Subject: Re: Proper pointer cleanup question
Posted by Paul Van Delst[1] on Mon, 07 Apr 2003 23:23:15 GMT
View Forum Message <> Reply to Message

"M. Katz" wrote:
>
> I want to make sure I'm taking all the steps necessary to clean up
> pointers and free memory when I'm done with them. Here's an example.
>
> Suppose I have a pointer to a structure that contains pointers.
>
> a = ptr_new({n:10, p:ptrarr(10)})
>
> So, a is a pointer, and ((*a).p)(i) are pointers as well.
> When I'm done with a and all of it's components, I can do a few things
> to clean it up, but I don't want to do more than what's necessary.
> Here's a few options.
>
> 1) Just a:
> ptr_free, a
>
> 2) a and all of its dependent pointers:
> for i=0,n_elements((*a).p)-1 do $
> ptr_free, ((*a).p)(i)
> ptr_free, a

#2 is the go. All the others leave you with dangling references and memory leaks. My personal mantra is that when it comes to pointers, be very explicit in their garbage collection i.e. don't assume freeing a pointer also frees any "child" pointers like the components "p" in your example. (I actually don't know of any languages that *do* do that, but I'm barely bilingual. :o)

>
> 3) Re-assign a to a scalar:
> a = 0
>
> 4) Re-assign *a to a scalar:
> *a = 0
>
> What's the best thing to do? My pointed-to arrays are going to get
> pretty large, so I don't want to strand any memory unnecessarily.

--
Paul van Delst
CIMSS @ NOAA/NCEP/EMC
Ph: (301)763-8000 x7748
Fax:(301)763-8545

Subject: Re: Proper pointer cleanup question
Posted by Paul Van Delst[1] on Mon, 07 Apr 2003 23:27:56 GMT
View Forum Message <> Reply to Message

Paul van Delst wrote:
>
> "M. Katz" wrote:
>>
>> I want to make sure I'm taking all the steps necessary to clean up
>> pointers and free memory when I'm done with them. Here's an example.
>>
>> Suppose I have a pointer to a structure that contains pointers.
>>
>> a = ptr_new({n:10, p:ptrarr(10)})
>>
>> So, a is a pointer, and ((*a).p)(i) are pointers as well.
>> When I'm done with a and all of it's components, I can do a few things
>> to clean it up, but I don't want to do more than what's necessary.
>> Here's a few options.
>>
>> 1) Just a:
>> ptr_free, a
>>
>> 2) a and all of its dependent pointers:
>> for i=0,n_elements((*a).p)-1 do $
>> ptr_free, ((*a).p)(i)
>> ptr_free, a
>
> #2 is the go. All the others leave you with dangling references and memory leaks. My
> personal mantra is that when it comes to pointers, be very explicit in their garbage
> collection i.e. don't assume freeing a pointer also frees any "child" pointers like the
> components "p" in your example. (I actually don't know of any languages that *do* do that,
> but I'm barely bilingual. :o)

Apologies for the replying to me own post, but one thing I've noticed in using pointers in Fortran (for large arrays that are allocated and deallocated frequently) is that nullifying them in the opposite order in which they're allocated (where possible) may minimise memory fragmentation. Anecdotal evidence only on my part, though.

paulv

--
Paul van Delst
CIMSS @ NOAA/NCEP/EMC
Ph: (301)763-8000 x7748
Fax:(301)763-8545

Subject: Re: Proper pointer cleanup question
Posted by MKatz843 on Tue, 08 Apr 2003 17:26:17 GMT
View Forum Message <> Reply to Message

>>> 2) a and all of its dependent pointers:
>>> for i=0,n_elements((*a).p)-1 do $
>>> ptr_free, ((*a).p)(i)
>>> ptr_free, a
>>
>> #2 is the go. All the others leave you with dangling references and memory leaks. My
>> personal mantra is that when it comes to pointers, be very explicit in their garbage
>> collection i.e. don't assume freeing a pointer also frees any "child" pointers like the
>> components "p" in your example. (I actually don't know of any languages that *do* do that,
>> but I'm barely bilingual. :o)
>

Thanks! I'll write myself a full reverse-ordered cleanup routine.
I suppose this explicit cleanup is just as important for objects as
well:
Object pointer fields should be explicitly freed in the Cleanup
method.

