Subject: Saved objects.
Posted by Imanol Echave on Tue, 08 Sep 1998 07:00:00 GMT
View Forum Message <> Reply to Message

Hi people:

When you create a new object with OBJ_NEW, IDL searchs automatically the file object__define.pro and compiles it, but... What do you have to do when you RESTORE an object? The object definition and methods aren't compiled, and an error is raised when you use the object. I can compile the file "manually", but ï¿½do you know a more ellegant method?.

Subject: Re: Saved objects.
Posted by davidf on Tue, 08 Sep 1998 07:00:00 GMT
View Forum Message <> Reply to Message

Imanol Echave (ccaeccai@sc.ehu.es) writes:

> When you create a new object with OBJ_NEW, IDL searchs automatically the file
> object__define.pro and compiles it, but... What do you have to do when you
> RESTORE an object? The object definition and methods aren't compiled, and an
> error is raised when you use the object. I can compile the file "manually", but

Here is an edited version of a newsgroup exchange on saving and restoring objects that J.D. Smith and I carried on recently. It describes a manual way of compiling the object's methods, but I don't think there is any way around this, short of putting each method routine in its own file.

Interestingly, I just ran into a project this week where it is important that we save the graphical output in a database along with the information we used to construct the plot. It must be stored in such as way that we can reconstruct the exact graphic window. I am thinking of implementing this functionality as a Save/Restore object.

Cheers,

David

---

Subject: Objects, File Names, and the Save command.
From: "J.D. Smith" <jdsmith@astrosun.tn.cornell.edu>

I am exploring a very promising use of the save/restore commands in conjunction with objects. Given some complex object which contains a host of different types of data (with pointers, etc.), as part of a class method, one adds:

save, self, FILENAME=fname

to register on disk an accurate snapshot of the object. To restore, later, use:

```
 restore,pname,RESTORED_OBJECTS=obj,/RELAXED_STRUCTURE_ASSIGN MENT
```

and the object is in obj, but also brought back as the local variable *self*. I'm not sure the relaxed structure assignment flag works for objects, but I don't see why it wouldn't. So this can be used in two ways ...

1. To allow an object to replace *itself* with another, perhaps older copy (or even an altogether different type of object -- but the utility of self-transmogrifying objects is not yet apparent to me). This works because the implicit self variable is passed by reference (as it has to be). This will lead to at least one unreferenced heap variable unless garbage collection steps are taken, I.e. by saying:

```
 oldself=self
 restore, pname,/RELAXED_STRUCTURE
 obj_destroy,oldself
```

2. To allow a program module to load up an object on the fly, through the obj variable in the above statement (only one should be loaded if only one was saved).

This is all very convenient but leads to the strange situation of a loaded object in memory which exists there *before* any of the class methods, and/or the __define procedure for that object class are compiled. Therefore, the usual paradigm of putting all class methods in the __define procedure file before this procedure (suggested by RSI itself in the manual) fails. How can the method be found if the __define doesn't have to be compiled and isn't in it's own file? I would like to come up with a solution which doesn't involve a separate class__method.pro file for each method. Any ideas?

Thanks,

JD

---

To which I replied like this:

How about something like this:

```
thisClass = Obj_Class(self)
Resolve_Routine, thisClass + '_define'
```

I haven't tested this, but don't see any reason it wouldn't work.
Resolve_Routine is the way IDL procedures and functions can
be compiled from *within* other procedures and functions.

Cheers,

David

---

J.D. replied with this:

This will certainly work, but has the unfortunate side-effect of
re-compiling every method each time an object is read from disk... I
thought of modifying it slightly to the tune of:

```
thisdef=Obj_Class(self)+'__DEFINE'
if (where(routine_info() eq thisdef))[0] eq -1 then
resolve_routine,thisdef
```

So that it would only compile if presently undefined.

JD

---

---

Subject: Re: Saved objects.
Posted by davidf on Wed, 09 Sep 1998 07:00:00 GMT
View Forum Message <> Reply to Message

Sean (seanr@possys.com) writes:

> I can see one large hole in this type of procedure. What if you are running
> the application on a runtime liscense? In this case, the runtime liscense
> cannot compile any source code, and will fail. Any ideas around that
> limitation?

If you are running runtime licenses you better have compiled
*all* the procedures and functions you are going to need
before you save that baby. This will include any object
definition code file you plan to use in your program.

Runtime programs that restore objects they don't know about
are designed to fail for all kinds of reasons. :-)

Cheers,

David

---
David Fanning, Ph.D.
Fanning Software Consulting
E-Mail: davidf@dfanning.com
Phone: 970-221-0438, Toll-Free Book Orders: 1-888-461-0155
Coyote's Guide to IDL Programming: http://www.dfanning.com/

---

Subject: Re: Saved objects.
Posted by seanr on Wed, 09 Sep 1998 07:00:00 GMT
View Forum Message <> Reply to Message

> How about something like this:
>
>  thisClass = Obj_Class(self)
>  Resolve_Routine, thisClass + '_define'
>
> I haven't tested this, but don't see any reason it wouldn't work.
> Resolve_Routine is the way IDL procedures and functions can
> be compiled from *within* other procedures and functions.
>
> Cheers,
>
> David
>
> ----------
>
> J.D. replied with this:
>
> This will certainly work, but has the unfortunate side-effect of
> re-compiling every method each time an object is read from disk... I
> thought of modifying it slightly to the tune of:
>
> thisdef=Obj_Class(self)+'__DEFINE'
> if (where(routine_info() eq thisdef))[0] eq -1 then
> resolve_routine,thisdef
>
> So that it would only compile if presently undefined.
>
> JD
> ---

>

I can see one large hole in this type of procedure. What if you are running the application on a runtime liscense? In this case, the runtime liscense cannot compile any source code, and will fail. Any ideas around that limitation?

Also, not in the quoted text, you were wondering if the /RELAXED_STRUCTURE_ASSIGNMENT works for objects. It does...sortof. I'm currently working with RSI tech support on an apparent bug when you save an object with nested structures in the self structure, then add/change a field in one of the nested structures, and attempt to restore it. The restore will respond with the verbose message from restore:

RESTORE, FILENAME = getfile, RESTORED_OBJECTS=obj_restore, /VERBOSE, /RELAXED_STRUCTURE_ASSIGNMENT

% RESTORE: Portable (XDR) SAVE/RESTORE file. % RESTORE: Save file written by user@Microsoft Windows Host, Fri Aug 28 09:04:00 1998. % RESTORE: IDL version 5.1 (Win32, x86). % RESTORE: Recovering incompatible definition of structure OUTPUTDATA using relaxed structure assignment rules. % RESTORE: Recovering incompatible definition of structure DIME_PROJECT using relaxed structure assignment rules. % RESTORE: Restored variable: TEMPPROJ. % RESTORE: Did not expect relaxed structure assignment definition for OUTPUTDATA to be in use.

If I then attempt to reference the restored object, I get the following error:

% XMANAGER: Caught unexpected error from client application. Message follows... % Undefined object class: DIME_PROJECT.

The strange thing is, if I do another restore again, I get the same response from RESTORE, but it succeeds and does not fail when I reference the object this time.

If I only make changes to the top level self structure, everything is fine.

Has anyone worked with saved objects that have nested structures and change the nested structure fields between save and restore sessions?

---------- ---------
Sean P. Rumelhart  Positive Systems, Inc.
seanr@possys.com  250 Second St. East
PH:  406.862.7745  Whitefish MT 59937
FAX: 406.862.7759  www.possys.com
---------- ----------