Subject: Re: IDL 8.0 compile_opt changes
Posted by penteado on Wed, 06 Jan 2010 18:35:13 GMT
View Forum Message <> Reply to Message

I realised now that I had not thought this through:

On Jan 5, 9:54 pm, pp <pp.pente...@gmail.com> wrote:
>> 1. Overloading structure dereference and method invocation breaks the
>> ability to semantically parse a.b.
> That is a good point that I had not considered. I only noticed that
> with the assumed idl2 option, there would be no ambiguity in complete,
> correct lines. But as you point out, there would be in the middle of
> writing a line. Though, as you indicate in (3), the ambiguity would
> only occur between methods and fields of self. In other cases, the
> interpreter should know whether you typed a structure or an object.

This is not actually a problem. It would just be the same kind of ambiguity that happens in the middle of typing a name, when there are more than one that begin with that same characters. At that point, using completion should give the user both options. In the case of the function/member ambiguity, the options would be "a.b(" and "a.b", as the first is the function, and the second is the structure member.

> Contrary to C++, Java and Python, the flat namespace and case-
> insensitivity already make it trickier to pick names in IDL, so it may
> be even more important to differentiate between methods and fields
> with the operator.

I had not remembered that in C++ and Java it is not possible to have the same name associated to a function and a variable at the same time. In Pyhton, the last use for a name rebinds it ("overwrites" the previous use), so there is only one thing with the name. So there is no ambiguity in them. In IDL, assuming the idl2 option would remove the ambiguity through the syntax, so no problems either.

So, yes, it is a bit easier when reading a program to resolve the meaning through a different operator, as it is now. But I find it only slightly easier than looking at the syntax, so not much of an issue. Contrary to what some people have written in this thread, this is not the same as the ()/[] issue, as only looking at a line written using () is not enough to know if it is a function or an array. In the case of the dot operator, assuming the idl2 option, there would not be any ambiguities in complete lines of code.

The only situation with an ambiguity would be the one mentioned by JD, in the middle of writing a line, when using autocompletion. But in those moments there is the unavoidable ambiguity from having several names that begin with the same characters anyway.

It could be argued that the dot is more proper because both data and functions are conceptually the same, they are "parts" of an object, and perhaps this is the reason for its choice in C++, Java and Python. So I do not see much difference between using either operator. Either is fine, as long as the idl2 option is assumed if the dot operator is used.

However, in the separate question of how to assume the idl2 compile option, which is already needed, even without the dot operator change, there are very important implications. Which is why I previously argued that a new extension is the best choice. The new extension is better even than the commented idl1 option, since no change to old files is required, and it prevents old IDL versions trying to interpret code that they cannot understand properly (as in the issue #2 mentioned by JD).
