Subject: Re: IDL 8.0 compile_opt changes
Posted by R.Bauer on Mon, 04 Jan 2010 10:30:39 GMT
View Forum Message <> Reply to Message

Chris Torrence schrieb:
> Hi all,
>
> I'm writing to you to ask your opinion on some potential changes in
> IDL 8.0. We have made some enhancements to the language to support the
> new graphics functions, and to make IDL simpler to learn, especially
> for new users.
>

Hi Chris

Happy new year to you and ittvis. Thanks for asking us.

> The primary change is the use of the dot "." for object method calls.
> The use of the "." for method calls is now industry standard, for
> example in languages such as Java, Python, etc. For example, in IDL
> 8.0, the following code will create a plot, and retrieve the first
> child object:
>
> p = PLOT(x,y)
> child = p.Get(index)

I support this change. I felt always the attempt of doing OO in idl is
extremly uncomfortable and ugly compared to python.
I think this change is important for new users or old users using OO in
different languages.

>
> Disregarding the actual syntax, the problem occurs when IDL tries to
> compile the second line of code. By default it will think that "Get"
> is just an array field inside of a structure, and that the parentheses
> are just indexing into the array.
>
> Now, you could use "compile_opt idl2", or "compile_opt strictarr" to
> remove the ambiguity. In this case, for IDL 8.0, it will then know
> that this is a function method call.
>
> That's fine for existing users, like us, who are happy to sprinkle
> "compile_opt idl2" all through our code. However, for new users this
> is strange, and it would be hard to explain why they are getting a
> syntax error.

In my eyes the change should have been done one release after the

parenthesis change. Now it is quite late. Some of my coleagues still use the old style '()' and they have produced lots of stuff. I often become told "Why should I change this - it works". There would have been less brokeness if they had done it right at the right time.

Because we have lots of users using `()` I want to get a refactoring tool which converts `()` into `[]`.

Also it is a good idea to refactor some of that old code or remove the crap.

I think it is urgent to skip this old compile opt feature for the old style as soon as possible. Yes I don't want it to be kept!

>
> For IDL 8.0, we want the new language features like the "dot" to work
> "out of the box". So, ideally, we would change the default compile_opt
> for IDL to be "idl2" - this includes both "defint32" and "strictarr".
> All integer scalar constants would be 32-bit by default, and
> parentheses would not be usable for array indexing. However, this
> creates a backwards compatibility problem. IDL .save files would be
> fine (the code is already compiled), but large libraries (like JHUAPL)
> would not be usable without changes to the code.
>
>
> Possible solutions:
>
> 1. Change the default to be "compile_opt idl2", add a new "compile_opt
> idl1" to restore the existing behavior, and require users to retrofit
> existing code.

No

see above

>
> 2. Only change the default behavior for arrays within structures. Add
> a new "compile_opt allow_parens_with_structure_fields" (obviously that
> would not be the name). Existing users would still need to retrofit
> code, but not as much as #1.

No

see above
>
> 3. Do #1 or #2, but also add a global preference for the compile_opt.

> By default it would be "idl2", but users could change it (we would
> need an "idl1" to turn off the new behavior). This is bad if the user
> wants to use existing libraries, but also wants to use the new method
> calls.
>

No

see above

> 4. Do nothing, require users to use "compile_opt idl2" if they want
> the new "dot" methods. This is bad for new users, as they will get
> strange syntax errors and will not understand why.
>

No

see above

> 5. Add a new ".prx" extension (name TBD). If you have an existing
> ".pro" file then the defaults remain unchanged. The new ".prx" would
> default to "compile_opt idl2". This solves the problem, but might
> cause a "split" in the code base and confusion for the users.
>
>

hmm No

At first glance it sounds like it can be the output of the refactored
old sources. And the old pro files were interpreted by a builtin idl7.

But that shifts the problem only into a different extension. You have by
this a compile opt by a file extension.

So I disagree - This is also no solution

> We would really like to hear your opinion on these potential
> solutions. Questions to think about:

>
> 1. How much code do you have that would break? Are you willing to
> retrofit your code?

Can't estimate

The code I am responsible for could be refactored by me. I like to have
a tool which helps to do it and which I don't have to write on my own.

>
> 2. Do you use existing libraries (like Astrolib or JHUAPL's) that
> would break? Could ITTVIS retrofit these 2 libraries and give them
> back to the community?

We use these libraries. One problem is that most of the code has only doctests as examples. If you convert these libs how do you test that you did it correct?

>
> 3. Are there potential issues with changing to "defint32" (32-bit
> integers), or is the only problem with parentheses for arrays?
>

For our sources it is "only problem with parentheses for arrays".

>
> Thanks for reading all of this, and we look forward to your replies.
> Happy Holidays!

Please also add to idl8 features we urgent need. netCDF4, svg output. Otherwise we don't need to do this effort.

best regards

Reimar
