Subject: Re: IDL 8.0 compile_opt changes
Posted by Maarten[1] on Wed, 06 Jan 2010 14:01:27 GMT
View Forum Message <> Reply to Message

A lot has been said about the subject, but I want to add a few
thoughts as well. This seems to be an appropriate time.

On Dec 18 2009, 10:51 pm, Chris Torrence <gorth...@gmail.com> wrote:

> 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.

There is a lot in IDL that is decidedly _not_ industry standard
(calling a function and discarding the result, the way procedures are
called (as opposed to functions), character handling, named
parameters, manual line breaking (when a smarter parser could figure
out that a $ is still needed, ...). Are there better arguments to
require this change?

Don't get me wrong: _finally_ dusting off some parts of IDL is good in
my opinion. But one of the important reasons people use IDL is the
existing code, not its particularly pretty syntax.

> 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.

Others have commented on this in depth. Some changes can be automated,

at least in part. For some of the really old code you have to wonder
if leaving it to run on old IDL would be a bad thing.

While you're at it: I'd like to have an option to use double precision
by default, only changing back to single precision when I explicitly
ask for it.

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

I think that these days (especially for new users) there are more
issues caused by the default short integers and single precision
values than the reverse. This excludes of course direct binary
reading, but for reading binary files you should have been explicit
from day one.

I do love the idea of negative indices, although I'd like them to mean
the same as in Python. The samples Ive seen so far are off by one.

Best,

Maarten