but this is
kinda seeming like a problem to be solved by an object? Oyvey.

Cheers,
Davida

Todd Clements wrote:

>   HILBERMAN <hilberma@colorado.edu> wrote:
>> I'm writing a program that takes in data and places it in a structure.
>> Everything is fine and dandy except that I would like to change the
>> length of the arrays in the structure after the data is read in and the
>> actual lengths (rather than the upper bound) of the arrays are
>> determined. I've tried to use a statement like:
>> po_basin[0].temp = (po_basin[0]).temp[0:1024]
>> but it's not working, and I don't know where to go from here. Any
>> suggestions?
>
> Pointers are going to be your friends here. They are the only way to
> change the size of data in structures at runtime (and really, you aren't
> changing the size of the data structure, but it seems like it). Pointers
> are fun and useful things, but that also means that you have to worry

> about cleaning them up when you're done.
>
> myStruct = {myStruct, array1: ptr_new()}
>
> Then, in your code:
>
> myStruct.array1 = ptr_new(fltarr(startSize))
>
> Of course, if you need to shorten or lengthen this later, you have to
> remember to dispose of the pointer that you made AFTER you make the new
> one.
>
> temp = myStruct.array1
> myStruct.array1 = ptr_new((*myStruct.array1)[0:1024])
> ptr_free, temp
>
> It's sometimes a lot of work to use pointers, but they do exactly what
> you describe you want to.
>
> Or, if you don't want to use pointers, you can do it the cheating way if
> your arrays aren't going to be too large. Put in the array the maximum
> size that you ever figure you'll use (no one will ever need more than
> 540K, right? =), and also keep an array size indicator in your
> structure, such as:
>
> myStruct = { myStruct, array1: fltarr(10000), maxArray1: 0L }
>
> Then you set maxArray1 to the "size" of the array and make sure to pay
> attention to that when you use the array.
>
>> Also, is there a way to make an array of an array of a structure, i.e.
>> something.something.data?
>> Please say 'yes'
>
> 'yes'
>
> struct1 = {struct1, a: 0, b: 0 }
> struct2 = {struct2, c: {struct1}, d: 0 }
>
> struct2.c.a = 3 ;; this works
>
> Good luck with the program. Hope this helps!
> Todd