Subject: Re: IDL 8.0 compile_opt changes
Posted by Kenneth P. Bowman on Wed, 06 Jan 2010 14:52:25 GMT
View Forum Message <> Reply to Message

In article
<5e4a2c1a-4de9-4a4e-8399-f5ec724e150c@j5g2000yqm.googlegroups.com>,
 JD Smith <jdtsmith.nospam@yahoo.com> wrote:

> 1. Overloading structure dereference and method invocation breaks the
> ability to semantically parse a.

I have been following this discussion and trying to understand the implications. I don't claim any expertise in compiler construction or parsing methods. I think I agree with JD on this one, though.

It seems to me that the fundamental problem with parentheses in IDL is that they can mean two different things (array subscripting or function referencing). This is partially remedied by allowing (but not absolutely requiring) square brackets for subscripting.

Now the idea is to create a similar ambiguity with the dot operator? And the main justification is to make IDL read more like Python? As someone who uses structures a *lot*, but not objects, I can see the potential for serious confusion, by the programmer, if not by the parser. I can also hardly wait to try teaching this to beginning programmers.

> "The dot operator is used to indicate a structure field, except
> it can also mean to use an object method. What's an object
> method? Oh, we haven't gotten to that yet, but when you read
> someone else's programs, make sure you know the difference."

I love JD's example

> self.limit, self.limit

A compiler option to force double precision floating point constants and variables would be very useful to me (DEFFLT64?).

If this change is radical enough to require a new file suffix, shouldn't the opportunity be taken to change so many other "minor" issues, like default 2-byte integers?

Cheers, Ken