Subject: Re: HELP: Multiple-file Applications
Posted by steinhh on Wed, 07 Dec 1994 12:13:27 GMT
View Forum Message <> Reply to Message

In article <3c224p$c7h@canopus.cc.umanitoba.ca>, djackson@ibd.nrc.ca (Dick Jackson) writes:

|> % Variable is undefined: PRE_HELPER.
|>
|> ---***--- This is the 'gotcha': when CALLER was compiled, PRE_HELPER
|>   looked
|>   like an array variable, since no function yet existed, I
|> suppose.

[....]

|> So, having multiple routines in a 'subordinate' file, and calling the
|> last one found in there first, will cause all the others to work
|> thereafter, unless there are functions, in which case they'll look like
|> array variables. It's a bit constraining, but if I keep it strictly
|> modular, so only the last pro/function in the 'subordinate' file is
|> called from outside, then I'll be OK.
|>
|> Thanks so far, any other tips? There must be lots of big widget-app
|> builders out there.

The modular approach is a good choice, but of course it could be
quite a bit of work to split a very large file into such modules, if
the program isn't already well organized.

Personally, I always have widget programs looking much like the
this:
File: application.pro

;
Auxiliary event routines ; In order to have a "tidy" application_event routine

pro application_event,event

pro application,parameters

Note that all the "auxiliary event routines" are ONLY called from within
this file -- they have no use what so ever in other applications -- if
they do, I make them into a separate file (one for each multi-use routine).

You should note that when IDL compiles statements like "help,pre_helper(a)",
it looks through the path for any file called "pre_helper.pro", and examines
them for a potential declaration of the function pre_helper(). So, even if

you compile a program referring to a function that's not compiled, you
will avoid the problem you mentioned if it's placed in a file (in the path)
that has the name of the function.

A warning: This also spells trouble if your'e using a variable name
that by coincidence is identical to a function name.

Try e.g.:

```
IDL> vel = 0
IDL> vel(0) = 0  ; This works ok, but during compile-time, the variables
                 ; aren't known that well, so:
IDL> delvar,vel
IDL> vel(0) = 1

vel(0) = 1
 ^
% Syntax error.
```

The lesson is, of course: Don't oversimplify function names, you'l
want to save that for your variables.

Regards,

Stein Vidar