Subject: Re: Structure field concatenation
Posted by davidf on Thu, 31 Aug 2000 16:12:17 GMT
View Forum Message <> Reply to Message

Ben Tupper (pemaquidriver@tidewater.net) writes:

> I know that this has been a subject of some discussion recently... but
> I'm still not on firm footing on the best method of changing the size of
> an anonymous structure's fields (i.e. I want to increase or decrease the
> size of a field). The code below shows an example of how I do it now:
> creating a new structure with the appropriately sized fields. Is there
> a better method?

No, no, no. A "better method!?" Yes, there is a better method. :-)

Here is a new rule for you.

ANY time you have a structure member changing either
size or data type, make that structure member a pointer.

```
struct = { fish: Ptr_New(runOfTheMillFish), ... }
```

Later, when something changes:

```
*struct.fish = sexyNewFish
```

All you have to do is remember to clean this pointer up
in some kind of cleanup routine, or before you exit the
program. And don't worry about cleaning up memory and
all of that when you make the new assignment. IDL is
going to take care of that for you. Isn't that nice? :-)

You must have taken my class a LONG time ago. It is
this kind of embarrassing nonsense that finally got
me off my duff and writing again. I'll send you a copy
of the new book (due out in September). :-)

> BTW: I don't want to steer the discourse toward a scandalous sidebar
> discussion, but... I'm wrestling with this because a shrimp starts out
> as a male and then ends up as a female a few years later. I'm working
> with a database that has the shrimp broken down into records by sex...
> but I need to add new records for aggregate sex (that is the sum of
> males, transitionals, females,...) I never thought IDL programming
> could be so titillating.

This kind of talk is TOTALLY inappropriate in this newsgroup.
Take it on over to alt.xxx.sex.fetish.fish.

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: Structure field concatenation
Posted by promashkin on Thu, 31 Aug 2000 16:22:14 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> All you have to do is remember to clean this pointer up
> in some kind of cleanup routine

I have been using the following cleanup routine. The only downside is
that it does not clean up nested heap variables.

```
;*****************************************
; Universal cleanup routine. Kills everything it finds,
; but will miss nested pointers or ORefs.

pro PARgrDisplay::cleanup
compile_opt IDL2, OBSOLETE
;Release all pointers and object references.
tags = n_tags({PARgrDisplay})
for i=0, tags-1 do if size(self.(i), /type) eq 10 $
 then ptr_free, self.(i)
for i=0, tags-1 do if size(self.(i), /type) eq 11 $
 then obj_destroy, self.(i)
end

;*****************************************
```

Cheers,
Pavel

Subject: Re: Structure field concatenation
Posted by Craig Markwardt on Thu, 31 Aug 2000 16:28:48 GMT
View Forum Message <> Reply to Message

Your technique is not incorrect. IDL doesn't make it easy to fiddle

with structures so you're not going to get too much better than what you have now.

A few things from my experience. First, creating and recreating structures appears to be a fairly expensive operation. If you do it once in a while that's fine, but if you do it with huge arrays, or many times repetitively then you may start to feel the burn.

Second, you should be aware that IDL has some peculiar behaviors when extracting arrays with trailing dimensions of 1. Basically, those trailing dimensions disappear. For arrays with only one element, you get a scalar out, which is entirely a bug. Is this fixed in IDL 5.4? My function TAGSIZES attempts to get around these problems but it's not easy. If you always have 1D arrays you should be okay.

Finally, be prepared for people to tell you to use pointers. They actually are a very natural way to have structures with elements that vary in size. However along with that comes the extra baggage of allocating and then freeing the heap memory.

Good luck,
Craig

The bug I mentioned can be exercised via the following:
x = {a:reform(intarr(1,1),1,1)}
help, x.a ;; Result will be a scalar, not a 1x1 array

TAGSIZES available from http://cow.physics.wisc.edu/~craigm/idl/idl.html

Ben Tupper <pemaquidriver@tidewater.net> writes:

> Hello,
>
> I know that this has been a subject of some discussion recently... but
> I'm still not on firm footing on the best method of changing the size of
> an anonymous structure's fields (i.e. I want to increase or decrease the
> size of a field). The code below shows an example of how I do it now:
> creating a new structure with the appropriately sized fields. Is there
> a better method?
>
> ;++++++++++++ START
> PRO Concatenate_Str_Fields
>
> D = {A:Indgen(12), B:Indgen(12)} ;define a dummy structure
>
> Help, D, /STR
>
> Tags = Tag_Names(D)

