Subject: Re: Creating pointer in structure
Posted by David Fanning on Thu, 01 Nov 2001 15:01:58 GMT
View Forum Message <> Reply to Message

K. Bowman (k-bowma@null.tamu.edu) writes:

> If I need to define a structure containing a pointer before I know the
> characteristics of the associated heap variable, which of the following
> is preferable? Does it make any difference, or is it simply a matter
> of programming taste?
>
>
> For example:
>
> a = {point: PTR_NEW()} ;Create struct w/ null pointer
> ... figure out what n is
> a.point = PTR_NEW(FINDGEN(n)) ;Replace null pointer
>
>
> or
>
>
> b = {point: PTR_NEW(/ALLOCATE_HEAP)} ;Create struct w/ pointer->undef
> ... figure out what n is
> *b.point = FINDGEN(n) ;Define heap var

I think it is a matter of programming taste,
although I have to admit, I find the latter
formulation easier to work with in larger,
more complicated programs because I don't have
to worry about whether the pointer is valid or not.

Cheers,

David
--
David W. Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Creating pointer in structure
Posted by Paul van Delst on Thu, 01 Nov 2001 15:07:28 GMT
View Forum Message <> Reply to Message

"K. Bowman" wrote:

>
> If I need to define a structure containing a pointer before I know the
> characteristics of the associated heap variable, which of the following
> is preferable? Does it make any difference, or is it simply a matter
> of programming taste?

This is an interesting question.

My personal opinion is that it's a little bit of taste, but more how you design the code. Choosing which method might depend on how you use the result of PTR_VALID():

```
IDL> a = {point: PTR_NEW()}
IDL> print, ptr_valid(a.point)
  0
IDL> b = {point: PTR_NEW(/ALLOCATE_HEAP)}
IDL> print, ptr_valid(b.point)
  1
```

I prefer the former option because the PTR_VALID() result tells me that the pointer is definitely not associated with a target (and I like to start with everything nullified). The second does not without some further checking. This may or may not be an issue in your initialisation or cleanup routines.

But I think either is fine as long as you're consistent (or not... :o) IDL is more forgiving than most when freeing a disassociated pointer:

```
IDL> ptr_free,a.point
IDL> ptr_free,b.point
IDL> print, ptr_valid(a.point)
  0
IDL> print, ptr_valid(b.point)
  0
```

My only other experience with pointers is in Fortran 90 where the initial status of a pointer is undefined and testing the status of an undefined pointer is an error - i.e. you have to explicitly nullify or allocate the pointer before you can test its status (fortunately, Fortran 95 introduced the ability to nullify pointers when they're declared.)

paulv

>
> For example:
>
> a = {point: PTR_NEW()} ;Create struct w/ null pointer
> ... figure out what n is
> a.point = PTR_NEW(FINDGEN(n)) ;Replace null pointer
>
> or

>
> b = {point: PTR_NEW(/ALLOCATE_HEAP)} ;Create struct w/ pointer->undef
> ... figure out what n is
> *b.point = FINDGEN(n) ;Define heap var
>
> Thanks, Ken

--
Paul van Delst Religious and cultural
CIMSS @ NOAA/NCEP purity is a fundamentalist
Ph: (301)763-8000 x7274 fantasy
Fax:(301)763-8545 V.S.Naipaul

Subject: Re: Creating pointer in structure
Posted by Pavel A. Romashkin on Thu, 01 Nov 2001 16:56:11 GMT
View Forum Message <> Reply to Message

I use both ways, but I prefer /ALLOCATE. The /ALLOCATE is very
convenient when you know your pointer must be defined, and it saves time
when the pointer is filled later. I sometimes do not allocate heap if
the pointer may not be required (like a UVALUE property of an object,
for example). I rarely check on validity of pointers because I make my
code issue an error earlier, at the INIT stage, if a vital pointer is
not filled with necessary data, so Paul's remark does not really apply
in my case. For writing object methods, having valid pointers makes it a
lot easier, especially if a field is changeable by several methods.

