Subject: Array of structures.
Posted by Nicolas Decoster on Tue, 16 May 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Hi.

I'd like to build an array of structure using one line. Something like that:

arr = [{sig:a, color:'red'}, {sig:b, color:'green'}]

I don't want to use the replicate function and then fill the array one by one:

arr = replicate({myStruct, sig:a, color:'red'}, 2)
arr[0] = {myStruct, sig:a, color:'red'}
arr[1] = {myStruct, sig:b, color:'green'}

Any suggestions ?

Thanks.

Nicolas.

ps: In fact, I'm trying to find a way to handle lists of anything. In the present case I want to pass as one argument a list of signals (arrays of value) associated with a color. The list must be of any size, we must be able to build it on the fly. Something like that : {{sig1 'red'} {sig2 'green'}} in a Tcl-like syntax.

--
Tï¿½l. : 00 (33) 5 62 88 11 16
Fax : 00 (33) 5 62 88 11 12
Nicolas.Decoster@Noveltis.fr

Noveltis
Parc Technologique du Canal
2, avenue de l'Europe
31520 Ramonville Saint Agne - France

Subject: Re: Array of structures.
Posted by Ben Tupper on Tue, 16 May 2000 07:00:00 GMT
View Forum Message <> Reply to Message

"J.D. Smith" wrote:

> Nicolas Decoster wrote:

>>
>> Hi.
>>
>> I'd like to build an array of structure using one line. Something like
>> that:
>>
>> arr = [{sig:a, color:'red'}, {sig:b, color:'green'}]
>>
>> I don't want to use the replicate function and then fill the array one
>> by one:
>>
>> arr = replicate({myStruct, sig:a, color:'red'}, 2)
>> arr[0] = {myStruct, sig:a, color:'red'}
>> arr[1] = {myStruct, sig:b, color:'green'}
>>
>> Any suggestions ?
>>
>> Thanks.
>>
>> Nicolas.
>>
>> ps: In fact, I'm trying to find a way to handle lists of anything. In
>> the present case I want to pass as one argument a list of signals
>> (arrays of value) associated with a color. The list must be of any size,
>> we must be able to build it on the fly. Something like that : {{sig1
>> 'red'} {sig2 'green'}} in a Tcl-like syntax.
>
> Pointers are your salvation, but the initial request can be solved without them:
>
> arr=[{myStruct,sig:a,color:'red'},{myStruct,b,'green'}]
>
> The key being using a named array, fully filled in on the first element. If you
> don't want to use a named array, you're stuck with replicate.
>
> As for you p.s., I'd use lots o' pointers... an example in an object oriented
> design:
>
> pro myClass__define
> struct={COLSIG, $
> Color:'', $
> Signals:ptr_new()} ; pointer to a list of signal values
>
> struct={myClass, $
> colsigs:ptr_new(),$;pointer to list of structs of type COLSIG
> blah:0L, $
> ...}
> end
>

> So you have the following flexibility:
>
> 1. each colsig record can have any number of signals associated with a given
> color.
> 2. There can be as many such colsig color-signal pairings (structs) as you like.
>
> Simply make sure to use the appropriate "if ptr_valid(self.colsigs)" and "if
> ptr_valid((*self.colsigs)[1].Signals) to test for the validity of a given record
> before using it.... though see some prior postings about the advantages of
> splitting up such complicated dereferencing/indexing statements.
>
> You can easily find records for a given color like:
>
> wh=where((*self.colsigs).Color eq 'Green',cnt)
>
> just remember that structure field extraction and array indexing have higher
> precedence than pointer derefencing. This is confusing since the manual lists
> pointer derefence as second in Operator Precedence, just below parentheses. But
> "." and "[]" are not included on that list! So just remember: "*" is weak,
> comparatively speaking, which is why you can get away with things like:
>
> a={Foo,b:ptrarr(3)}
> c=*a.b[0]
>
> both the "." and the "[0]" are evaluated *before* the "*". This is why we had
> to use parentheses above: we wanted to take a subscript or a structure field of
> the thing *pointed to* by self.colsigs, so we had to empower our "*" operator
> with parentheses.
>

Hello,

I just wanted to thank you for always giving such detailed answers.

