

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,

Cédric

(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 "literal" IDL-to-C translation at first, and not support IDL's numerous notational shortcuts)
