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.

Subject: Re: IDL to C translator

Posted by Axel Schweiger on Tue, 07 Apr 1998 07:00:00 GMT
View Forum Message <> Reply to Message

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.

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.

Axel

Liam Gumley wrote:

> Pedro J. Claudio wrote:
>> Does anyone know of an IDL to C or Fortran translator. I'm trying to
>> port a lot of IDL code and could really use anything that could automate
>> the process.
>
> No such translator exists.
>
> Just out of curiosity, why would you want to translate a working IDL
> application into C or FORTRAN?

--
#---------- ----------
# Axel Schweiger Tel:206-543-1312
# University of Washington Fax:206-543-3521
# Polar Science Center/Applied Physics Laboratory
# 1013 NE 40th Street
# Seattle, Wa 98105 axel@apl.washington.edu
#-------The secret of the polar bear is that he wears long underwear----------

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

Pedro J. Claudio wrote:
> Does anyone know of an IDL to C or Fortran translator. I'm trying to
> port a lot of IDL code and could really use anything that could automate
> the process.

No such translator exists.

Just out of curiosity, why would you want to translate a working IDL
application into C or FORTRAN?

---

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

---

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

Pedro J. Claudio wrote:
> IDL doesn't help me If I'm trying to port an application to an embedded
> architecture. That's our main line of business here. Take a toolset like
> MATLAB with Simulink and the Image Processing Toolbox, I can draw a
> block diagram, test it and generate code with the C code generator.
>
> Bottom Line: IDL and its cousin PVWAve are great for algorithm
> development but fall down when I need to implement in a real world
> embedded system.

Fair enough. I don't think IDL has ever advertised itself as supporting realtime applications. In your case, it sounds like an IDL to Matlab translation would be perhaps be more appropriate.

Is there anyone out there using IDL runtime applications in an embedded system?

Cheers,
Liam.

---

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

Ebeth Jones wrote:
>
> Liam Gumley wrote:
>
>>
>>> 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.
>
> I feel like a trout on a hook, but here goes....
>
> no matter how brilliantly your application is implemented in
> IDL, if you need to implement that code in a real system, i.e.
> a fielded military system, chances are the hardware that is
> going in the fielded system will not even entertain the thought
> of hosting IDL for your brilliant application, and your brilliant
> application in IDL will not run at real time (not near real time,
> real real time) no matter how much you tweak it. Unless you
> can enlighten me further....

Well surely you would never code a military real-time application in IDL in the first place. A *prototype*, certainly. But the idea behind prototypes is that once you're done with them, you throw them away. In fact, you might even throw away two prototypes, e.g.

(1) Prototype V1 (IDL) demonstrates the user interface, to get feedback from users (no functions)
(2) Prototype V2 (IDL) demonstrates basic functionality
(3) Realtime V1 (C++) is the first production version

Code re-use in this process might happen from steps (1) to (2), but it probably would not happen from step (2) to step (3). The key information that gets refined in this process is the application *design*. It is perfectly appropriate to code the application at step (3) in whatever language is appropriate. I'm just saying that you're probably wasting your time if you try and 'translate' the code from step (2) for use in step (3).

Cheers,
Liam.

---

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

Liam Gumley wrote:
>
> Pedro J. Claudio wrote:
>> Does anyone know of an IDL to C or Fortran translator. I'm trying to
>> port a lot of IDL code and could really use anything that could automate
>> the process.
>
> No such translator exists.
>

> Just out of curiosity, why would you want to translate a working IDL
> application into C or FORTRAN?

IDL doesn't help me If I'm trying to port an application to an embedded architecture. That's our main line of business here. Take a toolset like MATLAB with Simulink and the Image Processing Toolbox, I can draw a block diagram, test it and generate code with the C code generator.

Bottom Line: IDL and its cousin PVWAve are great for algorithm development but fall down when I need to implement in a real world embedded system.

Regards,

Pedro

---

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

Liam Gumley wrote:

>
>> 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.

I feel like a trout on a hook, but here goes....

no matter how brilliantly your application is implemented in IDL, if you need to implement that code in a real system, i.e. a fielded military system, chances are the hardware that is going in the fielded system will not even entertain the thought of hosting IDL for your brilliant application, and your brilliant application in IDL will not run at real time (not near real time, real real time) no matter how much you tweak it. Unless you can enlighten me further....

Ebeth "furrfu" Jones

---

Subject: Re: IDL to C translator
Posted by davis on Thu, 09 Apr 1998 07:00:00 GMT

On Tue, 07 Apr 1998 10:28:11 -0500, Liam Gumley <Liam.Gumley@ssec.wisc.edu>
wrote:
> Just out of curiosity, why would you want to translate a working IDL
> application into C or FORTRAN?

I can think of several reasons: speed, portability, and
maintainability.

IDL is fast only when one is able to vectorize everything.
Unfortunately, this is not always possible. For example, I once
re-wrote an IDL application in C that would not completely vectorize
and achived a factor of 20 in speed. Of course this was more portable
because the resulting program was able to compile and run on systems
that did not have IDL. Finally, it is more maintainable because, in
my opinion, IDL is not a good language for projects that require many
lines of code because it lacks basic support for variable
declarations, etc... In fact, when I ported the IDL program to C,
several variable name spelling errors were uncovered because all C
requires that one declare all variables before use.

--John