Subject: Re: "include" a file
Posted by Craig Markwardt on Sun, 28 Jun 2009 20:12:58 GMT
View Forum Message <> Reply to Message

On Jun 26, 3:50 pm, JD Smith <jdtsmith.nos...@yahoo.com> wrote:
> Various other languages have the option to include and evaluate the
> program contents of another file at runtime. IDL has the "@"
> operator, but that happens at compile time, so you need to know which
> file to include in advance. I find myself needing to drop small
> "parameter" files in individual directories for a routine to process
> as it crawls through. I could certainly prepare an IDL .sav file, or
> some other data format, parse that, and set-up structures and
> variables as needed, but that makes editing and updating the file very
> painful. What IDL needs is a way to "include" a file directly, and
> evaluate its contents. Finding nothing, I came up with the following
> concept:
>
...
> Do others encounter this problem, and has anyone solved it in a
> different way?

I have encountered this style of problem before, and I have solved it
your way as well. To be honest, I've pretty much always regretting
doing this because I let the "configuration" code get out of hand,
setting extra variables, running little bits of code, etc. I'd want
to call the config code in two different places for two different
purposes, and then it just started getting too complicated.

These days, my preferred way is to turn the "configuration" file into
a real IDL function (or procedure). Then I can compile that and
interogate it directly.

```
  ;; Compile my configuration file
  my_config_file = '/data/run1/config1.pro'
  file_compile, my_config_file, pro_name=mypro

  ;; Interogate my configuration function, in this case retrieve a
struct
  config_struct = call_function(mypro)

  ;; Use config_struct accordingly...
```

It takes very little extra effort to slap a function declaration at
the top of one's file to make something like that work, and it hands
the parsing wizardry to IDL where it belongs.

Here I'm just returning a structure from my configuration function,
but the possibilities are not really limited to that. The procedure

could be used to do actual work rather than just to return a static structure. Or in the case where I wanted to call my config function twice, once for initialization, and once for clean-up, it might be done something like this,

```
  call_procedure, mypro, 'INIT'
  ... do whatever ...
  call_procedure, mypro, 'CLEANUP'
```

The point is to apply a little discipline so things don't get too out of hand.

Craig

FILE_COMPILE is here: http://cow.physics.wisc.edu/~craigm/idl/misc.html

---