Subject: Re: pointer question
Posted by Mark Hadfield on Thu, 22 Mar 2001 03:47:09 GMT
View Forum Message <> Reply to Message

"Ted Graves" <egraves@socrates.Berkeley.EDU> wrote in message
news:99blck$ko7$1@agate.berkeley.edu...
> Hi all,
>
> Another lurker question ... let's say you define a pointer using the
PTR_NEW
> function and assign to a variable x. As long as you keep track of x and
don't
> reassign x and lose the pointer to the heap variable, things are great.
You
> can remove the heap variable from memory using the PTR_FREE procedure.
>
> But now let's say i have a function TEST that takes a pointer as an
argument,
> and i want to create a pointer on the fly to use in TEST. So i do
something
> like
>
> result = TEST(PTR_NEW(value))
>
> where value is whatever i want the heap variable to be. What happens to
the
> heap variable assigned in this statement after TEST returns? I'm assuming
> from that because of the way it was created, a heap variable now exists
that i
> can't easily get rid of without using HEAP_GC.

Yes.

But if you have access to the code of TEST you could do this:

```
pro test, a

   ; Do something with a

   if not arg_present(a) then if ptr_valid(a) then ptr_free, a

end
```

Mark Hadfield
m.hadfield@niwa.cri.nz http://katipo.niwa.cri.nz/~hadfield
National Institute for Water and Atmospheric Research

Subject: Re: pointer question
Posted by Jaco van Gorkom on Thu, 22 Mar 2001 14:01:32 GMT
View Forum Message <> Reply to Message

Mark Hadfield wrote:
>
> "Ted Graves" <egraves@socrates.Berkeley.EDU> wrote in message
> news:99blck$ko7$1@agate.berkeley.edu...
...
>> result = TEST(PTR_NEW(value))
>>
>> where value is whatever i want the heap variable to be. What happens toth e
>> heap variable assigned in this statement after TEST returns? I'm assuming
>> from that because of the way it was created, a heap variable now exists that i
>> can't easily get rid of without using HEAP_GC.
>
> Yes.
>
> But if you have access to the code of TEST you could do this:
>
> pro test, a
>
> ; Do something with a
>
> if not arg_present(a) then if ptr_valid(a) then ptr_free, a
>
> end

Very nice! However, what if you pass in 'a' by value, e.g., from an
array of pointers?
If I call test like
for i=0, n_elements(PointerArray)-1 do test(PointerArray[i])
then I lose all the heap variables, right?

I would prefer to avoid the problem altogether by making TEST accept
both pointers and values, something like:

```
pro test, a
   if size(a, /type) ne 10 then begin
      a = ptr_new(a, /no_copy)
      a2ptr = 1
   endif else a2ptr = 0

   ; Do something with a, pointer-based.

   if a2ptr and ptr_valid(a) then begin
      a_copy = a
      a = temporary(*a)
      ptr_free, a_copy
```

endif
end

If TEST is not your own code, this could easily be done in a wrapper routine as well. The flexibility of not having to bother about pointers-or-not is great for command-line use. But then again, using heap_gc on the command line every once in a while is not a big problem either...

Jaco

Jaco van Gorkom gorkom@rijnh.nl
FOM-Instituut voor Plasmafysica "Rijnhuizen", The Netherlands

---

Subject: Re: pointer question
Posted by Paul van Delst on Thu, 22 Mar 2001 14:52:28 GMT
View Forum Message <> Reply to Message

Ted Graves wrote:
>
> Hi all,
>
> Another lurker question ... let's say you define a pointer using the PTR_NEW
> function and assign to a variable x. As long as you keep track of x and don't
> reassign x and lose the pointer to the heap variable, things are great. You
> can remove the heap variable from memory using the PTR_FREE procedure.
>
> But now let's say i have a function TEST that takes a pointer as an argument,
> and i want to create a pointer on the fly to use in TEST. So i do something
> like
>
> result = TEST(PTR_NEW(value))
>
> where value is whatever i want the heap variable to be. What happens to the
> heap variable assigned in this statement after TEST returns? I'm assuming
> from that because of the way it was created, a heap variable now exists that i
> can't easily get rid of without using HEAP_GC.
>
> Me and my sloppy programming ...

