Subject: Re: IDL to C translator
Posted by David Fenyes on Wed, 08 Apr 1998 07:00:00 GMT
View Forum Message <> Reply to Message

Liam Gumley <Liam.Gumley@ssec.wisc.edu> writes:
Liam Gumley <Liam.Gumley@ssec.wisc.edu> writes:

>
> Axel Schweiger wrote:
>> Here are some very good reasons why an IDL -> C/Fortran translator (or a IDL
>> compiler ) would make sense:
>> 1) IDL can be extremely slow if your code/algorithm requires you to loop over
>> an array. I don't think a lot of people are coding applications
>> where speed is an issue in IDL.
snip
> "Let's convert all our FORTRAN to C, because C is more portable", or
"Let's auto-convert FORTRAN to C, because most people have a C compiler."
Done. f2c

> "Let's convert all our C to C++, because it's object oriented", or
If this is done automatically, it is to fit existing C code into a C++
framework quickly.

> "Let's convert all our C++ to Java, because it's platform independent".
Compiling C->jvm is reasonable, particularly if it's debugged and to
be linked with new JAVA code.

I think the original request is reasonable. In fact, IDL to C
directly would be something of a pain, but via LISP it would not be as
hard as you'd think. IDL4 is a lot like a crippled lisp with FORTRAN
syntax. For IDL4, and IDL->LISP compiler would be quite feasible, if
appropriate widget and math libraries were added. Lisp is easily
compiled to C or assembly.

For IDL5 this is more difficult, because of the pointers. If those
are avoided, IDL->Lisp may still be feasible.

>
> I feel much better now.
>
> Cheers,
> Liam.

--
David Fenyes University of Texas Medical School
dave@msrad23011.med.uth.tmc.edu Dept. of Radiology