Subject: Re: Recursive Objects
Posted by Dick Jackson on Tue, 09 Oct 2001 15:32:33 GMT
View Forum Message <> Reply to Message

"Francesco Belletti" <guenhwyvar@libero.it> wrote in message
news:e7b4ed30.0110090547.79d8e345@posting.google.com...
> I need to store a object reference in an object structure:
>
> pro my_obj__define
> struct = {my_obj, another_obj:OBJ_NEW()}
> end
>
> After another_obj is zeroed during object creation, it couldn't no
> more be used as an object reference!!
> I've tried to redeclare it
>
> another_obj=OBJ_NEW()
>
> but the problem remains.
> It's an IDL bad limit, or my error?

Inside a my_obj method definition, you should refer to this object as

self.another_obj = OBJ_NEW()

That should work fine.
--
-Dick

Dick Jackson / dick@d-jackson.com
D-Jackson Software Consulting / http://www.d-jackson.com
Calgary, Alberta, Canada / +1-403-242-7398 / Fax: 241-7392

Subject: Re: Recursive Objects
Posted by Karl Schultz on Tue, 09 Oct 2001 15:49:04 GMT
View Forum Message <> Reply to Message

"Francesco Belletti" <guenhwyvar@libero.it> wrote in message
news:e7b4ed30.0110090547.79d8e345@posting.google.com...
> Hello,
> I'm a very beginner with IDL so I'm sorry if my question is too
> stupid.
> I need to store a object reference in an object structure:
>
> pro my_obj__define
> struct = {my_obj, another_obj:OBJ_NEW()}
> end

This is correct. OBJ_NEW() just puts a null object ref in your struct,
which is what you want at this point.

> After another_obj is zeroed during object creation, it couldn't no
> more be used as an object reference!!
> I've tried to redeclare it
>
> another_obj=OBJ_NEW()

This just puts another null object ref in your structure.

You probably want to create this object in your ::Init method:

```
function my_obj::Init
  self.another_obj = OBJ_NEW('SomeOtherClass')
  return, 1
end
```

Your class "my_obj" is creating an instance of some other object of class
"SomeOtherClass" when it initializes, presumably because your class "my_obj"
needs the services of "SomeOtherClass". Depending on what your "my_obj"
class does, you can instead create the object of class "SomeOtherClass" at
some other time than in the ::Init method, but that all depends on when you
need it.

Also note that you do not need to explicitly destroy this instance of
"SomeOtherClass" in the my_obj::Cleanup method. Since the objref is in the
class struct, IDL will find and destroy it for you.

Karl

Subject: Re: Recursive Objects
Posted by Martin Downing on Tue, 09 Oct 2001 15:51:15 GMT
View Forum Message <> Reply to Message

"Francesco Belletti" <guenhwyvar@libero.it> wrote in message
news:e7b4ed30.0110090547.79d8e345@posting.google.com...
> Hello,
> I'm a very beginner with IDL so I'm sorry if my question is too
> stupid.
> I need to store a object reference in an object structure:
>
> pro my_obj__define
>   struct = {my_obj, another_obj:OBJ_NEW()}
> end
>

> After another_obj is zeroed during object creation, it couldn't no
> more be used as an object reference!!
> I've tried to redeclare it
>
> another_obj=OBJ_NEW()
>
> but the problem remains.
> It's an IDL bad limit, or my error?
>
> Thank you,
>
> Francesco Belletti

Francesco,
Objects are handled using OBJREFs, which are very much like pointers. What you have declared is a reference to an object, ie an OBJREF, which you have set to the NULLOBJ. What you need to do next is to point variable to a valid object, eg in your INIT method:

```
-----------------------------------------
function my_obj::init, other = other
 ; see if other is a valid object
 if obj_valid(other) then begin
   ; ok so set pointer to it
   self.another_obj = other
   print, "setting another"
 endif
 return, 1
end

function my_obj::GET_ANOTHER
 return, self.another_obj
end

pro my_obj__define
   struct = {my_obj, another_obj:OBJ_NEW()}
end
-----------------------------------------------
```

Now when you create some objects:

```
IDL> obj_a = obj_new('my_obj')

IDL> obj_b = obj_new('my_obj',other =obj_a)

setting another

IDL> help, obj_a
```

OBJ_A OBJREF = <ObjHeapVar1(MY_OBJ)>

IDL> help, obj_b

OBJ_B OBJREF = <ObjHeapVar2(MY_OBJ)>

IDL> help, obj_b->get_another()

<Expression> OBJREF = <ObjHeapVar1(MY_OBJ)>

The second call to obj_new has not created another copy of obj_a but has stored its address (OBJREF) within obj_b.