Question 1) But what about simple scalar pointers?
a = ptr_new(fltarr(10,10))

If I set
a = 0
Will I have stranded my fltarr(), or is IDL smart enough to deallocate
it properly?

Question 2) Then how about this scenario
a = ptr_new(fltarr(10,10))
b = ptr_new(dblarr(5,5))

a = b ;--- Does this strand the original a array?
*a = *b ;--- How about this?

From looking at help, /memory and testing the above, I think the
answer to both of my questions is that memory IS stranded unless you
explicitly free it in all of the above cases.

M. Katz

Subject: Re: Proper pointer cleanup question
Posted by David Fanning on Tue, 08 Apr 2003 18:03:38 GMT
View Forum Message <> Reply to Message

M. Katz (MKatz843@onebox.com) writes:

> Thanks! I'll write myself a full reverse-ordered cleanup routine.
> I suppose this explicit cleanup is just as important for objects as
> well:
> Object pointer fields should be explicitly freed in the Cleanup
> method.
>
> Question 1) But what about simple scalar pointers?
> a = ptr_new(fltarr(10,10))
>
> If I set
> a = 0
> Will I have stranded my fltarr(), or is IDL smart enough to deallocate
> it properly?

You will have leaking memory. IDL, as a weakly typed
language, always allows you to write dangerous code. :-)

> Question 2) Then how about this scenario
> a = ptr_new(fltarr(10,10))
> b = ptr_new(dblarr(5,5))
>
> a = b ;--- Does this strand the original a array?

Yes, for the same reason as above. You are re-defining A
before you have freed the memory the original pointer A
pointed to.

> *a = *b ;--- How about this?

This is perfectly OK. IDL handles pointer memory as it does
regular variable memory. Pointer A now points to the same data
as pointer B, but the memory pointer A originally pointed to
has been de-allocated by IDL.

> From looking at help, /memory and testing the above, I think the
> answer to both of my questions is that memory IS stranded unless you
> explicitly free it in all of the above cases.

No, *ptr = someNewData is always permissible. Just like
this is permissible:

```
a = 5
a = [10, 10]
```

Cheers,

David

--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
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: Proper pointer cleanup question
Posted by Paul Van Delst[1] on Tue, 08 Apr 2003 18:10:54 GMT
View Forum Message <> Reply to Message

"M. Katz" wrote:
>
>>>> 2) a and all of its dependent pointers:
>>>> for i=0,n_elements((*a).p)-1 do $
>>>> ptr_free, ((*a).p)(i)
>>>> ptr_free, a
>>>
>>> #2 is the go. All the others leave you with dangling references and memory leaks. My
>>> personal mantra is that when it comes to pointers, be very explicit in their garbage
>>> collection i.e. don't assume freeing a pointer also frees any "child" pointers like the
>>> components "p" in your example. (I actually don't know of any languages that *do* do that,
>>> but I'm barely bilingual. :o)
>>
>
> Thanks! I'll write myself a full reverse-ordered cleanup routine.
> I suppose this explicit cleanup is just as important for objects as
> well:
> Object pointer fields should be explicitly freed in the Cleanup
> method.
>
> Question 1) But what about simple scalar pointers?
> a = ptr_new(fltarr(10,10))
>
> If I set
> a = 0
> Will I have stranded my fltarr(),

Yes

> or is IDL smart enough to deallocate
> it properly?

No. To quote the IDL help, you can "reclaim" lost heap variables using the CAST keyword to PTR_VALID() but I wouldn't recommend using that on a regular basis (I've used it on the

command line after fat-fingering, but not in a routine)

> Question 2) Then how about this scenario
> a = ptr_new(fltarr(10,10))
> b = ptr_new(dblarr(5,5))
>
> a = b ;--- Does this strand the original a array?
> *a = *b ;--- How about this?

