

Replacements for the C preprocessor [closed]

Asked 15 years, 11 months ago Modified 7 years, 9 months ago

Viewed 17k times



45



Closed. This question is seeking recommendations for software libraries, tutorials, tools, books, or other off-site resources. It does not meet [Stack Overflow guidelines](#). It is not currently accepting answers.



We don't allow questions seeking recommendations for software libraries, tutorials, tools, books, or other off-site resources. You can edit the question so it can be answered with facts and citations.

Closed 7 years ago.

[Improve this question](#)

I'm interested in using something other than the **C** preprocessor to preprocess my **C** and Objective-C source code. Are there good alternatives?

An example would be something that allowed one to escape out into a python or perl snippet in the middle of **C** code, and where the snippet spit out **C** that is then compiled as normal.

objective-c

c

c-preprocessor

Share

Improve this question

Follow

edited Feb 3, 2016 at 15:32



Brian Tompsett - 汤莱恩

5,875 ● 72 ● 61 ● 133

asked Dec 28, 2008 at 20:32



Ken

13k ● 4 ● 31 ● 32

- 1 Could you provide a concrete example that shows how (and why) you'd like to generate source code at compile-time? That might help us come up with suggestions that target your specific needs. (For example, I'm guessing that C++ templates won't work for your case, but it would be good to confirm.) – [reuben](#) Dec 28, 2008 at 20:59
- 2 Not a bad idea: Here is an IBM article where the author uses a perl script to generate a lookup table.
ibm.com/developerworks/linux/library/l-metaprogramming1/index.html
– [user295190](#) Aug 26, 2011 at 20:45

13 Answers

Sorted by:

Highest score (default)



You can use **PHP** as a C preprocessor. The advantages are:

83



- very similiar syntax, so syntax highlighting works.
- `<?>` and `?>` are not used in standard C (with non-standard C, the only thing that gets broken is old





GCC extension operator that returns min/max)

- it's rich in libraries.
- it's turing complete.
- usage of macros is very explicit. (compared to sneaky C preprocessor macros)

For serious use though, making PHP print the #line directives is needed for debugging preprocessed code.

```
<?php include_once "stdio.h"; ?>

int main()
{
    <?php
        for($i = 0; $i < 20; $i++)
            echo 'printf("%d\n", '.$i. ');';
    ?>
}
```

Share Improve this answer

edited Apr 28, 2013 at 10:51

Follow

answered Apr 27, 2013 at 19:28



[milleniumbug](#)


15.8k ● 3 ● 51 ● 72

7 that is fascinating – [Félix Adriyel Gagnon-Grenier](#) Oct 6, 2016 at 22:50

54 As this is now spreading on Twitter, Reddit etc. people please remember - just because you *can* or *could* doesn't mean you *should*. – [Esko](#) Oct 7, 2016 at 9:55

2 @Esko: Actually, I think it's an awesome idea if you want to dynamically generate code during compilation - in particular if the C preprocessor isn't powerful enough for what you're trying to do. – [thejh](#) Oct 7, 2016 at 12:25

1 Wow. This is pessimum. – [Warren P](#) Oct 8, 2016 at 0:19

9 I finally found the true meaning of the acronym: "P"HP "H"eader "P"reprocessor. It's a perfect tool to preprocess `.h` files. – [Star Brilliant](#) Oct 8, 2016 at 11:02 



[Cog](#) isn't exactly a pre-processor, but it does go in-line in the code and generates stuff on the fly.

11

Share Improve this answer

answered Dec 28, 2008 at 20:55



Follow



[Michael Kohne](#)

12k ● 3 ● 50 ● 82



1 can you expand your answer (I believe it's best answer) by giving some example (otherwise I will make mine)
– [Xavier Combelle](#) Oct 7, 2016 at 14:39



You might want to consider m4.

<http://www.gnu.org/software/m4/>

10

Share Improve this answer

answered Dec 28, 2008 at 20:34



Follow



[David Poole](#)

3,512 ● 5 ● 38 ● 35





-
- 2 To me, m4 feels too much like a science project and not enough like a pre-processor language. Last time I looked, it didn't have a looping construct, but instead had instructions on how to make one using recursion. I took that as a clear signal that the authors were too smart for my own good.
– [Michael Kohne](#) Dec 28, 2008 at 20:55
-
- 1 looping constructs, like these?
[gnu.org/software/m4/manual/html_node/... – [Hasturkun](#) Dec 29, 2008 at 8:20
-
- 3 `m4` really is extremely versatile, but that syntax is just so brutal.! I really wish we had a more modern unix macro processor, but one compiled from C, like the venerable `m4` .
– [J. M. Becker](#) Jun 6, 2013 at 18:06 ✎
-
- 2 I wrote a blog post about using m4 with C which might help a few get started: kvanberendonck.id.au/using-m4-with-c
– [kvanbere](#) Apr 27, 2014 at 1:28
-
- @TechZilla, Michael, David, Hasturkun, Folks, am I missing something? What's wrong with using a real scripting language like PHP/JavaScript/Perl as a preprocessor?
– [Pacerier](#) May 16, 2015 at 0:37 ✎
-



9

The idea that you **run code, the result of which is then spliced in** is called **quasiquote**. The code you run is **antiquoted**.



