Subject: Re: Objects, File Names, and the Save command.
Posted by J.D. Smith on Thu, 23 Jul 1998 07:00:00 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
>
> J.D. Smith (jdsmith@astrosun.tn.cornell.edu) writes:
>
>> I am exploring a very promising use of the save/restore commands in
>> conjuction 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*.
>
>> [cut...]
>
>> 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?
>
> 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.
>

David,

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.

Thanks for the tip.

JD

```
--
 J.D. Smith                             |*|      WORK: (607) 255-5842
 Cornell University Dept. of Astronomy  |*|            (607) 255-6263
 304 Space Sciences Bldg.               |*|       FAX: (607) 255-5875
 Ithaca, NY 14853                       |*|
```

Subject: Re: Objects, File Names, and the Save command.
Posted by davidf on Thu, 23 Jul 1998 07:00:00 GMT

View Forum Message <> Reply to Message

J.D. Smith (jdsmith@astrosun.tn.cornell.edu) writes:

```
>  I am exploring a very promising use of the save/restore commands in
>  conjuction 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*.

>  [cut...]

>  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?

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

David Fanning, Ph.D.
Fanning Software Consulting
E-Mail: davidf@dfanning.com
Phone: 970-221-0438
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Objects, File Names, and the Save command.
Posted by mvukovic on Fri, 24 Jul 1998 07:00:00 GMT

View Forum Message <> Reply to Message

In article <35B769F7.73FC0139@astrosun.tn.cornell.edu>,
  "J.D. Smith" <jdsmith@astrosun.tn.cornell.edu> wrote:
> David Fanning wrote:
>>
>> J.D. Smith (jdsmith@astrosun.tn.cornell.edu) writes:
>>
>>> I am exploring a very promising use of the save/restore commands in
>>> conjuction 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*.
>>
>>> [cut...]
>>
>>> 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?
>>
>> 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.
>>
>
> David,
>
> 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.
>
> Thanks for the tip.
>
> JD
>
> --
>   J.D. Smith |*| WORK: (607) 255-5842
It would be nice for IDL to do it by itself. i.e., if it finds calling
an object's routines, and it is undefined, first compile "name"__define,
or if that fails, "name"__"routine_name".

This is becoming reminiscent of emacs lisp (my current learning project),

where you can specify dependencies,
and they get resolved (compiled) automatically as needed.

But to congratulate JD on his idea on "generic" writing and reading
an object from file. I had a need for that and was going to do a fairly
complex read/write routine.

But now, can you maybe save the object, and associated routines? This
complicates the saving procedure, as you need to invoke IDL, compile the
routines, create the object, and save before doing anything else.

But at least on PC's (NT) it is possible to invoke two IDL sessions at the
same time. Thus it may be possible to write up a routine to which you pass
the object name, it creates it, and compiles all the methods and saves. A
batch job to invoke an IDL session to execute that routine would then do the
trick fairly transparently.

mirko