Hope this helps, good luck with IDL!

Martin

Subject: Re: Recursive Objects
Posted by Mark Hadfield on Tue, 09 Oct 2001 20:06:00 GMT
View Forum Message <> Reply to Message

From: "Karl Schultz" <kschultz@researchsystems.com>
>> pro my_obj__define
>> struct = {my_obj, another_obj:OBJ_NEW()}
>> end
> ....
> Also note that you do not need to explicitly destroy this instance of
> "SomeOtherClass" in the my_obj::Cleanup method. Since the objref is in the
> class struct, IDL will find and destroy it for you.

Not in my experience!

When IDL destroys a my_obj instance it will erase an object reference stored in the my_obj class structure. It will *not* destroy the heap variable that this reference refers to. So to avoid a memory leak you need

```
pro my_obj::cleanup
   obj_destroy, self.another_obj
   ; Other cleanup tasks
end
```

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: Recursive Objects
Posted by David Fanning on Tue, 09 Oct 2001 20:15:51 GMT
View Forum Message <> Reply to Message

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

> From: "Karl Schultz" <kschultz@researchsystems.com>
>>> pro my_obj__define
>>> struct = {my_obj, another_obj:OBJ_NEW()}
>>> end
>> ....
>> Also note that you do not need to explicitly destroy this instance of
>> "SomeOtherClass" in the my_obj::Cleanup method. Since the objref is in
> the
>> class struct, IDL will find and destroy it for you.
>
> Not in my experience!
>
> When IDL destroys a my_obj instance it will erase an object reference stored
> in the my_obj class structure. It will *not* destroy the heap variable that
> this reference refers to. So to avoid a memory leak you need
>
> pro my_obj::cleanup
> obj_destroy, self.another_obj
> ; Other cleanup tasks
> end
>

I've thought all day long that this guy was really
trying to use an INHERITS in his structure definition,
not another structure definition. I'm sure this is what
Karl was thinking, too, in his reply.

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: Recursive Objects
Posted by David Fanning on Tue, 09 Oct 2001 20:17:24 GMT
View Forum Message <> Reply to Message

David Fanning (david@dfanning.com) writes:

> I've thought all day long that this guy was really
> trying to use an INHERITS in his structure definition,
> not another structure definition. I'm sure this is what
> Karl was thinking, too, in his reply.

Whoops! Of course, I meant "object" definition.

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: Recursive Objects
Posted by Pavel A. Romashkin on Tue, 09 Oct 2001 21:03:42 GMT
View Forum Message <> Reply to Message

David Fanning wrote:

> Whoops! Of course, I meant "object" definition.

Obviously, although David has been silent on the Mac issue, he is a
little shaken, too. Imagine if the only truly supported Windows will be ME?

Cheers,
Pavel

Subject: Re: Recursive Objects
Posted by guenhwyvar on Wed, 10 Oct 2001 08:09:52 GMT

Thank you very much for all the answers! Now my program work well!

Francesco Belletti

Subject: Re: Recursive Objects
Posted by Karl Schultz on Wed, 10 Oct 2001 14:52:01 GMT

"Mark Hadfield" <m.hadfield@niwa.cri.nz> wrote in message
news:001501c150fd$d00b6440$d938a8c0@Hadfield...
> From: "Karl Schultz" <kschultz@researchsystems.com>
>>> pro my_obj__define
>>>   struct = {my_obj, another_obj:OBJ_NEW()}
>>> end
>> ....
>> Also note that you do not need to explicitly destroy this instance of
>> "SomeOtherClass" in the my_obj::Cleanup method. Since the objref is in
> the
>> class struct, IDL will find and destroy it for you.
>
> Not in my experience!
>
> When IDL destroys a my_obj instance it will erase an object reference
stored
> in the my_obj class structure. It will *not* destroy the heap variable
that
> this reference refers to. So to avoid a memory leak you need
>
> pro my_obj::cleanup
>   obj_destroy, self.another_obj
>   ; Other cleanup tasks
> end

Oops, you are right. The object I was thinking about was a container object
(inherits from a container object) where the Init method added an object to
the container right after creating it. So, when the container gets
destroyed, the added object gets destroyed too. Sorry about that.