Subject: IDL 8.0 compile_opt changes
Posted by chris_torrence@NOSPAM on Fri, 18 Dec 2009 21:51:39 GMT
View Forum Message <> Reply to Message

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.

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)
```

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.

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.

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.

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.

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.

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.

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?

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?

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

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

-Chris

Dr. Christopher Torrence
IDL Software Development Manager
ITTVIS