I know how to solve this problem using Lua. I've used `string.gsub` with an **antiquotation function I wrote myself**. I've used shell syntax for the antiquotation. As in





the shell the **antiquoted code** returns a **string** which is then spliced into the code.

Below `prog` is the C code with antiquoted text, and `antiquote` is the antiquotation function. I've used Lua's special string quoting double square brackets to full advantage. *In practice you wouldn't do this*; you'd put `prog` in a separate file.

```
names = { 'John', 'Paul', 'George', 'Ringo' }

local prog = [=[
#include <stdio.h>

main() {
    $(local out = { }
        for _, n in ipairs(names) do
            table.insert(out, string.format([[ printf("The
n))
        end
        return table.concat(out, '\n  ')
    ])
}
]=]=]

local function antiquote(s)
    local body = s:match '^%$$(.*)%$'$
    return assert(loadstring(body))()
end

prog = prog:gsub('%$%b()', antiquote)
io.stdout:write(prog)
```

In use, the program looks like this:

```
: nr@curlycoat 1181 ; lua /home/nr/tmp/emit-c.lua
#include <stdio.h>
```

```
main() {
    printf("The name is %s\n", "John");
    printf("The name is %s\n", "Paul");
    printf("The name is %s\n", "George");
    printf("The name is %s\n", "Ringo");
}
```

Share Improve this answer

answered Dec 29, 2008 at 2:23

Follow



Norman Ramsey

202k ● 62 ● 371 ● 541



6

Sure, the standard C preprocessor is very limited.
I've done such a tool recently: <https://github.com/d-ash/perlpp>



For example this



```
<?
    my @types = ('char', 'int', 'long');
    foreach (@types) {
?>
        <?= $_ ?> read_<?= uc($_) ?>(<?= $_ ?>* v);
<?    } ?>
```

becomes this

```
char read_CHAR(char* v);
int read_INT(int* v);
long read_LONG(long* v);
```

Syntax is similar to PHP, but it uses Perl instead, and can capture texts into Perl stings.

Edit by [cxw](#) — With @d-ash's approval, I am also a maintainer of perlpp. If you have questions, feel free to drop me a line!

Share Improve this answer

Follow

edited Mar 14, 2017 at 11:21



[cxw](#)

17k ● 2 ● 48 ● 83

answered Feb 22, 2013 at 15:24



[drewcape](#)

61 ● 1 ● 2

1 You should add an example on how it works.

– [António Almeida](#) Feb 22, 2013 at 15:48

This works great for me, and thanks for accepting my pull request! – [cxw](#) Dec 4, 2016 at 14:57



5



If you abstract your problem a bit, then you are in fact looking for a templating engine for your code. Just as most websites inserts dynamically generated content in static templates, you want to insert dynamically generated code into your program.



I currently use *Jinja2* (Python) for most templating work - I've found it to be *very* configurable in every way.



Share Improve this answer

Follow

answered Jul 28, 2009 at 16:50



[Morten Siebuhr](#)

6,108 ● 4 ● 32 ● 43



5



If you're prepared to get your hands dirty with some C++, there's the Wave parser in Boost, which is built using the Spirit recursive descent parser. It's a complete C pre-processor that conforms to all the latest specs for C and C++ (and, by extension, Objective C, AFAICS).

It's highly modular so you can switch your own driver in that could do the extras you want.

<http://www.boost.org/libs/wave/doc/introduction.html>

Share Improve this answer

Follow

edited Jan 9, 2017 at 14:03



Thomas Perl

2,318 ● 24 ● 23

answered Dec 28, 2008 at 20:58



philsquared

22.5k ● 12 ● 72 ● 99

Updated link:

<boost.org/doc/libs/1_54_0/libs/wave/doc/introduction.html>

perpetual link: <boost.org/libs/wave/doc/introduction.html>

– Rhubbarb Oct 30, 2013 at 10:48



4



CPP does many important things for C code that you probably don't need re-implemented. What you seem to be looking for instead may be a templating process that emits C code.

[Cheetah](#) is just one of many that allows you to use python. There are others that use python and still more in



other languages, but Cheetah is known for being output-agnostic where some templating engines are very heavily geared towards HTML/XML. Do your research.

Share Improve this answer

answered Dec 29, 2008 at 15:20

Follow



[HUAGHAGUAH](#)

1,071 ● 6 ● 3



3

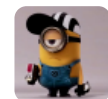


I've thought about this same problem in the past. Make sure you are OK with the fact that anyone who wants to compile your code will need the new pre-processing tool as well. If you are the only one who will ever work on it, no problem, but if you want to make the code available to others, then you might want to consider whether or not adding a tool requirement is a good idea.

