Subject: Re: IDL Shapefile Object
Posted by David Fanning on Thu, 29 Nov 2001 21:26:19 GMT
View Forum Message <> Reply to Message

Kelly Dean (krdean@lamar.colostate.edu) writes:

> I created a little procedure to plot Shapefiles with IDLffShape.
> However, it has a memory leak. Can someone point out a plug to stop my
> memory leak?
>
> The routine works great, but when I have to add graphics to 800 images,
> I run out of memory at around 500 images.
>
> A sample routine is available at ...
>
> ftp://ftp.cira.colostate.edu/Dean/teststate.pro

The problem here is that the shapefile object returns
a structure that itself has pointers in it. (This is really
the only way it can be done, because the object doesn't
really know anything about the actual shapefile you will
load.)

When this happens, you are responsible for cleaning
those pointers up yourself. Your particular test program
can clean itself up by changing these lines in the DrawSHPMap
module:

```
  pEnts = PTR_NEW(/ALLOCATE_HEAP)
  *pEnts = oShapefile->GetEntity(/ALL, /ATTRIBUTES)
  ;
  FOR I = N_ELEMENTS(*pEnts)-1, 0, -1 DO BEGIN
    PlotEnt, (*pEnts)[I], color=color
  ENDFOR
```

To this:

```
  pEnts = PTR_NEW(/ALLOCATE_HEAP)
  *pEnts = oShapefile->GetEntity(/ALL, /ATTRIBUTES)
  ;
  FOR I = N_ELEMENTS(*pEnts)-1, 0, -1 DO BEGIN
    PlotEnt, (*pEnts)[I], color=color
    Ptr_Free, ((*pEnts)[I]).vertices
    Ptr_Free, ((*pEnts)[I]).measure
    Ptr_Free, ((*pEnts)[I]).parts
    Ptr_Free, ((*pEnts)[I]).part_types
    Ptr_Free, ((*pEnts)[I]).attributes
```

```
  ENDFOR
  Ptr_Free, pEnts
```

That should do it. :-)

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: IDL Shapefile Object
Posted by David Fanning on Thu, 29 Nov 2001 22:31:41 GMT
View Forum Message <> Reply to Message

I wrote a few minutes ago:

> The problem here is that the shapefile object returns
> a structure that itself has pointers in it. (This is really
> the only way it can be done, because the object doesn't
> really know anything about the actual shapefile you will
> load.)
>
> When this happens, you are responsible for cleaning
> those pointers up yourself.

It has been pointed out to me that there is
a new routine in IDL 5.5 called HEAP_FREE that
is to be used for exactly this purpose. (I think
I overlooked it because it's not in the IDL 5.5 on-line
help, obviously, since the help is in... Oh, never mind.
I have probably beat that horse enough, although I still
think it is an ass-backwards way to release software.)

Anyway, despite the documentation that is meant to
frighten you away from using it (ala Heap_GC), it
is designed to help you clean up in those situations
where you don't know what it is you have been handed.
It will release (clean-up) all the heap variables
referenced by the argument to HEAP_FREE. So, in Kelly's
case, he could have cleaned up by doing something like
this:

```
  pEnts = PTR_NEW(/ALLOCATE_HEAP)
  *pEnts = oShapefile->GetEntity(/ALL, /ATTRIBUTES)
  ;
  FOR I = N_ELEMENTS(*pEnts)-1, 0, -1 DO BEGIN
    PlotEnt, (*pEnts)[I], color=color
  ENDFOR
  HEAP_FREE, pEnts
```

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: IDL Shapefile Object
Posted by Mark Hadfield on Thu, 29 Nov 2001 23:47:25 GMT
View Forum Message <> Reply to Message

From: "David Fanning" <david@dfanning.com>
> It has been pointed out to me that there is
> a new routine in IDL 5.5 called HEAP_FREE...
> ...
> Anyway, despite the documentation that is meant to
> frighten you away from using it (ala Heap_GC), it
> is designed to help you clean up in those situations
> where you don't know what it is you have been handed.

I note that the first "frightener" in the HEAP_FREE documentation is:

"When freeing object heap variables, HEAP_FREE calls OBJ_DESTROY
without supplying any plain or keyword arguments. Depending on the objects
being released, this may not be sufficient. In such cases, the caller must
call
OBJ_DESTROY explicitly with the proper arguments rather than using
HEAP_FREE."

Hey, I didn't know you *could* supply arguments to OBJ_DESTROY (though I
should have known because it's right there in the OBJ_DESTROY
documentation). Has anyone actually written code that *uses* this feature.
And if so, why? It seems to me that when you tell an object to destroy

itself, then it's up to the object to know how to do it.

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

--
Posted from clam.niwa.cri.nz [202.36.29.1]
via Mailgate.ORG Server - http://www.Mailgate.ORG

Subject: Re: IDL Shapefile Object
Posted by David Fanning on Fri, 30 Nov 2001 00:40:33 GMT
View Forum Message <> Reply to Message

Mark Hadfield (m.hadfield@niwa.cri.nz) writes:

> Hey, I didn't know you *could* supply arguments to OBJ_DESTROY (though I
> should have known because it's right there in the OBJ_DESTROY
> documentation). Has anyone actually written code that *uses* this feature.
> And if so, why? It seems to me that when you tell an object to destroy
> itself, then it's up to the object to know how to do it.