They are keepers!

Ben

--
Ben Tupper

Bigelow Laboratory for Ocean Science
tupper@seadas.bigelow.org

pemaquidriver@tidewater.net

Subject: Re: Array of structures.
Posted by John-David T. Smith on Tue, 16 May 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Nicolas Decoster wrote:
>
> Hi.
>
> I'd like to build an array of structure using one line. Something like
> that:
>
> arr = [{sig:a, color:'red'}, {sig:b, color:'green'}]
>
> I don't want to use the replicate function and then fill the array one
> by one:
>
> arr = replicate({myStruct, sig:a, color:'red'}, 2)
> arr[0] = {myStruct, sig:a, color:'red'}
> arr[1] = {myStruct, sig:b, color:'green'}
>
> Any suggestions ?
>
> Thanks.
>
> Nicolas.
>
> ps: In fact, I'm trying to find a way to handle lists of anything. In
> the present case I want to pass as one argument a list of signals
> (arrays of value) associated with a color. The list must be of any size,
> we must be able to build it on the fly. Something like that : {{sig1
> 'red'} {sig2 'green'}} in a Tcl-like syntax.

Pointers are your salvation, but the initial request can be solved without them:

arr=[{myStruct,sig:a,color:'red'},{myStruct,b,'green'}]

The key being using a named array, fully filled in on the first element. If you don't want to use a named array, you're stuck with replicate.

As for you p.s., I'd use lots o' pointers... an example in an object oriented design:

```
pro myClass__define
 struct={COLSIG,   $
  Color:'', $
  Signals:ptr_new()} ; pointer to a list of signal values

 struct={myClass, $
  colsigs:ptr_new(),$  ;pointer to list of structs of type COLSIG
```

```
  blah:0L, $
  ...}
end
```

So you have the following flexibility:

1. each colsig record can have any number of signals associated with a given color.
2. There can be as many such colsig color-signal pairings (structs) as you like.

Simply make sure to use the appropriate "if ptr_valid(self.colsigs)" and "if ptr_valid((*self.colsigs)[1].Signals) to test for the validity of a given record before using it.... though see some prior postings about the advantages of splitting up such complicated dereferencing/indexing statements.

You can easily find records for a given color like:

```
wh=where((*self.colsigs).Color eq 'Green',cnt)
```

just remember that structure field extraction and array indexing have higher precedence than pointer derefencing. This is confusing since the manual lists pointer derefence as second in Operator Precedence, just below parentheses. But "." and "[]" are not included on that list! So just remember: "*" is weak, comparatively speaking, which is why you can get away with things like:

```
a={Foo,b:ptrarr(3)}
c=*a.b[0]
```

both the "." and the "[0]" are evaluated *before* the "*". This is why we had to use parentheses above: we wanted to take a subscript or a structure field of the thing *pointed to* by self.colsigs, so we had to empower our "*" operator with parentheses.

Good luck,

JD

--
J.D. Smith |*| WORK: (607) 255-5842
Cornell University Dept. of Astronomy |*| (607) 255-6263
304 Space Sciences Bldg. |*| FAX: (607) 255-5875
Ithaca, NY 14853 |*|

Subject: Re: Array of Structures
Posted by K. Bowman on Thu, 15 Apr 2004 15:29:01 GMT
View Forum Message <> Reply to Message

In article <a6ebc8fb.0404150705.7a9f883b@posting.google.com>,
dsteinberg@whrc.org (Dan Steinberg) wrote:

> I am wondering if it is possible to create an array of structures. I
> want to read multiple different anonymous structures into a single
> variable so I can reference each structure using the variable name and
> the array index. Is this possible?
>
> -Dan

See "Arrays of Structures" in Chapter 7 of "Building IDL Applications.
You can use REPLICATE to create the array of structures.

