Subject: Re: IDL 8.0 compile_opt changes
Posted by H. Evans on Fri, 08 Jan 2010 09:29:51 GMT
View Forum Message <> Reply to Message

On Jan 7, 5:35 pm, Paul van Delst <paul.vande...@noaa.gov> wrote:
> Chris Torrance 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.
>
> So? You open Pandora's box simply because newer languages do it a particular way?
>
> And there are languages that don't use "." for method calls or structure dereference: e.g.
> Fortran90/95/2003/2008
>
> In Fortran, "%" does that. (2003+ for the object method invocation)
>
> I remember when Fortran90 was still being adopted and there was much wailing, gnashing of
> teeth, and wringing of hands about the fact that the Fortran standard used "%" rather than
> "." (the latter being used in some earlier DEC extentions) for structure component
> dereferencing.
>
> Well, you know, we all got over it. :o)

Not all of us >:-(. I still consider it a bone-headed decision. The "%" sign as a delimiter is atrocious graphically. An ideal delimiter is simple, e.g. '.', '_', '-', ' ', allowing the eye to quickly identify it and use it to separate the surrounding text. Then there is all the legacy code that's 'broken'. Which is easier to quickly parse by eye: This.that, This_that, This-that, This%that. I'd love to know what the compelling argument was over the previous glorious DEC Fortran standard of '.'. Back on topic: the IDL '->' is quite reasonable in this respect.

The file extension is something that's easier to understand and live with on a daily basis. It's a single 'configuration' of the source code file at a single instance in time: it's creation.

>
> My PO is if people get annoyed when they see
> child = p->Get(index)
> because they think it should look like
> child = p.Get(index)
> they need to gain perspective about what is *really* important.
>

Hallelujah! I agree completely. Is there a better reason for migration

to '.' from '->' other than it might be 'easier for new users' as I
consider that argument a bit specious.

If we are rewriting the syntax in a big way, how about replacing the
'$' continuation line character for the industry standard' semi-colon
for command delimiting, and then using the '//' characters for comment
delimiting.

> And, I would prefer ITTVIS's time be spent doing useful stuff like providing additional
> functionality rather than generating make-work for themselves and their users. E.g.:
>
And
- Extensions to the histogram function to calculate histograms
  of weighted values - not all data is created equal,
  e.g. h = histogram(x, weights=w), where it is the "w" elements
  that are tallied in the bins, and not the number of 'x's in the bin.

- Anti-aliasing built into the direct graphics functions (or am I the
only person
  left that finds the object graphics to be overly complex and tedious
to
  produce a simple plot, while the direct graphics quality not of
publication
  quality?) Alternatively, a complementary function for the old direct
graphics
  routines that use object graphics, e.g. dgplot as a direct
replacement of
  plot. Particularly one that can be used in batch mode (no attached X
or window
  device - a problem with using the iTools for batch processing).

- definitely support localisation of namespaces for common blocks/
functions. Ok,
  perhaps we should be moving over to using objects anyway to
encapsulate, but
  even then there is name space clashing potential for object names.

- an ability to run dynamic library (call_external, dlm, etc) routines
in a multi-threaded
  execution space. I'm currently using a bridge object, but it's a bit
hit/miss. Shared
  memory in these bridge objects for strings, etc?

- more functionality on accessing the contents of SAVE files (delete
variables, overwrite,
  extract just a single named one, rename contents, etc.)

H.