Subject: Re: "(national) C compiler"
Posted by mr_ped on Wed, 01 Apr 2009 09:58:26 GMT
View Forum Message <> Reply to Message

kbyte wrote on Wed, 01 April 2009 11:20
1- "Actually, C is pretty simple. I believe I could do basic C compiler in a week (badly optimizing, compiling to e.g. x86 assembly)."

This is the most interesting but even not wanting an optimized compiler I think it requires a great knowledge on assembly, which is not my case.

Actually in your special case as you don't care about the performance too much as much you care about language, you don't need to compile into assembly, you can compile to C (simple search & replace compilation) and then call some C compiler (GCC for example).

This is same to macro approach, but instead of #include with macros you can handle this with your own source preprocessor and then postprocess compiler output.

This returns me back to CParser and search&replace concept, but I'm not sure how much CParser can be used in case it does not understand some more complex class or macros.
Thinking about it more, this is quite a special case. You basically don't need macros, you need to translate the source into C before preprocessing will apply (but you have to translate all native language includes too, maybe doable separately without parsing their includes... basic clib includes I would left as they are)

Maybe you can have .hn files for includes with native language, and parse those separately into .h files, then in the final C source include .h instead of .hn.
All you need is special preprocessor which will take turn ahead of C/C++ preprocessor? Which will replace all native keywords into C keywords, but will recognize C/C++ strings/comments and replace just code. With ordinary code this should be very simple with CParser class, with macros you may run into some problems, maybe Mirek will get the idea and answer you with more details which cases it would handle well.
And the postprocess parser for compiler output, that's pure text search&replace I think.

edit:
I would start probably with the 3rd option too ... with some friendly compiler which is easy to build and allow an easy change of keywords, you are done withing minutes without actually coding anything.