Subject: Restored Object Method Compilation
Posted by J.D. Smith on Sun, 04 Oct 1998 07:00:00 GMT
View Forum Message <> Reply to Message

Object Programmers,

I have recently discovered a problem with the object method compilation technique discussed earlier in this newsgroup by David Fanning and myself. This discussion is archived on David's IDL page (http://www.dfanning.com/tips/saved_objects.html). The technique discussed basically allows for the compilation of uncompiled methods for objects restored from file when the entire set of methods resides in the file class__define.pro. The alternative (unacceptable in my opinion) is to give each method its own file of the form class__method.pro

The problem relates to superclasses. The normal IDL implicit compilation of an object's methods (contained in class__define.pro), as when invoking obj_new('class'), proceeds up the inheritance tree, compiling and defining all of the superclasses which are not yet defined. Our technique does not compile superclasses, and so our methods are broken if they fail to override, or chain up to a superclass method.

One might think that all you would need to do is call the class__define procedure for the class of the restored object, and all superclasses would be defined and compiled as encountered. This works in any context but this one. The reason why is that a restored object contains implicitly in its definition the class structure definitions of all its superclasses. Therefore, when class__define is called, it doesn't call or compile any of the superclass__define definitions. As far as IDL is concerned, "superclass" is already a valid class (struct) and nothing further need be done.

The only solution is to proceed up the inheritance tree compiling by hand. A procedure, resolve_obj, which accomplishes this, is attached. Note that the recursion must be breadth first as coded... that is, a class must be compiled before its superclasses (because how would we know which superclasses it has otherwise). The way to call this routine on a restored object is:

resolve_obj,obj

Alternatively, if you have the name of a class rather than an object instance, you could do:

resolve_obj,CLASS=class

and the recursion would proceed in the same way. To reiterate, this

procedure is necessary only when an class has been *defined* but not *compiled*.

JD

```
pro resolve_obj,obj,CLASS=class,ROUTINE_INFO=ri
  if n_params() ne 0 then begin
    if NOT obj_valid(obj) then begin
      message,'Object is not valid.'
    endif
    class=obj_class(obj)
  endif

  if n_elements(ri) eq 0 then ri=routine_info()

  for i=0,n_elements(class)-1 do begin
    defpro=class[i]+'__DEFINE'
    if (where(ri eq defpro))[0] eq -1 then resolve_routine,defpro
    supers=obj_class(class[i],/SUPERCLASS,COUNT=cnt)
    if cnt gt 0 then resolve_obj,CLASS=supers,ROUTINE_INFO=ri
  endfor
end
```

## File Attachments

1) resolve_obj.pro, downloaded 138 times