Regards, Ken Bowman

Subject: Re: Array of Structures
Posted by R.Bauer on Thu, 15 Apr 2004 15:51:17 GMT
View Forum Message <> Reply to Message

Dan Steinberg wrote:

> I am wondering if it is possible to create an array of structures. I
> want to read multiple different anonymous structures into a single
> variable so I can reference each structure using the variable name and
> the array index. Is this possible?
>
> -Dan

Yes

but there is a small risk that it only works with named structures.
It is only guaranted that it works with named structures.

First way

```
IDL> a=create_struct(name='tst','file','','bmp',bytarr(10,20))
IDL> help,a,/str
** Structure TST, 2 tags, length=212, data length=212:
   FILE            STRING    ''
   BMP             BYTE      Array[10, 20]

IDL> s=replicate(a,10)
IDL> help,s
S               STRUCT    = -> TST Array[10]
```

second way is to use the structure name

IDL> b=[a,{tst}]
IDL> help,b
B   STRUCT   = -> TST Array[2]

third way

IDL> c=[a,a]
IDL> help,c,/str
IDL> help,c
C   STRUCT   = -> TST Array[2]

This was the creation.

Accessing:

IDL> help,c[0].(0)
<Expression>   STRING   = ''

IDL> help,c[0].(1)
<Expression>   BYTE   = Array[10, 20]

IDL> help,c[0].(1)[0,*]
<Expression>   BYTE   = Array[1, 20]

IDL> help,c[0].bmp[0,*]
<Expression>   BYTE   = Array[1, 20]

If the sizes or type of your variables are different I suggest to use
pointer.

Cheers

Reimar
--
Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de
http://www.fz-juelich.de/icg/icg-i/
 ========== ======
a IDL library at ForschungsZentrum Juelich
 http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_lib_intro. html

Subject: Re: Array of Structures
Posted by Craig Markwardt on Thu, 15 Apr 2004 16:03:59 GMT
View Forum Message <> Reply to Message

Kenneth Bowman <k-bowman@null.tamu.edu> writes:

> In article <a6ebc8fb.0404150705.7a9f883b@posting.google.com>,
> dsteinberg@whrc.org (Dan Steinberg) wrote:
>
>> I am wondering if it is possible to create an array of structures. I
>> want to read multiple different anonymous structures into a single
>> variable so I can reference each structure using the variable name and
>> the array index. Is this possible?
>>
>> -Dan
>
> See "Arrays of Structures" in Chapter 7 of "Building IDL Applications.
> You can use REPLICATE to create the array of structures.

That was my first reaction too, but if you read the OP carefully, he discusses "multiple *different* anonymous structures," (emph added).

It is impossible to make an array of dissimilar structures in IDL. The poster really must use an array of pointers, and then each pointer can point to a different structure.

Craig

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

Subject: Re: Array of Structures
Posted by Rick Towler on Thu, 15 Apr 2004 16:38:53 GMT
View Forum Message <> Reply to Message

"Dan Steinberg" wrote in message...

> I am wondering if it is possible to create an array of structures. I
> want to read multiple different anonymous structures into a single
> variable so I can reference each structure using the variable name and
> the array index. Is this possible?

Yes, but if when you say different, you mean that their lengths are different, you need to create an array of pointers which point to your

structures.

You can have arrays of identical structures:

```
IDL> x=[{a:'one',b:1},{a:'two',b:2},{a:'three',b:3}]
IDL> help, x
X               STRUCT    = -> <Anonymous> Array[3]
IDL> print, x[2].a
three
```

You can have arrays of structs that are the same length but their *initial*
tags are different:

```
IDL> y=[{a:'one',b:1},{c:'two',d:2},{e:'three',f:3}]
IDL> help, y[0], /struct
** Structure <15b85b0>, 2 tags, length=16, data length=14, refs=2:
   A               STRING    'one'
   B               INT             1
IDL> help, y[1], /struct
** Structure <15b85b0>, 2 tags, length=16, data length=14, refs=2:
   A               STRING    'two'
   B               INT             2
IDL> help, y[2], /struct
** Structure <15b85b0>, 2 tags, length=16, data length=14, refs=2:
   A               STRING    'three'
   B               INT             3
```