The first one would (I think....I don't like to see pointers used with "=" signs. It confuses me. I prefer the Fortran operator => or the C one ->). The second one, I'm not sure. In either case, I would explicitly destroy what I do not need anymore before reusing the variable/pointer name.

paulv

--
Paul van Delst
CIMSS @ NOAA/NCEP/EMC
Ph: (301)763-8000 x7748
Fax:(301)763-8545

Subject: Re: Proper pointer cleanup question
Posted by tam on Tue, 08 Apr 2003 18:33:08 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> M. Katz (MKatz843@onebox.com) writes:
>
>
>> Thanks! I'll write myself a full reverse-ordered cleanup routine.
>> I suppose this explicit cleanup is just as important for objects as
>> well:
>> Object pointer fields should be explicitly freed in the Cleanup
>> method.
>>
>> Question 1) But what about simple scalar pointers?
>> a = ptr_new(fltarr(10,10))
>>
>> If I set
>> a = 0
>> Will I have stranded my fltarr(), or is IDL smart enough to deallocate
>> it properly?
>
>
> You will have leaking memory. IDL, as a weakly typed
> language, always allows you to write dangerous code. :-)

>

Hi David,

I'm curious why you associate this with IDL's weak typing
(or is it just that that's another avenue for making
mistakes in IDL). Perl would reclaim memory even
though it's also weakly typed.

On a slightly different tack.... Is this situation where
users need to explicitly deallocate memory now something
that is mandated for all future versions? Or could IDL
implement garbage collection in the future?
Garbage collection seems like the kind of detail that languages at
IDL's level should take care of automatically. I hope
RSI doesn't think it's now precluded by considerations
of backwards compatibility. I don't think there's any problem
with explicit deallocations, but I could imagine someone building
programs that depend upon the existence of allocated but
unassociated memory. [Though I can't imagine why!]

Regards,
Tom McGlynn

Subject: Re: Proper pointer cleanup question
Posted by David Fanning on Tue, 08 Apr 2003 19:11:05 GMT
View Forum Message <> Reply to Message

Tom McGlynn (tam@lheapop.gsfc.nasa.gov) writes:

> I'm curious why you associate this with IDL's weak typing
> (or is it just that that's another avenue for making
> mistakes in IDL). Perl would reclaim memory even
> though it's also weakly typed.

PERL must be the software of choice then for all
important programming projects. :-)

I was just making a point that it is easy to leak
memory if you turn a pointer variable into some
other kind of variable without de-allocating the
pointer memory. Perhaps RSI could put a check in
the code to have a quick look to see if the variable
that is being re-defined is a pointer (or object, or
something that *contains* pointers or objects, etc.,
etc.), but my guess is this would slow IDL down so
much we would have all kinds of people complaining

that their prayers were answered, again.

But really, pointers and objects are SUPPOSE to
persist over time. That is the whole point of them.
How could IDL possibly know if you are *really* done
with them? You should tell IDL by freeing them
explicitly. I believe this is something that is
covered in Programming 101, if I'm not mistaken.

(And, anyway, IDL has already done us a favor
by having the *things* pointers point at act like
regular IDL variables. That's a nice touch and gives
us a lot more free time during the day to read a book
rather than free up pointers.)

> On a slightly different tack.... Is this situation where
> users need to explicitly deallocate memory now something
> that is mandated for all future versions?

Uh, yes.

> Or could IDL
> implement garbage collection in the future?

They could. I'd bet all of last month's salary (very,
very little, unfortunately) that they won't. It would
be so costly in terms of time everyone would complain.
Perhaps when quantum computers are in vogue.

> Garbage collection seems like the kind of detail that languages at
> IDL's level should take care of automatically.

I'm not sure what "level" IDL is on (and neither does
RSI, to judge from the disconnect between language features
and the people that they are trying to sell it to), but
a language that handles this automatically will be a slow
language indeed.

> I hope
> RSI doesn't think it's now precluded by considerations
> of backwards compatibility. I don't think there's any problem
> with explicit deallocations, but I could imagine someone building
> programs that depend upon the existence of allocated but
> unassociated memory. [Though I can't imagine why!]

