Subject: Re: IDL 8.0 compile_opt changes
Posted by wallabadah on Mon, 21 Dec 2009 01:10:56 GMT
View Forum Message <> Reply to Message

On Dec 20, 10:15 pm, Robbie <ret...@iinet.net.au> wrote:
> Another suggestion, although this feature might already exist in IDL
> 7.1 or 7.2.
>
> Would it be possible to get the workbench to automatically put
> compile_opt idl2 at the top of every new .pro file? This would ensure
> that new users experience the new language features. I personally
> don't use compile_opt as often as I should.
>
> Thanks
>
> Robbie

This is easy to implement - I wrote a routine to generate a function/
procedure skeleton with a few default options already written
(including compile_opt idl2), and a comment file header that
encourages me to comment my code properly (it works!). I use it on Mac
OS X at the IDL prompt in the Terminal like:
IDL> idl_codefile, "my_great_function", /is_function
and the skeleton file is constructed, named and opened for editing in
BBEdit/TextWrangler. There's also a version for creating new object
code files, where you supply an array of function names and an array
of procedure names, and the skeleton is generated so you can fill in
the appropriate bits of code. Can save a hundred or so lines of coding
- and I'm sure something similar could be integrated with the
workbench.

I think moving to a new file extension is a mistake, except if it were
used to distinguish between scripts/@includes and files of functions/
procedures to be compiled and run.
I think compile_opt idl2 should be the default, and also that there
should be a better mechanism for checking IDL version at runtime and
compile time, and people *should use it* in new code. How about if the
preprocessor/compiler issued warnings when code requiring a certain
IDL version is compiled... similar to the !warn structure variable.
For example, turning on the minimum_idl_version_required flag would
cause IDL to print that "IDL Version 6.X is required to run this
code". One could use it to check code before sending to an elderly
colleague still running IDL 5.x (or to someone running IDL 7.X, for
that matter). This would be very useful. It could even add it to the
source code itself (eg. top line, as a comment, including the line
number and context).
If we could then write conditional code that correctly allowed for
different IDL version (ie. don't parse the next bit of code if the IDL

version is such that IDL won't understand it anyway - read
http://www.dfanning.com/tips/backwards.html for a refresher), life
would be a fair bit simpler. No need for separate files for each IDL
version, just code that takes advantage of new features if the IDL
version allows, and uses older methods/workarounds/defaults to
something else if the IDL version is too low. This doesn't help with
legacy code, but I think it's a good feature to implement now for the
next time this sort of thing happens, and a standard practice for code
going into "publicly-accessible" libraries.

Writing a couple of routines to check existing code libraries and
update them should be a relatively simple exercise (for the IDL
developers!), as they already have the code required to parse IDL
routines written into IDL itself. As a previous poster has said, this
sort of thing should take care of the majority of existing code, the
remainder would probably require some human interaction. Some of this
old code probably needs a serious review anyway.

If people expect IDL to be updated and become a 'modern' language,
there need to be some serious changes, which are likely to affect
existing code. If you'd like to stay on IDL 6.X for the next decade,
go for it, but you can't have your cake and eat it too.

Will