If you recognise this as sloppy programming along with all it's problems (heap variable you can't get rid of easily), then why do this? Seems to me to be a good example of how *not* to use pointers. What's wrong with:

x=ptr_new(value)
result = test(x)

?

what if value is some huge array? won't your original call suck up a bunch of memory that you can't free without the use of HEAP_GC?

From the HEAP_GC online help:

"Note - Garbage collection is an expensive operation. When possible, applications should be written to avoid losing pointer and object references and avoid the need for garbage collection."

Note the last sentence.

paulv

--
Paul van Delst A little learning is a dangerous thing;
CIMSS @ NOAA/NCEP Drink deep, or taste not the Pierian spring;
Ph: (301)763-8000 x7274 There shallow draughts intoxicate the brain,
Fax:(301)763-8545 And drinking largely sobers us again.
paul.vandelst@noaa.gov Alexander Pope.

Subject: Re: pointer question
Posted by Pavel A. Romashkin on Thu, 22 Mar 2001 16:24:16 GMT
View Forum Message <> Reply to Message

Mark Hadfield wrote:

> pro test, a
>
> ; Do something with a
>
> if not arg_present(a) then if ptr_valid(a) then ptr_free, a
>
> end

I am sorry, I have not had my first cup of coffe yet. How is that supposed to work? If there is *no* argument present, *then* try to check if the missing argument is a pointer? What am I missing? Should it be

if arg_present(a) then if ptr_valid(a) then ptr_free, a ;?

Also, in the example provided by Ted, the parameter is not going to be recognized by Arg_present, because it is not passed by reference since it is an expression.
But I think Paul answered already how to do this sort of thing properly.

Cheers,
Pavel

Subject: Re: pointer question
Posted by John-David T. Smith on Thu, 22 Mar 2001 16:56:16 GMT
View Forum Message <> Reply to Message

Ted Graves wrote:
>
> Hi all,
>
> Another lurker question ... let's say you define a pointer using the PTR_NEW
> function and assign to a variable x. As long as you keep track of x and don't
> reassign x and lose the pointer to the heap variable, things are great. You
> can remove the heap variable from memory using the PTR_FREE procedure.
>
> But now let's say i have a function TEST that takes a pointer as an argument,
> and i want to create a pointer on the fly to use in TEST. So i do something
> like
>
> result = TEST(PTR_NEW(value))
>
> where value is whatever i want the heap variable to be. What happens to the
> heap variable assigned in this statement after TEST returns? I'm assuming
> from that because of the way it was created, a heap variable now exists that i
> can't easily get rid of without using HEAP_GC.
>
> Me and my sloppy programming ...

On an only slightly related note, does everyone know that you can recover a pointer to a "lost" heap variable using ptr_valid? Here's an example:

```
IDL> a=ptr_new(1)
IDL> print,a
<PtrHeapVar4>
IDL> a='oh no, I overwrote my pointer variable'
IDL> help,/heap_variables
Heap Variables:
   # Pointer: 1
   # Object : 0

<PtrHeapVar4>   INT      =       1
IDL> a=ptr_valid(4,/CAST)
IDL> print,*a
      1
```

You can also get a vector of pointers to every heap variable using:

IDL> pvec=ptr_valid()

While this isn't exactly useful programatically, it may get you out of a pinch.

JD

Subject: Re: pointer question
Posted by Pavel A. Romashkin on Thu, 22 Mar 2001 17:21:59 GMT
View Forum Message <> Reply to Message

John-David Smith wrote:

> While this isn't exactly useful programatically, it may get you out of a pinch.

I could see even how you could get the output from Help into a string,
then use Cast to get tthe heap variables, but the effort involved seems
much larger than avoiding losing the pointers tostart with. Besides,
they say not to use Help to gather any info used in a program, as they
can change it at any time.

Cheers,
Pavel

Subject: Re: pointer question
Posted by Paul van Delst on Thu, 22 Mar 2001 18:00:12 GMT
View Forum Message <> Reply to Message

"Pavel A. Romashkin" wrote:
>
> John-David Smith wrote:
>
>> While this isn't exactly useful programatically, it may get you out of a pinch.
>
> I could see even how you could get the output from Help into a string,
> then use Cast to get tthe heap variables, but the effort involved seems
> much larger than avoiding losing the pointers tostart with. Besides,
> they say not to use Help to gather any info used in a program, as they
> can change it at any time.