In my experience people build all *kinds* of programs. Most
of them leave me shaking my head, to be honest. :-)

Cheers,

David

--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
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: Proper pointer cleanup question
Posted by tam on Tue, 08 Apr 2003 20:29:57 GMT
View Forum Message <> Reply to Message

... responding to David's comments on the speed of garbage
collection and the likelihood of it being implemented in IDL.
...

Hi David,

I don't believe modern garbage collectors need be slow.
It's harder to make garbage collectors that work well
in real time programming, but I don't see that as an
big issue with IDL. Modern versions of Java are quite speedy
despite being garbage collected. Even in earlier
versions speed problems were more due to slowness
in allocating objects, not in cleaning them up. Reallocating
garbage collectors might even speed up a program since they
can deal with memory fragmentation. However
I daresay your prognostication on the odds of it being
added to IDL are correct.

Regards,
Tom McGlynn

Subject: Re: Proper pointer cleanup question
Posted by JD Smith on Tue, 08 Apr 2003 23:45:43 GMT
View Forum Message <> Reply to Message

On Mon, 07 Apr 2003 16:01:27 -0700, M. Katz wrote:

> I want to make sure I'm taking all the steps necessary to clean up
> pointers and free memory when I'm done with them. Here's an example.
>

> Suppose I have a pointer to a structure that contains pointers.
>
> a = ptr_new({n:10, p:ptrarr(10)})
>
> So, a is a pointer, and ((*a).p)(i) are pointers as well. When I'm done
> with a and all of it's components, I can do a few things to clean it up,
> but I don't want to do more than what's necessary. Here's a few options.
>
> 1) Just a:
> ptr_free, a
>
> 2) a and all of its dependent pointers: for i=0,n_elements((*a).p)-1
> do $
> ptr_free, ((*a).p)(i)
> ptr_free, a
>
> 3) Re-assign a to a scalar:
> a = 0
>
> 4) Re-assign *a to a scalar:
> *a = 0

Another option is:

 heap_free,a

which will accomplish the same as #2 (avoiding a memory leak), and is
only slightly slower. When you're feeling truly lazy, it's quite a
blessing. You have no flexibility to pick and choose what to parts of
a data structure to free, but often this isn't an issue.

Good luck,

JD

Subject: Re: Proper pointer cleanup question
Posted by Paul Van Delst[1] on Wed, 09 Apr 2003 00:09:01 GMT
View Forum Message <> Reply to Message

JD Smith wrote:
>
> On Mon, 07 Apr 2003 16:01:27 -0700, M. Katz wrote:
>
>> I want to make sure I'm taking all the steps necessary to clean up
>> pointers and free memory when I'm done with them. Here's an example.
>>
>> Suppose I have a pointer to a structure that contains pointers.

>>
>> a = ptr_new({n:10, p:ptrarr(10)})

<snip>

>> 2) a and all of its dependent pointers: for i=0,n_elements((*a).p)-1
>> do $
>>   ptr_free, ((*a).p)(i)
>> ptr_free, a
>>

<snip>

> Another option is:
>
>   heap_free,a
>
> which will accomplish the same as #2 (avoiding a memory leak), and is
> only slightly slower. When you're feeling truly lazy, it's quite a
> blessing. You have no flexibility to pick and choose what to parts of
> a data structure to free, but often this isn't an issue.

Wha..? Is that another one of those undocumented IDL routines? It works on my current version, but bugger me if I can find it documented anywhere.

paulv

--
Paul van Delst
CIMSS @ NOAA/NCEP/EMC
Ph: (301)763-8000 x7748
Fax:(301)763-8545

Subject: Re: Proper pointer cleanup question
Posted by Mark Hadfield on Wed, 09 Apr 2003 01:49:40 GMT
View Forum Message <> Reply to Message

"Paul van Delst" <paul.vandelst@noaa.gov> wrote in message news:3E93649D.C9FBEBA0@noaa.gov...
> JD Smith wrote:
>> Another option is:
>>
>>   heap_free,a
>>
>> which will accomplish the same as #2 (avoiding a memory leak), and is
>> only slightly slower. When you're feeling truly lazy, it's quite a
>> blessing. You have no flexibility to pick and choose what to parts of

>> a data structure to free, but often this isn't an issue.
>
> Wha..? Is that another one of those undocumented IDL routines? It works on my current
> version, but bugger me if I can find it documented anywhere.