I've never used it. (Guess I should make a habit
of reading the documentation that *is* there!)
But I can imagine a case for it.

Suppose one of the fields for the object was a
pointer to some image data. The same image pointer
might be present in several objects (to save
copying the huge image). Any decent object cleanup
routine would certainly free the pointer, but maybe
you don't want it destroyed because then the other
objects that are using it wouldn't work properly.

In this case a HANG_ON_DONT_DO_IT keyword on the cleanup
method might be appropriate.

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: IDL Shapefile Object
Posted by Mark Hadfield on Fri, 30 Nov 2001 01:26:22 GMT
View Forum Message <> Reply to Message

From: "David Fanning" <david@dfanning.com>
> Suppose one of the fields for the object was a
> pointer to some image data. The same image pointer
> might be present in several objects (to save
> copying the huge image). Any decent object cleanup
> routine would certainly free the pointer, but maybe
> you don't want it destroyed because then the other
> objects that are using it wouldn't work properly.
>
> In this case a HANG_ON_DONT_DO_IT keyword
> on the cleanup method might be appropriate.

I have run into this situation and when I did (in my ignorance) I added a HANG_ON_DONT_DESTROY_THE_DATA keyword to the Init method and a corresponding tag in the class structure. Then in the cleanup method:

```
   if not self.hang_on_dont_destroy_the_data then $
      ptr_free, thedata
```

If I had to justify doing it this way I would say that destruction of objects is often carried out by code that doesn't know much about the object's properties, and this is less true of object-creation code.

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

--

Subject: Re: IDL Shapefile Object
Posted by alt on Fri, 30 Nov 2001 03:29:16 GMT

Kelly Dean <krdean@lamar.colostate.edu> wrote in message
news:<3C06A316.518B7B83@lamar.colostate.edu>...
> I created a little procedure to plot Shapefiles with IDLffShape.
> However, it has a memory leak. Can someone point out a plug to stop my
> memory leak?
>
> The routine works great, but when I have to add graphics to 800 images,
> I run out of memory at around 500 images.
>
> A sample routine is available at ...
>
> ftp://ftp.cira.colostate.edu/Dean/teststate.pro
>
> Kelly Dean
> CSU/CIRA

Quotation from IDLffShape::GetEntity topic IDL 5.4 help:
"Note - Since an entity structure contains IDL pointers, you must free
all the pointers returned in these structures when the entity is no
longer needed using the IDLffShape::DestroyEntity method. "

It's working.

Altyntsev Dmitriy
alt@iszf.irk.ru
Remote Sensing Center, ISTP
Irkutsk, Russia
http://ckm.iszf.irk.ru

Subject: Re: IDL Shapefile Object
Posted by Kelly Dean on Fri, 30 Nov 2001 19:52:29 GMT

Thanks gentlemen,

Using the undocumented feature "HEAP_FREE, pEnts" solved my problem.

Kelly

Kelly Dean wrote:

> I created a little procedure to plot Shapefiles with IDLffShape.
> However, it has a memory leak. Can someone point out a plug to stop my
> memory leak?
>

> The routine works great, but when I have to add graphics to 800 images,
> I run out of memory at around 500 images.
>
> A sample routine is available at ...
>
> ftp://ftp.cira.colostate.edu/Dean/teststate.pro
>
> Kelly Dean
> CSU/CIRA

Subject: Re: IDL Shapefile Object
Posted by mvukovic on Mon, 03 Dec 2001 15:15:02 GMT
View Forum Message <> Reply to Message

David Fanning <david@dfanning.com> wrote in message
news:<MPG.1670836a9029670e9897a1@news.frii.com>...
> Mark Hadfield (m.hadfield@niwa.cri.nz) writes:
>
>> Hey, I didn't know you *could* supply arguments to OBJ_DESTROY (though I
>> should have known because it's right there in the OBJ_DESTROY
>> documentation). Has anyone actually written code that *uses* this feature.
>> And if so, why? It seems to me that when you tell an object to destroy
>> itself, then it's up to the object to know how to do it.
>
> I've never used it. (Guess I should make a habit
> of reading the documentation that *is* there!)
> But I can imagine a case for it.
>
> Suppose one of the fields for the object was a
> pointer to some image data. The same image pointer
> might be present in several objects (to save
> copying the huge image). Any decent object cleanup
> routine would certainly free the pointer, but maybe
> you don't want it destroyed because then the other
> objects that are using it wouldn't work properly.
>
> In this case a HANG_ON_DONT_DO_IT keyword on the cleanup
> method might be appropriate.
>
> Cheers,
>
> David

An object should know what heap variables it created, and thus, only
destroy those heap variables (that the object itself created). All
other heap variables that were passed to it from the outside should

not be within its responsabilities.

Mirko