And it wouldn't inspire confidence in code when the method used to recover lost pointer references is via a procedure called "help". :o)

paulv

--
Paul van Delst A little learning is a dangerous thing;
CIMSS @ NOAA/NCEP Drink deep, or taste not the Pierian spring;
Ph: (301)763-8000 x7274 There shallow draughts intoxicate the brain,
Fax:(301)763-8545 And drinking largely sobers us again.
paul.vandelst@noaa.gov Alexander Pope.

Subject: Re: pointer question
Posted by Craig Markwardt on Fri, 23 Mar 2001 04:15:54 GMT
View Forum Message <> Reply to Message

John-David Smith <jdsmith@astro.cornell.edu> writes:
> On an only slightly related note, does everyone know that you can recover a
> pointer to a "lost" heap variable using ptr_valid? Here's an example:
>
...
> IDL> a=ptr_valid(4,/CAST)
> IDL> print,*a
> 1

So, is there a way to go the *other* direction? Which is to say, if
you have a pointer, can you get its index number? [not using HELP of
course.]

Craig

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

Subject: Re: pointer question
Posted by R.Bauer on Sun, 25 Mar 2001 17:24:43 GMT
View Forum Message <> Reply to Message

Ted Graves wrote:
>
> Hi all,
>
> Another lurker question ... let's say you define a pointer using the PTR_NEW
> function and assign to a variable x. As long as you keep track of x and don't
> reassign x and lose the pointer to the heap variable, things are great. You
> can remove the heap variable from memory using the PTR_FREE procedure.

>
> But now let's say i have a function TEST that takes a pointer as an argument,
> and i want to create a pointer on the fly to use in TEST. So i do something
> like
>
> result = TEST(PTR_NEW(value))
>
> where value is whatever i want the heap variable to be. What happens to the
> heap variable assigned in this statement after TEST returns? I'm assuming
> from that because of the way it was created, a heap variable now exists that i
> can't easily get rid of without using HEAP_GC.
>
> Me and my sloppy programming ...
>
> Ted Graves
> Magnetic Resonance Science Center, UCSF

We have a routine in our library which I am using often in this case.

http://www.fz-juelich.de/icg/icg1/idl_icglib/idl_source/idl_ html/dbase/download/rec_ptr_free.tar.gz

For further routines and licensing please look at
http://www.fz-juelich.de/icg/icg1/idl_icglib/idl_lib_intro.h tml

regards

Reimar

--
Reimar Bauer

Institut fuer Stratosphaerische Chemie (ICG-1)
Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de
http://www.fz-juelich.de/icg/icg1/
==========
a IDL library at ForschungsZentrum Jï¿½lich
http://www.fz-juelich.de/icg/icg1/idl_icglib/idl_lib_intro.h tml

http://www.fz-juelich.de/zb/text/publikation/juel3786.html

Subject: Re: pointer question
Posted by Mark Hadfield on Sun, 25 Mar 2001 22:07:06 GMT

"Craig Markwardt" <craigmnet@cow.physics.wisc.edu> wrote in message
news:onelvpxgqt.fsf@cow.physics.wisc.edu...
>
> So, is there a way to go the *other* direction? Which is to say, if
> you have a pointer, can you get its index number? [not using HELP of
> course.]

Err... parse the output of string(myptrvar, /PRINT)

Mark Hadfield
m.hadfield@niwa.cri.nz http://katipo.niwa.cri.nz/~hadfield
National Institute for Water and Atmospheric Research

Subject: Re: pointer question
Posted by Mark Hadfield on Sun, 25 Mar 2001 22:16:24 GMT

"Pavel A. Romashkin" <pavel.romashkin@noaa.gov> wrote in message
news:3ABA272F.538D0EB0@noaa.gov...
> Mark Hadfield wrote:
>
>> pro test, a
>>
>> ; Do something with a
>>
>> if not arg_present(a) then if ptr_valid(a) then ptr_free, a
>>
>> end
>
> I am sorry, I have not had my first cup of coffe yet. How is that
> supposed to work? If there is *no* argument present, *then* try to check
> if the missing argument is a pointer? What am I missing?