>
> NewD = Create_Struct(Tags[0], [D.(0), Indgen(10)]) ;define the new
> structure with amended field
>
> For i = 1, N_ELEMENTS(Tags) -1 Do $;for each tag recreate
> the structure
> NewD = Create_Struct(NewD, Tags[i],[D.(i), Indgen(10)])
>
> Help, NewD, /STR
>
> END
>
> ;--------------------END
>
> Thanks,
>
> Ben
>
> BTW: I don't want to steer the discourse toward a scandalous sidebar
> discussion, but... I'm wrestling with this because a shrimp starts out
> as a male and then ends up as a female a few years later. I'm working
> with a database that has the shrimp broken down into records by sex...
> but I need to add new records for aggregate sex (that is the sum of
> males, transitionals, females,...) I never thought IDL programming
> could be so titillating.
>
> --
> Ben Tupper
> 248 Lower Round Pond Road
> POB 106
> Bristol, ME 04539
>
> Tel: (207) 563-1048
> Email: PemaquidRiver@tidewater.net
>
>

--
---------- ----------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
---------- ----------

Subject: Re: Structure field concatenation
Posted by davidf on Thu, 31 Aug 2000 16:50:33 GMT
View Forum Message <> Reply to Message

Craig Markwardt (craigmnet@cow.physics.wisc.edu) writes:

> A few things from my experience. First, creating and recreating
> structures appears to be a fairly expensive operation. If you do it
> once in a while that's fine, but if you do it with huge arrays, or
> many times repetitively then you may start to feel the burn.

I'm not sure what you mean by "expensive" here.
But if you mean every time you want to add a new
field to the structure you spend an hour and a half
of your expensive time tracking down all the places
in your code where you are defining the structure,
then I agree with you.

I say define it once and be done with it. It
will make your code a LOT easier to maintain. :-)

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: Structure field concatenation
Posted by Craig Markwardt on Thu, 31 Aug 2000 17:57:28 GMT
View Forum Message <> Reply to Message

davidf@dfanning.com (David Fanning) writes:

> Craig Markwardt (craigmnet@cow.physics.wisc.edu) writes:
>
>> A few things from my experience. First, creating and recreating
>> structures appears to be a fairly expensive operation. If you do it
>> once in a while that's fine, but if you do it with huge arrays, or
>> many times repetitively then you may start to feel the burn.
>
> I'm not sure what you mean by "expensive" here.

I meant computationally expensive. For some reason putting and
extracting large data arrays in structures is not very efficient.

Craig

--
---------- ----------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
---------- ----------

Subject: Re: Structure field concatenation
Posted by Ben Tupper on Thu, 31 Aug 2000 20:52:03 GMT
View Forum Message <> Reply to Message

Craig Markwardt wrote:

>
> I meant computationally expensive. For some reason putting and
> extracting large data arrays in structures is not very efficient.
>

Hello and thanks to all,

My experience has been quite the same, shuffling large data arrays in and out of structures is time consuming.

I guess I'm headed toward pointers as fields. In truth, that is what I have now despite the example I posted, but I can't lose my original data set (embedded in a database object), so I shall have to truck around a copy of the user's last query definition in the form of a resulting data structure. The issue comes to a head when the user calls for the data structure (via GetProperty method.) I'll be standing there with a pointer filled structure (correctly queried with something foolish like 'give me the sum of the males and the transitionals'). I could just pass the data structure full of pointers and allow the user to dereference the field pointers, or I could build an anonymous structure filled with the derefenced fields. The former is easy, but relys upon the good judgement of the caller not to free the pointers. The latter seems safer, but I will still need to build a structure.

Bing! I think I just got it. Make each of the structures (original and queried subset) the same 'automatically defined' named variety complete with pointers for each field. I'll still need to build a new derefenced anonymous structure when the caller demands the queried subset data (or even all the data for that matter) but I can live with that.

Thank you! Thank you! Thank you!

Ben

--
Ben Tupper
248 Lower Round Pond Road
POB 106
Bristol, ME 04539

Tel: (207) 563-1048
Email: PemaquidRiver@tidewater.net

Subject: Re: Structure field concatenation
Posted by Amara Graps on Tue, 05 Sep 2000 12:47:46 GMT
View Forum Message <> Reply to Message

Ben Tupper wrote:
>
> Hello and thanks to all,
>
> I guess I'm headed toward pointers as fields.

There's alot of discussions here about structures and pointers as fields inside of the structures. I'm learning alot, but still having some trouble. I have made my best attempt to getting a structure-with-pointer array concatenation to work properly, and I've not been successful yet.

