Subject: Re: IDL to C translator
Posted by Liam Gumley on Tue, 07 Apr 1998 07:00:00 GMT
View Forum Message <> Reply to Message

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.

IMHO if loops are slowing down your IDL program, then your IDL program isn't designed properly. A much more effective way of speeding up your application would be to spend some removing the loops from the IDL code. This could be done in a much shorter time than recoding and testing the whole program in FORTRAN or C. One of the key steps on the way to becoming effective IDL programmer is learning to think in IDL (rather than FORTRAN or C).

A couple of related principles:
- You can write bad FORTRAN in any language,
- A bad design is still bad whether it's coded in IDL or C or FORTRAN.

> 2) Not everybody has IDL and you may want to share an application.
> BTW: Matlab has a matlab -> C translator. I suspect for the above reasons.

Surely if your application is brilliantly implemented in IDL, then anyone that wants to use it will fall over themselves in the rush to buy an IDL license.

But seriously, I know it's often assumed that porting to a different language will fix all the evils in your code. Does any of the following sound familiar?
"Let's convert all our FORTRAN to C, because C is more portable", or
"Let's convert all our C to C++, because it's object oriented", or
"Let's convert all our C++ to Java, because it's platform independent".
The truth is, if your IDL code works, then you ought to get on with using it to do something useful. Don't waste a bunch of time and money converting it to another language because of some poorly defined benefits.

I feel much better now.

Cheers,
Liam.