Cheers,
Pavel

"K. Bowman" wrote:
>
> If I need to define a structure containing a pointer before I know the
> characteristics of the associated heap variable, which of the following
> is preferable? Does it make any difference, or is it simply a matter
> of programming taste?
>
> For example:
>
> a = {point: PTR_NEW()} ;Create struct w/ null pointer
> ... figure out what n is
> a.point = PTR_NEW(FINDGEN(n)) ;Replace null pointer
>
> or
>
> b = {point: PTR_NEW(/ALLOCATE_HEAP)} ;Create struct w/ pointer->undef
> ... figure out what n is

> *b.point = FINDGEN(n) ;Define heap var
>
> Thanks, Ken

Subject: Re: Creating pointer in structure
Posted by John-David T. Smith on Thu, 01 Nov 2001 20:57:20 GMT
View Forum Message <> Reply to Message

"K. Bowman" wrote:
>
> If I need to define a structure containing a pointer before I know the
> characteristics of the associated heap variable, which of the following
> is preferable? Does it make any difference, or is it simply a matter
> of programming taste?
>
> For example:
>
> a = {point: PTR_NEW()} ;Create struct w/ null pointer
> ... figure out what n is
> a.point = PTR_NEW(FINDGEN(n)) ;Replace null pointer
>
> or
>
> b = {point: PTR_NEW(/ALLOCATE_HEAP)} ;Create struct w/ pointer->undef
> ... figure out what n is
> *b.point = FINDGEN(n) ;Define heap var
>
> Thanks, Ken

I use both methods. When I'd like ptr_valid() to be the ultimate test of whether anything is in the slot, I use the former. For lists of variable length, including 0, this is an excellent test, since it doesn't rely on what precisely is in the slot. It does, however, force you to check before dereferencing your pointer.

Even an undefined variable could get stuck the slot to indicate something is there. It's unusual, but you could end up with:

```
foo=1 & boo=temporary(foo)
a.point=ptr_new(foo)
```

which is the same really as a.point=ptr_new(/ALLOCATE_HEAP). In this case:

```
IDL> print,n_elements(*a.point)
           0
IDL> print,ptr_valid(a.point)
```

1

On the other hand, when I have a list which will always have at least one element, I often use the latter method, to remind me of the fact. Then you can dereference freely, and use n_elements() to see just what is there.

As long as you remember the difference, you'll be fine.

One other case where having true null pointers is useful: when saving objects or structures, it's sometimes useful to kull out useless bits (like widget states and id's), which aren't really appropriate for keeping on disk. If you've stuck them behind a pointer, you can just do a:

```
state=self.myuselesswidgetstate
self.myuselesswidgetstate=ptr_new()
```

before saving and restore it afterwards.

JD

Subject: Re: Creating pointer in structure
Posted by Pavel A. Romashkin on Thu, 01 Nov 2001 22:19:18 GMT
View Forum Message <> Reply to Message

JD Smith wrote:
>
> One other case where having true null pointers is useful: when saving
> objects or structures, it's sometimes useful to kull out useless bits
> (like widget states and id's), which aren't really appropriate for
> keeping on disk.

Let's see... A widget ID costs us 4 bytes. To fill up one kilobyte on your 80 Gb disk, it'll take... wait... 250 widgets. I have never been able to make even that many. My screen isn't big enough. If I stored a droplist widget with its value array, 100 characters wide by 1000 entries long (which is crazy by itself), that would take up 100 Kb, roughly, right?

The time needed to weed out GUI elements before saving seems to be worth more than the space savings. I just re-fill my restored, useless GUI IDs with new IDs when I restore older legacy structures, and for the newer code, I drop GUI altogether for saving, creting it from scratch on restore by the Restore method.

