Subject: Re: HELP: Multiple-file Applications
Posted by djackson on Tue, 06 Dec 1994 16:05:45 GMT
View Forum Message <> Reply to Message

James Tappin writes:

> However if your code is a routine rather than a main program then you
> can use CALL_PROCEDURE or CALL_FUNCTION to compile (if not already
> compiled) and execute it (these are more efficient than EXECUTE).

Norbert Hahn writes:

> I played a little with CALL_PROCEDURE and it seems to do what you like.
> I wrote
> sub = 'fader'
> and typed
> call_procedure, sub, out=0.25

[note: I'm not actually worried about variable procedure compilation,
I know what I want to be compiled, but _when/if_ it's compiled
is to be left to run-time. No harm done, it's the same
otherwise.]

> and noticed that fader.pro was found in the search path, compiled and
> executed.
>
> Note that IDL only compiles those procedure that haven't been compiled
> before in this session
>
> and
>
> that IDL only compiles the file including all procedure contained there
> *until* it has reached the END statement of the procedure requested.
>
> Thus, if you have nested procedures, it is best to put the outermost
> procedure (that's the one you call explicitly) at the end of the file.

This sounds good, and I didn't come to the same conclusion, since I
confused myself (and found another 'gotcha' here) by writing these test
routines, caller.pro and helper.pro: (bear with me!)

::::::::::::::
caller.pro
::::::::::::::
```
pro caller
  a = randomu(seed) + 1.0
  help, a, helper(a)
```

```
  pre_pre_helper, a
  help, pre_helper(a)
end
:::::::::::::::
helper.pro
:::::::::::::::
pro pre_pre_helper, x
  print, "In pre_pre_helper, X =", x
end

function pre_helper, x
  return, x+42.4242
end

function helper, x
  return, pre_helper(x)/2.02
end

function post_helper, x
  return, -x
end
```

Then, to test them in a fresh IDL session:

```
IDL> caller
% Compiled module: CALLER.
% Compiled module: HELPER.
A               FLOAT     =       1.23121
<Expression>    FLOAT     =       21.6116
In pre_pre_helper, X =      1.23121

---***--- OK, in CALLER, that one was compiled as a procedure call,
          then the procedure was compiled.

% 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.

% Execution halted at CALLER <caller.pro(   7)> .
%      Called from $MAIN$ .
IDL> help,pre_helper(3)
<Expression>    FLOAT     =       45.4242
IDL> help,post_helper(3)
% Variable is undefined: POST_HELPER.
```

```
% Execution halted at CALLER <caller.pro(   7)> .
%      Called from $MAIN$ .
IDL>
```

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.

Cheers,
-Dick

Dick Jackson djackson@ibd.nrc.ca Institute for Biodiagnostics
Opinions are mine alone. National Research Council Canada, Winnipeg

---