Subject: Re: Function referencing/automatic defintion question.
Posted by Paul Van Delst[1] on Fri, 30 May 2003 16:29:41 GMT
View Forum Message <> Reply to Message

"Robert S. Hill" wrote:
>
> Paul van Delst <paul.vandelst@noaa.gov> writes:
>> Please...interject. This is driving me nuts (can't you tell :o)
>
> Well, then -- emboldened, I press on.
>
>> My understanding is that when I _run_ the routine containing the snippet
>> above it gets to the line where the structure is defined and _compiles_
>> all the routines in the source file emiscoeff__define.pro. After that
>> my assumption is that all of those emiscoeff__define.pro contained
>> routines are available for use in the current scope, i.e. in the
>> routine that calls Allocate_EmisCoeff().
>
> Just to be clear, here is more detail on what I think is probably
> happening. I'm assuming here that your calling code is from a main
> level program that you run using the .run command.

Nope. I never do that. My calling code is itself a function that I invoke from the main level thusly:

IDL> print, compute_emissivity_coefficients('test_sensor_emissivity.nc', EmisCoeff)

> The .run command doesn't interpret your calling program line by line.
> Instead, it compiles it all into bytecode (or whatever RSI calls it),
> then executes it. During this execution, the execution engine arrives
> at your structure invocation with the curly braces, and it then invokes
> the compiler to compile all the routines in the __define file.
> Subsequently, the execution engine reaches the Allocate_EmisCoeff()
> invocation, but this has *already* been compiled as an array, so it
> doesn't recognize it as a function (an array and function of the same
> name can coexist happily).

I'm thinking that something like this is happening but I don't understand exactly why. My assumption has always been that once a routine has been compiled by default (i.e. it precedes the routine that a source code file is named after in the file - your "inner" routine) then that routine is accessible in all and any subsequent procedure/function independent of their heirarchy. I think that assumption is flawed. My working assumption now is that the "inner" routines in an IDL source code file are really only accessible to the "outer" routine in a source file (i.e. with the same name as the file itself.) *unless* you do something like the compile_opt strictarr or forward_function thingo.

> Although compile time and run time are not globally separated as in

> Fortran or C, they are separate for each routine, including any main
> level script. Even when you put a bunch of routines in one file, you
> need to be aware of the dependence hierarchy of any of them that are
> functions, and put the inner ones higher up in the file.

This I religiously do so...

> (Or use strictarr or forward_function.)

I've never needed these before.

The germ of a resolution is forming in my mind. I'm going to test some stuff when I get some time next week. Thanks very much.

When I finally figure this out I just *know* everybody else will say "Well,...yeah, of course - why would you think it would work the other way?" :o)

paulv

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