Share Improve this answer

answered Dec 28, 2008 at 20:53

Follow



[Michael Kohne](#)

12k ● 3 ● 50 ● 82

I'm definitely not sure I'm okay with it. But I'm interested in looking. :-) – [Ken](#) Dec 28, 2008 at 21:54

@Ken, this consideration isn't exactly so cut and dry, for example... how many times have you extracted a tar.gz and ran `./configure` ? All the source for regenerating that was still right there, but you had no requirement to go chase down the correct versions of automake/autoconf. So all you must do, is provide BOTH the original code, and an already processed version. – [J. M. Becker](#) Jun 2, 2013 at 22:28

@TechZilla - that's not the best idea in the world, frankly. If you provide already processed code, then people will end up

MODIFYING the already processed code, and unless you are REALLY lucky, someone's going to modify the generated sections (in spite of the 'don't modify - it's generated!' warnings). Then if you need to deal with that code down the line, you have a SERIOUS problem on your hands.

– [Michael Kohne](#) Jun 3, 2013 at 0:25

- 2 @MichaelKohne: I don't dispute this concern, but it's not the problem you originally mentioned, "Make sure you are OK with the fact that anyone who wants to compile your code will need the new pre-processing tool as well." Additionally this new concern you mentioned, should equally apply to my automake/makefile analogy. Every automake dependant project would also need to address these same concerns, which was my initial point. The reason for providing both, just like automake dependant projects, is so regular compiling end-users don't require the code-generating tools.

– [J. M. Becker](#) Jun 6, 2013 at 18:01 



3



The short answer is "no." The preprocessor is so intimately tied to the semantics of C that you can't really remove it, and in fact in some compilers isn't even a separate phase like it used to be in the old days --- compiling Objective C on a Mac just parses the Objective C syntax. So while you could certainly use another macro-processor, like m4, to process your source text before passing it to C, you wouldn't be *eliminating* the C preprocessor, you'd be *adding* another step of preprocessing.

But there's a deeper question here: what do you want to *gain* by eliminating the CPP phase?

Share Improve this answer

answered Dec 28, 2008 at 20:56

Follow



Charlie Martin

112k ● 26 ● 196 ● 264

I agree that he should supplement, not replace, the CPP. However, I disagree that preprocessing is forced by the compiler. At least GCC provides a method for skipping the preprocessing stage. – [strager](#) Dec 28, 2008 at 20:58

I don't expect to eliminate the cpp phase - an additional phase is fine. I'm interested in alternatives because cpp is limited and full of pitfalls. No loops, no arrays - anything at all complex is probably just a bad idea in cpp. – [Ken](#) Dec 28, 2008 at 21:02

Then I'd be really tempted to use something like Perl and be done with it. M4 will certainly do it -- it's Turing-complete -- but it's covered in hair and unpleasantly arcane. Even compared to perl. – [Charlie Martin](#) Dec 28, 2008 at 22:07

If you feed a C sourcefile to perl, it just won't compile. You could instead make a perl script that embeds the C source as more or less one big string that gets printed out, but it'd be ugly. – [Ken](#) Dec 28, 2008 at 22:42

Dude, I don't mean feed the C to Perl. Write a Perl script that does templating using a C source file as input. Something like the Template toolkit template-toolkit.org – [Charlie Martin](#) Dec 29, 2008 at 0:39



3



You can use your favourite programming language to build a script/tool to generate source files (.c/.cpp or .h, or whatever). Simply `#include` them or compile them into your project. It may help to have comments near the `#include` to identify what/where the tool is and what is generated.

This may not be as handy (or clean) as using a "real" preprocessor, but it would work. Then again, it really depends on your case.

Share Improve this answer

answered Dec 28, 2008 at 21:14

Follow



strager

89.9k ● 27 ● 138 ● 179



1



I see a 2001 paper introducing a python-like pre-processor

<http://ray.cg.tuwien.ac.at/rft/Papers/PYM/pym.html>. It's not clear that anyone is using it..

Share Improve this answer

answered Dec 28, 2008 at 20:36

Follow



Ken

13k ● 4 ● 31 ● 32

1 The link is dead --- I have posted a mirror [on GitHub](#). Contributions welcome! – cxw Feb 20, 2017 at 17:04



I'd be interested to see what people come up with. I've tended to do small custom things with preprocessors

1



written in Perl. It's easy to rig up a Makefile that calls the preprocessor. For example, here's a rule to call a program named 'meta' to generate 'file.c' from 'file.c.meta'.

```
% :: %.meta
meta $< > $@
```

I'm doing fun things with 'meta' like generating custom fit C data structures. It's definitely a direction I'd suggest exploring. My hope is to eventually come up with a meta library roughly parallel to C++ templates.

[Share](#) [Improve this answer](#)

answered Dec 29, 2008 at 6:28

[Follow](#)



[Nathan Kurz](#)

1,689 ● 1 ● 15 ● 29