Subject: Re: taming the shrew, a.k.a. structure
Posted by Todd Clements on Tue, 31 Jul 2001 22:35:37 GMT
View Forum Message <> Reply to Message

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

Subject: Re: taming the shrew, a.k.a. structure
Posted by david[2] on Tue, 31 Jul 2001 23:26:49 GMT

View Forum Message <> Reply to Message

Todd Clements writes:

> 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))
>

Too true. Although one could easily write a little
program to free structure pointers:

```
  PRO Free_Pointers, structure
  FOR j=0,N_Tags(structure)-1 DO BEGIN
     type = Size(structure.(j), /TName)
     IF type EQ 'POINTER' THEN Ptr_Free, structure.(j)
  ENDFOR
  END
```

Extra credit for making this recursive. :-)

> 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.

Not really. The nice thing about IDL pointers is that
IDL will take care of all the memory manipulation
for you. You don't have to worry about it at all.

```
>  temp = myStruct.array1
>  myStruct.array1 = ptr_new( (*myStruct.array1)[0:1024] )
>  ptr_free, temp
```

This really becomes nothing more than this:

```
   *myStruct.array1 = (*myStruct.array1)[0:1024]
```

There is no need to free the old pointer, make
a new one etc. Pointers are like IDL variables
in this respect. They *always* point to the
current thing you have pointed them too.

> It's sometimes a lot of work to use pointers, but they do exactly what
> you describe you want to.

If by "a lot of work" you mean you have to use
more parentheses than normal, I would agree with
you. Sometimes that syntax drives me crazy! But
the benefits you gain far exceed the cost.

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

Subject: Re: taming the shrew, a.k.a. structure
Posted by marc schellens[1] on Wed, 01 Aug 2001 14:35:44 GMT
View Forum Message <> Reply to Message

Using IDL 5.4 I get the following:

```
IDL> month_struct = {month_struct, name: ptr_new( ), day: ptr_new( ),
temp_c:ptr_new( )}
IDL> station = {station, number:0L, month:{month_struct}}
IDL> po_basin = replicate (station,10)
IDL> help,po_basin
```

```
PO_BASIN        STRUCT    = -> STATION Array[10]
IDL> po_basin.month.name = ptr_new( strarr(2190) )
IDL> help,po_basin.month.name
<Expression>    POINTER   = Array[10]
IDL> help,po_basin[0].month.name
<Expression>    POINTER   = <PtrHeapVar1>
IDL> help,po_basin[1].month.name
<Expression>    POINTER   = <PtrHeapVar1>
```

note that you set all 10 (or howmany ever) pointer to the SAME heap
variable. I assume somehow later therefore you will get an error.

This is a situation were you definitely (objections?) need a for loop:

```
for i=0,9 do po_basin[i].month.name= ptr_new( strarr(2190))
```

Anyway, your snippet works without error.
But you are right,
an object would do the task indeed more elegant/ less error-prone.

hope this helps,
marc

HILBERMAN wrote:
>
> 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

Subject: Re: taming the shrew, a.k.a. structure
Posted by Todd Clements on Wed, 01 Aug 2001 15:55:53 GMT
View Forum Message <> Reply to Message

david@dfanning.com (David Fanning) wrote:
>> temp = myStruct.array1
>> myStruct.array1 = ptr_new((*myStruct.array1)[0:1024])
>> ptr_free, temp
>
> This really becomes nothing more than this:
>
>   *myStruct.array1 = (*myStruct.array1)[0:1024]
>
> There is no need to free the old pointer, make
> a new one etc. Pointers are like IDL variables
> in this respect. They *always* point to the
> current thing you have pointed them too.

That's a useful thing to know. I didn't realize that IDL would do that
for you. I guess coming from the world of C, my default reaction is to
think that pointers are static and you have to make sure to explicitly
resize them appropriately if that's what you want to do. I guess that
works with all variables in C, but I'd gotten used to regular variables
not being static.