But you can not have an array whose elements differ in length:

```
IDL> y=[{a:'one',b:1},{a:'two',b:2,c:2.2},{a:'three',b:3,d:0D}]
% Conflicting data structures: <STRUCT    Array[1]>,<STRUCT    Array[1]>.
% Execution halted at: $MAIN$
```

In this case you need to create an array of pointers which point to your
structs:

```
z =
 PTR_NEW([PTR_NEW({a:'one',b:FLTARR(3)}),PTR_NEW({a:'two',b:D BLARR(10)})])

IDL> help, *(*z)[0], /struct
** Structure <1b20198>, 2 tags, length=24, data length=24, refs=1:
```

```
   A               STRING    'one'
   B               FLOAT     Array[3]
IDL> help, *(*z)[1], /struct
** Structure <1bc3420>, 2 tags, length=96, data length=92, refs=1:
   A               STRING    'two'
   B               DOUBLE    Array[10]
```

-Rick

Subject: Re: Array of Structures
Posted by R.Bauer on Thu, 15 Apr 2004 17:39:55 GMT
View Forum Message <> Reply to Message

Craig Markwardt wrote:

>
> Kenneth Bowman <k-bowman@null.tamu.edu> writes:
>
>> In article <a6ebc8fb.0404150705.7a9f883b@posting.google.com>,
>> dsteinberg@whrc.org (Dan Steinberg) wrote:
>>
>>> I am wondering if it is possible to create an array of structures. I
>>> want to read multiple different anonymous structures into a single
>>> variable so I can reference each structure using the variable name and
>>> the array index. Is this possible?
>>>
>>> -Dan
>>
>> See "Arrays of Structures" in Chapter 7 of "Building IDL Applications.
>> You can use REPLICATE to create the array of structures.
>
> That was my first reaction too, but if you read the OP carefully, he
> discusses "multiple *different* anonymous structures," (emph added).
>
> It is impossible to make an array of dissimilar structures in IDL. The
> poster really must use an array of pointers, and then each pointer can
> point to a different structure.
>
> Craig
>

I can show a trick with two of our routines to get arrayed pointers on the tags and then we can do a reformation of the structure if this is necessary.

In difference to your answer Craig the pointers are on the tags not several structures are as pointer arrayed. One tag could be changed without redefinition of the whole structure. But I am not sure if this is wanted at the moment. So this is only an addition to your answer.

```
IDL> names=['file','img']
IDL>
 ptr=[ptr_new([ptr_new('file1'),ptr_new('file2')]),ptr_new([p
tr_new(bytarr(10,5)),ptr_new(bytarr(10,10))])]

IDL> s=names_and_ptrs2struct(names,ptr)
IDL> help,s,/str
** Structure <81de694>, 2 tags, length=16, data length=16, refs=1:
   FILE            POINTER   Array[2]
   IMG             POINTER   Array[2]

IDL> help,*s.img[0]
<PtrHeapVar21>  BYTE      = Array[10, 5]

IDL> help,*s.img[1]
<PtrHeapVar22>  BYTE      = Array[10, 10]


IDL> x=reform_struct(s,2,/struct_arr)
% Compiled module: REFORM_STRUCT.
IDL> help,x,/str
** Structure <81de814>, 2 tags, length=8, data length=8, refs=1:
   FILE            POINTER   <PtrHeapVar18>
   IMG             POINTER   <PtrHeapVar21>
IDL> help,x
X               STRUCT    = -> <Anonymous> Array[2]
```

The routines are in the library at
 http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_lib_intro. html

regards
Reimar

--
Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de
http://www.fz-juelich.de/icg/icg-i/
 ========== ======
a IDL library at ForschungsZentrum Juelich