Subject: Re: Compiling IDL code with a C compiler
Posted by David Fanning on Sat, 17 Jul 2004 23:25:42 GMT
View Forum Message <> Reply to Message

Cedric writes:

> So before I embark on such an endeavor, I decided to write here to get
> some input. Is there anything like that already out there? Is there a
> fundamental flaw in my thinking? Any suggestions, advice?

Has anyone pointed out to you that IDL is a weakly typed
language? Well, best of luck to you! :-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Compiling IDL code with a C compiler
Posted by cedricl on Sun, 18 Jul 2004 15:16:22 GMT
View Forum Message <> Reply to Message

David Fanning <davidf@dfanning.com> wrote
> Has anyone pointed out to you that IDL is a weakly typed
> language?

Frankly, I've always had trouble distinguishing between
strong/weak/static/dynamic typing. Looking on the Internet, they say
that C++ is weakly typed, too. How does it create a problem? If you
look at my example, I have type declarations as ""preprocessor
directives"" at the top:

; Type Declarations
;#field = fltarr(101, 101)
;#area = fltarr(101, 101)
;#sum = float(0.)

(I'm thinking of replacing those with ASSERT_TYPE, field, fltarr(101,
101) in the IDL version)

Besides, if I restrict my support of IDL syntax to a bare minimum (so
much that the user is essentially writing C with IDL syntax, but
without having to care about the interfacing), then there really
doesn't seem to be that much to the "translation" phase.

Another advantage of this approach is that the IDL code would still be compilable directly with .compile, since the preprocessor directives are commented. So it would be good for testing the correctness of the code.

> Well, best of luck to you! :-)

Thank you!

Cedric

Subject: Re: Compiling IDL code with a C compiler
Posted by David Fanning on Sun, 18 Jul 2004 15:52:05 GMT
View Forum Message <> Reply to Message

Cedric writes:

> Besides, if I restrict my support of IDL syntax to a bare minimum (so
> much that the user is essentially writing C with IDL syntax, but
> without having to care about the interfacing), then there really
> doesn't seem to be that much to the "translation" phase.

I guess my point is that if you restrict the user to
"essentially writing C with IDL syntax" the user might
as well write the C code. That way, he would know for
sure that it will compile. :-)

Cheers,

David

P.S. Let's just say I'd be curious to meet the user
you have in mind. :-)

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Compiling IDL code with a C compiler
Posted by Haje Korth on Mon, 19 Jul 2004 11:59:10 GMT
View Forum Message <> Reply to Message

Cedric,

You re not the first one with this idea. Unfortunately, we never hear a success story from the people here in this news group. I would strongly suggest to drop this thought and enjoy the summer instead. :-)

Cheers
Haje

"Cedric" <cedricl@videotron.ca> wrote in message news:f09ca592.0407171110.53f3b776@posting.google.com...
> Hi everyone,
>
> I'm interested in writing an IDL-to-C compiler, for optimization
> purposes. To be clear about what I'm talking about, here's what some
> sample IDL code would look like:
>
> ;#COMPILE gcc -O1
> function EvaluateEnergy, field, area
> ; Type Declarations
> ;#field = fltarr(101, 101)
> ;#area = fltarr(101, 101)
> ;#sum = float(0.)
>
> sum = 0
> for x=0, 100 do begin
> for y=0, 100 do begin
> sum = sum + field[x,y] ^ 2 * area[x, y]
> endfor
> endfor
> return, sum
> end
>
> My IDL-to-C (pre)compiler would parse the IDL pro files, looking for
> functions preceded by a ;#COMPILE (aka ~preprocessor directive) and
> would translate the subsequent IDL code into the equivalent C code,
> compiling it with the options specified before. It would then replace
> the body of EvaluateEnergy with the proper external function call, and
> compile it with IDL's normal .compile compiler.
>
> So before I embark on such an endeavor, I decided to write here to get
> some input. Is there anything like that already out there? Is there a
> fundamental flaw in my thinking? Any suggestions, advice?
>
> Thank you,
>

>

> (Note: I'm fully aware that the function of my example should be a
> one-liner; it's for demonstration purposes, and because I intend to do
> a "litteral" IDL-to-C translation at first, and not support IDL's
> numerous notational shortcuts)

Subject: Re: Compiling IDL code with a C compiler
Posted by cedricl on Mon, 19 Jul 2004 13:05:51 GMT
View Forum Message <> Reply to Message

David Fanning <davidf@dfanning.com> wrote
> I guess my point is that if you restrict the user to
> "essentially writing C with IDL syntax" the user might
> as well write the C code. That way, he would know for
> *sure* that it will compile. :-)

Well, I'm writing it first and foremost for myself. Advantages:

- As I've said, I would have a testable IDL version to ensure code correctness _in case_ I have doubts about my IDL-to-C compiler.

- It would be much more integrated into the IDL code, much faster for debugging, or small modifications. More portable, too.

- Nothing prevents me from supporting more of IDL's syntax, like the [:] notation, so that I can write more natural IDL.

Personally, I'd be much more tempted to optimize a routine if I only had to make a few loops explicit and a few type declarations, than if I have to start a separate .c file, compile it, debug it, and make an IDL wrapper.

Perhaps I'm more lazy than your are. They say that it's a quality for a programmer ;-)

Anyway, you've sown the seeds of doubt in my soul, so I'll meditate it some more. Thank you!

Cï¿½dric