The question answered by ARG_PRESENT is not "Is this argument present", it
is "If I change this variable, will it be passed back to the caller?". So
the intention of the code in routine "test" was to see whether the calling
program holds a reference to variable a.

I have never used this in my own code, but tt seemed like a clever idea when
I suggested it. Jaco has pointed out the flaw: what if the caller holds the
pointer heap variable (say in an array) but chooses to pass it by value?

Mark Hadfield

m.hadfield@niwa.cri.nz http://katipo.niwa.cri.nz/~hadfield
National Institute for Water and Atmospheric Research

Subject: Re: pointer question
Posted by Martin Schultz on Mon, 26 Mar 2001 09:05:13 GMT
View Forum Message <> Reply to Message

Ted Graves wrote:
>
> Hi all,
>
> Another lurker question ... let's say you define a pointer using the PTR_NEW
> function and assign to a variable x. As long as you keep track of x and don't
> reassign x and lose the pointer to the heap variable, things are great. You
> can remove the heap variable from memory using the PTR_FREE procedure.
>
> But now let's say i have a function TEST that takes a pointer as an argument,
> and i want to create a pointer on the fly to use in TEST. So i do something
> like
>
> result = TEST(PTR_NEW(value))
>
> where value is whatever i want the heap variable to be. What happens to the
> heap variable assigned in this statement after TEST returns? I'm assuming
> from that because of the way it was created, a heap variable now exists that i
> can't easily get rid of without using HEAP_GC.
>
> Me and my sloppy programming ...
>
> Ted Graves
> Magnetic Resonance Science Center, UCSF

Let me second Paul here. WHY? This is the real question here. As I relearned only recently, IDL automatically passes variables by reference unless you index them (or do whatever other weird things to them). So from a program efficiency standpoint, you are passing a pointer when you simply write

```
result = test(value)
```

Now if you really want to give test only the data of the first column (or row or whatever), you really should, as Paul suggests, create and destroy the pointer in the caller routine, i.e.:

```
x = Ptr_New(value[0,*])
result = test(x)
Ptr_Free, x
```

But even then: you don't even need a pointer here! The following is exactly the same in terms of efficiency, and it doesn't require a cleanup (if you write it inside a procedure or function).

```
x = value[0,*]
result = test(x)
```

So, why use pointers at all, the witty lurker might ask now? Let me dare to say that you only need them within structures (or objects for that purpose), i.e. if you need to access a variable at a certain place but you don't know it's shape or size beforehand.

Cheers,

Martin

--

[[[[[[[[[[
[[Dr. Martin Schultz   Max-Planck-Institut fuer Meteorologie   [[
[[Bundesstr. 55, 20146 Hamburg   [[
[[phone: +49 40 41173-308   [[
[[fax: +49 40 41173-298   [[
[[martin.schultz@dkrz.de   [[
[[[[[[[[[[

Subject: Re: pointer question
Posted by Craig Markwardt on Mon, 26 Mar 2001 12:56:16 GMT
View Forum Message <> Reply to Message

"Mark Hadfield" <m.hadfield@niwa.cri.nz> writes:

> "Craig Markwardt" <craigmnet@cow.physics.wisc.edu> wrote in message
> news:onelvpxgqt.fsf@cow.physics.wisc.edu...
>>
>> So, is there a way to go the *other* direction? Which is to say, if
>> you have a pointer, can you get its index number? [not using HELP of
>> course.]
>
> Err... parse the output of string(myptrvar, /PRINT)

Hey, that's kind of neat! (in an IDL-kludgey-sort-of-way)

Craig

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

Subject: Re: pointer question
Posted by egraves on Tue, 27 Mar 2001 01:06:44 GMT
View Forum Message <> Reply to Message

Thanks to all who chimed in on my pointer question. The background of me posing this dilemma was that i was under the assumption that IDL passed variables by value. Since i am dealing with a number of rather large data structures, i thought it would be wise to pass pointers to these structures in function calls instead of passing the structures themselves. And naturally after coding for a while, i found situations where i wanted to execute a function using a hybrid of several data structures, and instead of creating one explicitly, i hastily concocted one, used it to define a pointer with PTR_NEW, and directly passed the result of this to my function.

I have gone back and corrected this naughty practice. Thanks everyone!

Ted Graves
Magnetic Resonance Science Center, UCSF