Subject: Re: A Simple IDL Manifesto
Posted by Michael Galloy on Thu, 20 Nov 2008 16:42:10 GMT
View Forum Message <> Reply to Message

On Nov 20, 2:11 am, Reimar Bauer <R.Ba...@fz-juelich.de> wrote:
> If one changes the code rules behaviour he can also write a migration
> tool which converts your old program into a better new program.
>
> But this wasn't also done by this company in the past, so we can assume
> it won't be in the future.
>
> And it is not unusual to do so. e.g. if the moinmoin wikisoftware
> project changes the wiki text syntax in a newer version we provide a
> migration tool for the old wiki text syntax on pages to the new syntax.

Yes, a similar change like this is happening for Python 3000 i.e.
Python 3.0. They are providing a py2to3 tool that will convert Python
2.6 to Python 3.0 code. This tool will do most of the grunt work, but
I believe some hand coding will still be necessary.

Backwards compatibility is a noteworthy goal, but the design of any
language will eventually show its age. I think eventually you have to
change things that turned out to be mistakes (hindsight is 20/20).

The things that I would change about the core language of IDL (not the
library) that would break backward compatibility would be:

1. get rid of that extra "blankity, blank" comma when calling a
procedure (the one right after the name of the procedure)

2. consistent handling of arrays with a last dimension of 1 (don't
remove dimensions for me, thank you)

3. allow arrays of length 0

4. make "compile_opt idl2, logical_predicate" the default

A conversion tool could probably do 1 and the "idl2" part of 4 pretty
easily. 2, 3, and the "logical_predicate" of 4 would be a bit harder
and probably require some overview by the developer.

Of course, this means that code written for this new "IDL 8" would not
work in previous versions (the tool would only convert from old to new
style). If the .sav file format didn't change, then at least "IDL
8" .sav files could be used in previous versions.

Another solution would more compile_opt flags, but I'm not sure what
should happen if a routine with the new compile_opt flag had an array

of length 0 and passed it to a routine without the new compile_opt flag. And I'm getting tired of putting a compile_opt statement in every routine I write.

Mike
--
www.michaelgalloy.com
Tech-X Corporation
Associate Research Scientist