I can appreciate it though if the reason not to store widget hierarchies

is not so much storage space driven as it contradicts JD's aestetic code
of IDL programming :-)

Cheers,
Pavel

Subject: Re: Creating pointer in structure
Posted by John-David T. Smith on Thu, 01 Nov 2001 22:49:43 GMT
View Forum Message <> Reply to Message

"Pavel A. Romashkin" wrote:
>
> JD Smith wrote:
>>
>> One other case where having true null pointers is useful: when saving
>> objects or structures, it's sometimes useful to kull out useless bits
>> (like widget states and id's), which aren't really appropriate for
>> keeping on disk.
>
> Let's see... A widget ID costs us 4 bytes. To fill up one kilobyte on
> your 80 Gb disk, it'll take... wait... 250 widgets. I have never been
> able to make even that many. My screen isn't big enough. If I stored a
> droplist widget with its value array, 100 characters wide by 1000
> entries long (which is crazy by itself), that would take up 100 Kb,
> roughly, right?
>
> The time needed to weed out GUI elements before saving seems to be worth
> more than the space savings. I just re-fill my restored, useless GUI IDs
> with new IDs when I restore older legacy structures, and for the newer
> code, I drop GUI altogether for saving, creting it from scratch on
> restore by the Restore method.
>
> I can appreciate it though if the reason not to store widget hierarchies
> is not so much storage space driven as it contradicts JD's aestetic code
> of IDL programming :-)
>

A more useful reason (for non-aesthetes) to avoid saving unnecessary
data: it may contain implicit definitions of structures or object
classes which you don't want to maintain as an external type, but keep
malleable and flexible. See the copious postings on the pitfalls of
restoring saved objects for more info on this subtle but important
problem.

JD

Subject: Re: Creating pointer in structure
Posted by K. Bowman on Fri, 02 Nov 2001 15:21:16 GMT
View Forum Message <> Reply to Message

Well isn't this just typical ... ;-)

One reply says (A), one says (B), one says (A) or (B), and then
it turns into a discussion of restoring saved objects.

I LOVE this newsgroup.

(No sarcasm intended, just a big smile on my face.)

Ken

Subject: Re: Creating pointer in structure
Posted by Ron Syml on Fri, 02 Nov 2001 16:02:32 GMT
View Forum Message <> Reply to Message

On Fri, 2 Nov 2001, K. Bowman wrote:

> One reply says (A), one says (B), one says (A) or (B), and then
> it turns into a discussion of restoring saved objects.
>
> I LOVE this newsgroup.

Me too. Saved objects... it could be worse, IMHO. The sign of any truely
dead thread is a discussion regarding the silly people at RSI thinking that
nobody still uses Alpha CPUs or Apple computers ;)

Better still, someone could have posted code from one of the *free*
alternatives to IDL... Ever wonder how to create a pointer in a structure
in
R or Yorick?

I vote we rename the newsgroup to:
'comp.lang.idl-pvwave-R-yorick-ana-scilab-octave-opendx'

Subject: Re: Creating pointer in structure
Posted by hcp on Fri, 02 Nov 2001 16:31:39 GMT
View Forum Message <> Reply to Message

In article <3BE35A84@MailAndNews.com>, Ron Syml <RonSmyl@MailAndNews.com> writes:
|> Better still, someone could have posted code from one of the *free*
|> alternatives to IDL... Ever wonder how to create a pointer in a structure
|> in R or Yorick?

I have never wanted to do this in any IDL-like language. In Fortran 95,
yes.....

|> I vote we rename the newsgroup to:
|>   'comp.lang.idl-pvwave-R-yorick-ana-scilab-octave-opendx'

I'm suspect I am supposed to hear laid-on-thick sarcasm here, but
the man has a point. While comp.lang.idl-pvwave will probably be with us for
a while and posts about free alternatives will remain only marginally on-topic
there is a case for creating comp.lang.scientific-visualisation or
something like that for more generic discussion and flamewars between
adherents of the various possibilities.

Not that I have the faintest idea whether a NG with a similar charter already
exists or how to create one if it doesnt.....

Hugh "I'm using R today. So shoot me." Pumphrey

--

==========
Hugh C. Pumphrey | Telephone 0131-650-6026
Department of Meteorology | FAX 0131-650-5780
The University of Edinburgh | Replace 0131 with +44-131 if outside U.K.
EDINBURGH EH9 3JZ, Scotland | Email hcp@met.ed.ac.uk
OBDisclaimer: The views expressed herein are mine, not those of UofE.
==========