Well, guess I can shut my brain off for the day since I've already
learned something new. =)

Todd

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
```

Subject: Re: taming the shrew, a.k.a. structure
Posted by Pavel A. Romashkin on Wed, 01 Aug 2001 17:39:08 GMT
View Forum Message <> Reply to Message

I can't really see why you are getting an error. I copy-pasted your code onto the command line and it works just fine. I got rid of a few spaces though :-)

```
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, 10)
po_basin.month.name = ptr_new(strarr(2190))
```

Cheers,
Pavel

BTW, I'd suggest using /Allocate keyword when making those pointers.
This way you can simply say
*(po_basin.month.name)[i] = strarr(2190). Also, looks to me that all
elements of po_basin in your example will have the same strarr(2190) as
their Month.Name. Is this what you wanted? If you want different
contents, you will have to specify pointer contents individually for
every element of the array.

Subject: Re: taming the shrew, a.k.a. structure
Posted by david[2] on Wed, 01 Aug 2001 17:48:38 GMT
View Forum Message <> Reply to Message

HILBERMAN writes:
>
> 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.

Here is what I tried:

```
IDL> month_struct = {month_struct, name: ptr_new( ), day: ptr_new(), $
      temp_c: Ptr_New()}
IDL> station = {station, number:0L, month:{month_struct}}
IDL> po_basin = replicate (station, 2)
IDL> po_basin.month.name = ptr_new( strarr(2190) )
IDL> *((po_basin)[0].month.name) = 'dad'
IDL> *((po_basin)[1].month.name) = 'mom'
```

What is the problem!?

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

Subject: Re: taming the shrew, a.k.a. structure
Posted by HILBERMAN on Thu, 02 Aug 2001 17:30:51 GMT

View Forum Message <> Reply to Message

David Fanning wrote:

```
> Here is what I tried:
>
> IDL> month_struct = {month_struct, name: ptr_new( ), day: ptr_new(), $
>        temp_c: Ptr_New()}
> IDL> station = {station, number:0L, month:{month_struct}}
> IDL> po_basin = replicate (station, 2)
> IDL> po_basin.month.name = ptr_new( strarr(2190) )
> IDL> *((po_basin)[0].month.name) = 'dad'
> IDL> *((po_basin)[1].month.name) = 'mom'
>
> What is the problem!?
>
```

Good question!
I took snipets of advice from all the responses and played around with them a
bit, but I'm still not successful at getting the pointers to work.
Unfortunately, I'm not quite sure what I'm doing wrong. Since I only started
programming in march, and that was in C (IDL just a few weeks ago), my comfort
with pointers is...well...closer to discomfort. I appreciate you all taking the
time to help out. If you want to get the full picture, I just did a quick and
dirty job of posting the .pro file (along with the data file it requires) on my
website at
http://instaar.colorado.edu/~hilberma/
click on the idl link.

Two questions come to mind:
David, you gave the nice example of *((po_basin)[0].month.name) = 'dad' as a way to define the zeroth element of the po_basin array; is this the same sytax you would use to access and manipulate po_basin[0]? In other words, does the * function like the & does in C? I assume so, but I don't yet trust my assumptions w.r.t. IDL.

Also, Pavel mentioned a stylistic preference w.r.t. spaces--is there a good web resource for beginers that gives some tips on good IDL programming style?

Thanks!
RDH

Subject: Re: taming the shrew, a.k.a. structure
Posted by air_jlin on Fri, 03 Aug 2001 19:30:00 GMT
View Forum Message <> Reply to Message

hi Davida,

Mike Schienle at interactive visuals has posted his style guide on the web:

http://www.ivsoftware.com/IDL_Style.html

does anyone else know of other style guides?

best,
-Johnny

Johnny Lin
CIRES, University of Colorado
Work Phone: (303) 735-1636
Web: http://cires.colorado.edu/~johnny/

HILBERMAN <hilberma@colorado.edu> wrote in message
news:<3B698E4B.B61D40E9@colorado.edu>...

>
> Also, Pavel mentioned a stylistic preference w.r.t. spaces--is there a
> good web resource for beginers that gives some tips on good IDL programming
> style?
>
> Thanks!
> RDH