Subject: SOLVED. Re: Function referencing/automatic defintion question.
Posted by Paul Van Delst[1] on Fri, 30 May 2003 18:02:03 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
>
> Paul van Delst (paul.vandelst@noaa.gov) writes:
>
>> 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)
>
> I'm already thinking that, but--of course--I haven't
> seen the solution. :-)

First off thanks to Bob Hill and another informed source for allowing the light to shine through a crack in the dome. Second, thanks to clip readers for putting up with me. Ehem. :o)

The main stumbling block for me was the definition of run- and compile- time. I never really separated them for IDL stuff.

Let's say I have some IDL code like so:

```
PRO test_stuff, EmisCoeff
  EmisCoeff = { EmisCoeff }
  result = Allocate_EmisCoeff( 2, 3, 4, EmisCoeff )
END
```

where the Allocate_EmisCoeff() is in the file emiscoeff__define.pro before the actual EmisCoeff__Define procedure.

When I invoked my main function,

```
IDL> test_stuff, e
```

This *compiles* test_stuff.pro into intermediate p-code (what Robert Hill called bytecode in his post). When the line "result = Allocate_EmisCoeff()" is reached during the compilation, IDL has to make a decision as to whether this is a function call or an array reference using () instead of []. It does this based on the syntax alone. Because my function call does not contain any keywords -- which would be a giveaway that it's a function -- IDL does the easiest(fastest) thing and assumes it's an array reference and creates the appropriate p-code. So, when the code is actually *run*, the p-code for the line of code "result = Allocate_EmisCoeff()" is for an array reference _despite_ the fact that the function Allocate_EmisCoeff() itself is compiled via the automatic structure creation. And sure enough, the result is:

```
% Compiled module: TEST_STUFF.
% Compiled module: EMISCOEFF__DEFINE.
% Variable is undefined: ALLOCATE_EMISCOEFF.
% Execution halted at: TEST_STUFF          5 test_stuff.pro
%                      $MAIN$
IDL>
```

And thus it's apparent why the COMPILE_OPT STRICTARR solves the problem. It's an extra syntax hint IDL uses in the compilation phase.

All this may be clear as crystal to you IDL gurus out there, but it sure wasn't to me. Robert's email was the first eureka event identifying the difference between compile and run time in IDL. And a patient phone call from another smart feller was the clincher.

Everybody have a couple beers after work today. Tell 'em Paul says it's o.k. :o)

Thanks,

paulv

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