
Subject: Automatic Compilation of IDL Programs, Was: Lost Functions

Posted by [davidf](#) on Tue, 04 Nov 1997 08:00:00 GMT

[View Forum Message](#) <> [Reply to Message](#)

Martin Schultz (mgs@io.harvard.edu) makes some excellent points when he writes:

- > let me just add a little note on separate files vs.
- > "libraries", because I don't think this is mentioned
- > in the IDL manuals: It seems as if the automatic
- > compilation of IDL always compiles up to point where
- > it finds the procedure or function with the same name
- > as the filename.

Martin is mostly right about this, although the information *is* found in the manuals. (I will admit that its reference is a bit obscure for such an important point, however.)

I would like to elaborate on this a moment because you simply would not believe the number of people I run into who do not understand how programs get compiled. Many of these people try to run IDL in the compile-link-run fashion of the good ol' Fortran days. :-)

If you use the IDL executive command `.Compile` to compile a file, all the program modules in the file get compiled. (The executive command `.Run` can also be used this way, although it confuses new IDL users because nothing actually gets `*run*`.) To run the compiled programs, just use their names on the IDL command line or in an IDL program.

But one of the wonderful features of IDL is the ability to have your programs compiled *automatically* when they are needed. This makes it possible to write IDL code that is not cluttered up with a bunch of `INCLUDE` statements, etc. You simply put all of your program files in some directory that is on the IDL path and you don't have to worry whether the files are pre-compiled or not when you write the next IDL program that uses one of these commands.

But the key word here is "command". When you write a program you are, in effect, creating another IDL command. The key rule for getting your programs to compile automatically is to make sure that the "command" module is the *last* program module in a file that has the *same name* as the command you are trying to build (with a `".pro"` extension, of course).

For example, suppose I were trying to write an IDL program

or command named VIEW_IT. What I am trying to do is have something happen when I type the name VIEW_IT at the IDL command line or put that name in an IDL program. But VIEW_IT might be a huge program with lots of subroutines, etc. I might have routines called MAKE_BIG_WINDOW and GET_MY_DATA, etc, etc. that I need to make my VIEW_IT program run.

What I should do is put the MAKE_BIG_WINDOW and GET_MY_DATA and all the rest of the VIEW_IT utility routines in my file *in front of* the VIEW_IT program module. In fact, VIEW_IT should be the *last* module in the file. And I should name the file "view_it.pro".

Now, when I type "view_it" at the IDL command line, IDL looks for a file named "veiw_it.pro" (all lowercase letters on UNIX file systems, no matter *how* you type it on the IDL command line!). It compiles the program modules in that file *until* it gets to the program module with the same name as the file (which, don't you know, is the *last* program module :-), at which point it stops compiling and *runs* that program module automatically.

Now, it sometimes happens that a utility routine like MAKE_BIG_WINDOW becomes a whole lot more useful to you than it was before. In fact, you may want to use it in a new program you are building called SEE_IT. If that is the case, then take MAKE_BIG_WINDOW out of the VIEW_IT code and put it in its own file called "make_big_window.pro". It has now become another IDL command.

Automatic compilation doesn't solve *all* the problems you might run into. (Hence, the Forward_Function command.) But in my experience it solves a lot of them. And it sure beats having to compile all of your program modules before you can do anything in IDL. :-)

Cheers,

David

David Fanning, Ph.D.
Fanning Software Consulting
E-Mail: davidf@dfanning.com
Phone: 970-221-0438
Coyote's Guide to IDL Programming: <http://www.dfanning.com/>
