Subject: Re: IDL 8.0 compile_opt changes
Posted by penteado on Sat, 19 Dec 2009 20:32:06 GMT
View Forum Message <> Reply to Message

On Dec 19, 5:11 pm, Chris Torrence <gorth...@gmail.com> wrote:
> Another point of information: We are adding a lot of new language
> features to IDL 8.0. Things like a new "foreach" operator, negative
> array subscripting a[0:-1], plus a lot more. You would need to avoid
> using *any* new language features if you wanted to run on an older
> version. This was always true, like when we added && || ++, etc.

I was wondering about that. Which is why I said that code could be translated, *if* all the syntax changes were as simple as the "." operator. If there are a lot of improvements (will lists, maps, and other containers finally arrive?), then code backwards compatible will be so limited (in the new perspective) that people will have to adapt around the change. In that case, it is good if the change is big, because it makes it easier to convince other people to upgrade.

But that is a separate issue, unrelated to making idl2 the default option. The big problem I see with just making it default is that some old code will stop working, and a lot of people will resist upgrading because of that. They already resist upgrading to IDL 7, which does not break much (I noticed a few changes in the iTools objects that did). The main difficulties to upgrades now are that it is hard to install and make it all work (in Linux anyway; I have not noticed difficulties in Windows, and I do not know the situation in Macs), and the license file may be limited to older versions. Remember that a lot of people use IDL on their own computer, which they can upgrade, but with the license through a department server, which they can not do anything about. It is very necessary to make upgrades easier, and breaking compatibility would only make them harder. And while a lot of people do not upgrade, it is hard for those who did to write code using the new features, and to teach them to new users, thus slowing progression to the new ways, which is already dreadfully slow.

Which is why I see the new extension as a better solution. Yes, just putting a compile_opt idl1 would immediately make the old routines work, but then the same file would not work on an older IDL (because of the unknown idl1), and a lot of people use the same programs on multiple computers, some of which they might not be able to update. So if they cannot retrofit old code (either because they cannot immediately stop their work to do it, or because they did not write the code and do not understand it), they will need to keep two sets of files, one with compile_opt idl1, to run in IDL 8, and another without it, to run on older versions.

With a new extension, old stuff keeps working, new users can learn in

the new ways, and retrofitting can be gradual, as time permits.

I do not see problem in associating IDL with .prx (or whatever, just make sure it is not in use by some other common program). I think it makes it easier to explain and interpret. With it, you immediately know the program takes IDL 8 to run; without it, if the writer did not inform you, there would be only a bunch of syntax errors to hint at an incompatibility.

Even though a new extension is like compile_opt, it is better, because it is more visible: everybody would see there is something new, even without opening the file, while a compile_opt may easily go unnoticed inside it. I myself only learned about [] and compile_opts after years of using IDL without knowing they were there. Yes, it is the result of faulty education I got, but it is the typical kind of faulty computer programming education that scientist keep getting. I would probably have wondered about why there is another extension much sooner.

Also, today when we encounter a .pro file we already wonder if it is old and crufty anyway, so we have to look inside it.

---