Subject: Re: Function referencing/automatic defintion question.
Posted by Robert.S.Hill.1 on Fri, 30 May 2003 15:22:07 GMT
View Forum Message <> Reply to Message

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.

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).

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. (Or use strictarr or forward_function.)

-Bob Hill