Everything works fine up to the point that I concatenate the array, then I lose the previous definition of the pointer. Is there a pointer cleanup that I should be doing?

My test code is the following:

```
;-------begin test code
;Create an anonymous structure
thisstruc = {orbit:'',freq:ptr_new(/allocate_heap)}

;Create an array of anonymous structure
periodcube = replicate(thisstruc,1)

;Assign the structure values
periodcube(0).orbit = 'G2'
*periodcube(0).freq=DINDGEN(100)  ;first pointer array is len 100

;set a variable to the pointer
test1 = *periodcube[0].freq
;Take a look
```

```
help, test1   ;(it's len 100, and OK)

;Update the structure by creating a temporary structure like the
;original, and then concantenating
tempperiod = thisstruc
;Assign values to the structure
tempperiod.orbit='C3'
*tempperiod.freq = DINDGEN(50)+50  ;this pointer array is len 50

;Set a variable to the pointer
test2 = *tempperiod.freq
;Take a look
help, test2       ;(it's len 50, and OK)

;Concatenating   (here is where the problem begins)
new = [periodcube,tempperiod]

;Set variables to the pointers in the structure
freq1 = *new[0].freq
freq2 = *new[1].freq

;Take a look
help, freq1, freq2     ;WHY ARE FREQ1 FREQ2 THE SAME?
help, *new[0].freq,*new[1].freq     ;BOTH LENGTH 50.. WHY?
help, new[0].orbit, new[1].orbit   ;THESE STRINGS ARE OK THOUGH

END
;---end test code
```

Can anyone give me a hint about why the second pointer is overwriting
the definition of the first pointer?

Thanks very much, in advance,
Amara

--

 *** ***
Amara Graps | Max-Planck-Institut fuer Kernphysik
Interplanetary Dust Group | Saupfercheckweg 1
+49-6221-516-543 | 69117 Heidelberg, GERMANY
                    * http://galileo.mpi-hd.mpg.de/~graps
 *** ***

    "Never fight an inanimate object." - P. J. O'Rourke

Subject: Re: Structure field concatenation
Posted by davidf on Tue, 05 Sep 2000 14:08:21 GMT
View Forum Message <> Reply to Message

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

> Can anyone give me a hint about why the second pointer is overwriting
> the definition of the first pointer?

The problem here, Arara, is not that the second pointer is overwriting the definition of the first pointer. It is that the second pointer *IS* the first pointer! And you have re-defined what it points to.

The problem comes in how you define the original structure:

```
thisstruc = {orbit:'',freq:ptr_new(/allocate_heap)}
```

By allocating heap memory to the pointer, you make it a valid pointer (to an undefined variable). When you replicate the structure, each pointer in each structure points to the very same undefined variable. Another way of saying this is that each pointer is pointing to the same area of heap memory. Obviously, if what you store there changes, then all the pointers point to the same changed thing.

What you want to do is define your structure like this:

```
thisstruc = {orbit:'',freq:ptr_new()}
```

Then, when you fill them up:

```
new[0].freq = Ptr_New(DINDGEN(100))
new[1].freq = Ptr_New(DINDGEN(50))
```

Each freq field is a *separate* pointer to a *different* area of memory on the heap. This is what you had in mind.

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/

Subject: Re: Structure field concatenation
Posted by Amara Graps on Wed, 06 Sep 2000 13:31:23 GMT
View Forum Message <> Reply to Message

(at the risk of being the only one here who still hasn't
figured this out)

David Fanning wrote:
>
> Amara Graps (Amara.Graps@mpi-hd.removethis.mpg.de) writes:

> The problem here, Amara, is not that the second pointer is
> overwriting the definition of the first pointer. It is that
> the second pointer *IS* the first pointer! And you have
> re-defined what it points to.

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>)>.

So since my pointer problem from yesterday was that I was pointing
to the same heap, I tried creating separate structures
to concatenate:

thisstruc1 = {orbit:'',freq:ptr_new(/allocate_heap)}
thisstruc2 = {orbit:'',freq:ptr_new(/allocate_heap)}

And then I can't concatenate:
% Conflicting data structures: TEMPPERIOD,concatenation.

I seem to be living in an oscillating universe. :-(

Amara

--

 *** ***
Amara Graps | Max-Planck-Institut fuer Kernphysik
Interplanetary Dust Group | Saupfercheckweg 1
+49-6221-516-543 | 69117 Heidelberg, GERMANY
* http://galileo.mpi-hd.mpg.de/~graps
 *** ***

"Never fight an inanimate object." - P. J. O'Rourke