Subject: Re: object argument passing behaviour changed in v8.2.2?
Posted by Paul Van Delst[1] on Tue, 22 Oct 2013 13:03:49 GMT
View Forum Message <> Reply to Message

O.k. I think I've figured out a workaround -- but I would still like to
know why the test case I posted doesn't work.

I went through our repository logs and saw that I recently added the
creation of a new object in the method.

So, if I remove the line

```
   obj = obj_new('IDLitComponent')
```

from the IDLitComponent::test_pass_objelement method, everything works
as expected:

```
IDL> test_pass
% Compiled module: TEST_PASS.
In method. Name = scalar work obj
In caller. Name = scalar work obj

In method. Name = scalar work obj
In caller. Name = scalar work obj

In method. Name = scalar work obj
In caller. Name = scalar work obj

In method. Name = scalar work obj
In caller. Name = scalar work obj

In method. Name = scalar work obj
In caller. Name = scalar work obj
```

So, are we dealing with pass-by-reference/value confusion on my part here?

That is, the creation of new object in the method doesn't do anything
about the reference to the original object that was passed in, and
eventually returned back to the caller?

cheers,

paulv

On 10/22/2013 08:16 AM, Paul van Delst wrote:
> Hi Chris,
>
> Your test code works fine for me. But I changed it to more closely

```
> approximate what I am doing (and added more output of the properties).
>
> The important change is that the procedure in question has been changed
> to a method.
>
> -----%<-----
> pro IDLitComponent::test_pass_objelement, obj
>    obj = obj_new('IDLitComponent')
>    self->GetProperty, name=name
>    obj->SetProperty, NAME=name       ; Set the name...
>    obj->GetProperty, NAME=name       ; ...get it...
>    print, "In method. Name = ", name  ; ...and print
> end
>
> pro test_pass
> o = objarr(5)
> for i=0,4 do begin
>    scalarobj = obj_new('IDLitComponent',name='scalar work obj')
>    o[i]      = obj_new('IDLitComponent',name='empty rank-1')
>    scalarobj->test_pass_objelement, o[i]
>    o[i]->getproperty, name = name     ; Get the name...
>    print, "In caller. Name = ", name  ; ...and print
>    print
> endfor
> end
> -----%<-----
>
> When I run "test_pass" I get the following output:
>
> IDL> test_pass
> % Compiled module: TEST_PASS.
> In method. Name = scalar work obj
> In caller. Name = empty rank-1
>
> In method. Name = scalar work obj
> In caller. Name = empty rank-1
>
> In method. Name = scalar work obj
> In caller. Name = empty rank-1
>
> In method. Name = scalar work obj
> In caller. Name = empty rank-1
>
> In method. Name = scalar work obj
> In caller. Name = empty rank-1
>
>
> So, while the *passed* object is valid upon return, all of the changes
```

> made to it in the method didn't "take".
>
> This is only a problem when the object is an argument to one of its own
> methods, but invoked via a different object (in this case "scalarobj").
>
> When the object is passed as an argument to a "regular" procedure (as in
> your test case) everything works as expected.
>
> Any ideas? The actual code in question (my operational equivalent of the
> main "test_pass") has existed since 2011 - and it's been used to process
> data for several satellite sensors.
>
> cheers,
>
> paulv
>
>
> On 10/21/2013 07:07 PM, Chris Torrence wrote:
>> Hi Paul,
>>
>> Nothing has changed with the way IDL passes objects. However, I'm a
>> little confused by your code. When you say that the
>> "compute_interpolation_frequency" procedure "allocates the resulting
>> object", do you really mean that it just fills in some properties on
>> that object? Because it looks like you are doing an obj_new on those
>> objects before passing them in.
>>
>> It looks like something strange is going on with garbage collection,
>> where it is somehow freeing up your object inside
>> compute_interpolation_frequency. However, I can't imagine why this
>> would be happening. I just create a test program which approximates
>> what you are doing:
>>
>> pro test_pass_objelement, obj obj->getproperty, name = name
>> obj->SetProperty, NAME='NewName' end o = objarr(5) for i=0,4 do
>> begin o[i] = obj_new('IDLitComponent', NAME=STRTRIM(i,2))
>> test_pass_objelement, o[i] print, obj_valid(o[i]) endfor end
>>
>> When I run this code (at least in IDL 8.3), the objects are all valid
>> after the procedure call. Can you try running this code to make sure
>> it passes for you? If it does, then maybe you can post the details of
>> your compute_interpolation_frequency procedure, so we can diagnose
>> what is happening inside.
>>
>> Thanks! -Chris ExelisVIS
>>