It's in the 5.6 documentation, which says it's been in the language since 5.3.

But I'd never heard of it either.

--

Mark Hadfield "Ka puwaha te tai nei, Hoea tatou"
m.hadfield@niwa.co.nz
National Institute for Water and Atmospheric Research (NIWA)

Subject: Re: Proper pointer cleanup question
Posted by JD Smith on Wed, 09 Apr 2003 23:29:00 GMT
View Forum Message <> Reply to Message

On Tue, 08 Apr 2003 18:49:40 -0700, Mark Hadfield wrote:

> "Paul van Delst" <paul.vandelst@noaa.gov> wrote in message
> news:3E93649D.C9FBEBA0@noaa.gov...
>> JD Smith wrote:
>>> Another option is:
>>>
>>> heap_free,a
>>>
>>> which will accomplish the same as #2 (avoiding a memory leak), and is
>>> only slightly slower. When you're feeling truly lazy, it's quite a
>>> blessing. You have no flexibility to pick and choose what to parts
>>> of a data structure to free, but often this isn't an issue.
>>
>> Wha..? Is that another one of those undocumented IDL routines? It works
>> on
> my current
>> version, but bugger me if I can find it documented anywhere.
>
> It's in the 5.6 documentation, which says it's been in the language
> since 5.3.
>
> But I'd never heard of it either.

It's rather nice. Think of it as a light-weight HEAP_GC, and you'll

understand the time-savings it can offer you. At first I felt a bit like I was cheating by using it, often inserting self-chiding comments like "Replace with a real Cleanup!", but gradually I realized I didn't really *need* to go to this trouble. The only time it's not helpful is when you want to trim various parts of your data structure selectively. This is an uncommon case though.

As far as the one remaining reason not to prefer HEAP_FREE, I decided to test the notion that it is slower than doing it "the proper way" yourself. I created an arbitrary, complex, nested structure. I spent 15 minutes figuring out how to un-roll all my pointers to structures with pointer arrays and objects so I could free them, and here's what I got:

"Correct" full loop free:

TIME: 0.15468192

HEAP_FREE time:

TIME: 0.11966300

Oh my. The supposedly cheating way is faster! Presumably this is because the internal HEAP_FREE doesn't have to pay the IDL looping penalty, which more than makes up for the dynamic data structure search it must perform to find all the heap data. Just to give you a flavor of which looks better in your Cleanup code, here are the two versions:

```
 if ptr_valid(a.a) then begin
    for i=0,n_elements(*a.a)-1 do begin
       obj_destroy,(*(*a.a)[i].dr).is
       ptr_free,(*a.a)[i].dr
    endfor
    ptr_free,(*a.a).dr,a.a
 endif
 if ptr_valid(a.b) then begin
    for i=0,n_elements(*a.b)-1 do ptr_free,*(*a.b)[i],(*a.b)[i]
    ptr_free,*a.b,a.b
 endif
```

vs.

```
 heap_free,a
```

I think I know which method I'll be preferring in the future.

Subject: Re: Proper pointer cleanup question
Posted by David Fanning on Thu, 10 Apr 2003 13:52:09 GMT
View Forum Message <> Reply to Message

JD Smith (jdsmith@as.arizona.edu) writes:

>> But I'd never heard of it either.
>
> It's rather nice. Think of it as a light-weight HEAP_GC, and you'll
> understand the time-savings it can offer you

I thought I had heard of this before. Lo and behold,
I found an article about it on that Coyote web page!
I've *really* got to spend more time reading that darn
thing.

http://www.dfanning.com/fileio_tips/memleak.html

Cheers,

David

--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
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