Subject: Re: Function referencing/automatic defintion question.
Posted by Paul Van Delst[1] on Thu, 29 May 2003 16:12:51 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
>
> Paul van Delst (paul.vandelst@noaa.gov) writes:
>
>> So my question is: what's the go here? Why doesn't my calling procedure "see" the compiled
>> functions that precede my structure definition? I thought the whole point of sticking
>> these routines *before* the procedure in my emiscoeff__define.pro file that actually does
>> the definition meant that they would be compiled?
>>
>> Any insights appreciated,
>>
>> paulv
>>
>> p.s. When I manually compile the emiscoeff__define.pro file I get the following:
>>
>> IDL> .run emiscoeff__define
>> % Compiled module: ASSOCIATED_EMISCOEFF.
>> % Compiled module: DESTROY_EMISCOEFF.
>> % Compiled module: ALLOCATE_EMISCOEFF.
>> % Compiled module: ASSIGN_EMISCOEFF.
>> % Compiled module: COUNT_EMISCOEFF_SENSORS.
>> % Compiled module: EMISCOEFF__DEFINE.
>>
>> How come I don't get this list when I do the automatic compilation via
>>
>> EmisCoeff = { EmisCoeff }
>>
>> ???
>
> Having the function in front of the object definition
> module is a necessary, but not sufficient (at least in
> this case) condition for getting it to compile correctly. :-)
>
> The problem (almost certainly) is that a program
> module that *calls* this function is being compiled
> before the function is compiled.

Umm...I'm not sure exactly what you mean. I have function,
Compute_Emissivity_Coefficients() that calls another function,
Compute_Theta_Coefficients(), which calls the EmisCoeff__Define procedure via the
structure definition,
EmisCoeff = { EmisCoeff }
and then calls the Allocate_EmisCoeff() function (which resides in the
emiscoeff__define.pro source file *in front* of the EmisCoeff__Define procedure.

My apparently mistaken understanding is that the simple act of doing:
  EmisCoeff = { EmisCoeff }
will automatically compile Allocate_EmisCoeff() and make it available in the current scope of the Compute_Theta_Coefficients() function (at the very least)

And, at the point where the function in question is called, it *has* already been compiled. If I print out a list of the resolved functions *immediately* prior to the Allocate_EmisCoeff function call, it's in the list:

```
IDL> .reset_session
IDL>  print, compute_emissivity_coefficients( 'test_sensor_emissivity.nc', EmisCoeff,
/pause)
% Compiled module: COMPUTE_EMISSIVITY_COEFFICIENTS.
% Compiled module: VALID_STRING.
% Compiled module: READ_NCDF.
% Compiled module: IS_NCDF.
% Compiled module: EMISCOEFF__DEFINE.
```

Printing the resolved function output from ROUTINE_INFO:

```
ALLOCATE_EMISCOEFF ASSIGN_EMISCOEFF ASSOCIATED_EMISCOEFF
CHECK_VECTORS
COMPUTE_EMISSIVITY_COEFFICIENTS COMPUTE_EMISSIVITY_FIT
COMPUTE_THETA_COEFFICIENTS
CONVERT_STRING
DESTROY_EMISCOEFF IS_NCDF MPCURVEFIT READ_NCDF SPLINE UNIQ VALID_STRING
% COMPUTE_THETA_COEFFICIENTS: Variable is undefined: ALLOCATE_EMISCOEFF.
% COMPUTE_EMISSIVITY_COEFFICIENTS: Error computing emissivity vs. theta fit coefficients.
       -1
IDL>
```

Note that ALLOCATE_EMISCOEFF is in the list.

> You could solve this problem in several ways. (1) Take
> the function out of this file and put it in a file of
> its own. (2) Make the function a method of the object.
>
> I think solution 2 is probably the better one in this case,
> since the function is obviously related to the object in
> a tight way. (In fact, I can't see why *all* of these modules
> aren't object methods. Do you have a reason for this that is
> not apparent to me?)

Because I don't want this project to descend into a object programming exercise. I like data encapsulation, but data hiding that requires get and set functions is just too much overhead for what I want to do (to say nothing of the terribly confusing [to me at least] syntax that uses "->"). From my point of view my named structure EmisCoeff *is* an

"object". But it has public, rather than private, components.

At any rate, I just want to get my numbers and write them to a file so I can use my Fortran code to do something useful. The worst thing I did here was go from doing an "inline" structure definition to (what I thought would be) the more natty method of automatic structure defn.

> But if you want to keep it the way it is, I would just move
> this function to the top of the file, or add a FORWARD_FUNCTION
> statement in the module that uses it.

Thanks very much for the FORWARD_FUNCTION tip. That worked....but I don't understand in the least why it should be necessary.

cheers,

paulv

--
Paul van Delst
CIMSS @ NOAA/NCEP/EMC
Ph: (301)763-8000 x7748
Fax:(301)763-8545