Subject: Re: Hide functions from other procedures
Posted by fugu on Tue, 03 Mar 2009 17:03:33 GMT
View Forum Message <> Reply to Message

On Feb 28, 10:56 am, Reimar Bauer <R.Ba...@fz-juelich.de> wrote:
> mgalloy schrieb:
>
>
>
>> fugu wrote:
>>> I have two procedures, and in both .pro files, I define slightly
>>> different functions with the same name. I would like to make sure,
>>> that only the procedure in the same .pro file can see 'it's own'
>>> function, but no other procedure. The reason is, that I often define a
>>> plotfunctionfirst (which I call my_plot), which gets than called by
>>> the actual procedure several times, to plot on screen and to ps etc.
>
>> IDL has a global namespace for routines. There is no way to ensure that
>> only the procedure in the same .pro file can see 'it's own'function
>> except by carefully managing manual re-compiles of the routines (as
>> successive compiles bump old routines of the same name out of memory).
>
>>> I can off course, give the functions different names, but there are
>>> good reasons for calling it the same (mainly because I later know the
>>> functioncall my_plot without having to look it up etc.)
>
>>> As an example:
>
>>> I have two files, called test1.pro and test2.pro, which hold the
>>> procedures (not surprisingly) test1 and test2.
>
>>> test1.pro looks like this
>
>>> HEADER
>>> FUNCTIONmy_plot
>>> ...
>>> END
>
>>> PRO TEST1
>>> ...
>>> calls my_plot
>>> END
>
>>> and test2 looks exactly the same, but thefunctionmy_plot in test2 is
>>> different from thefunctionmy_plot in test1. Now I hoped, that
>>> compiling and running test2 would know nothing about the my_plot
>>> functionin test1 which I compiled before. But that does not seem to
>>> be the case.

>
>> So starting from a fresh IDL session, compiling TEST1, then compiling
>> TEST2, and then calling MY_PLOT results in TEST1's MY_PLOT being called?
>> I don't see how that's possible. Be careful with compile order: it
>> happens only the first time you use afunction. So calling TEST2, then
>> TEST1, and then TEST2 again would cause a problem (TEST2 would be
>> calling TEST1's MY_PLOT). Of course, using RESOLVE_ROUTINE, manually
>> compiling, doing a .reset, etc. changes everything.
>
>> My suggestion: name them TEST1_MY_PLOT and TEST2_MY_PLOT.
>
>> Mike
>
> Or do refactor this code into objects.
>
> Then you can inherite from one object and overwrite the methods you want
> to have different.
>
> If you want keep your plotfunctionadd cases and keywords to it. So
> that you have only onefunctionas separate routine where you add every
> different part from the others.
>
> cheers
> Reimar

Thanks everyone for helpful comments! Much appreciated.

Daniel