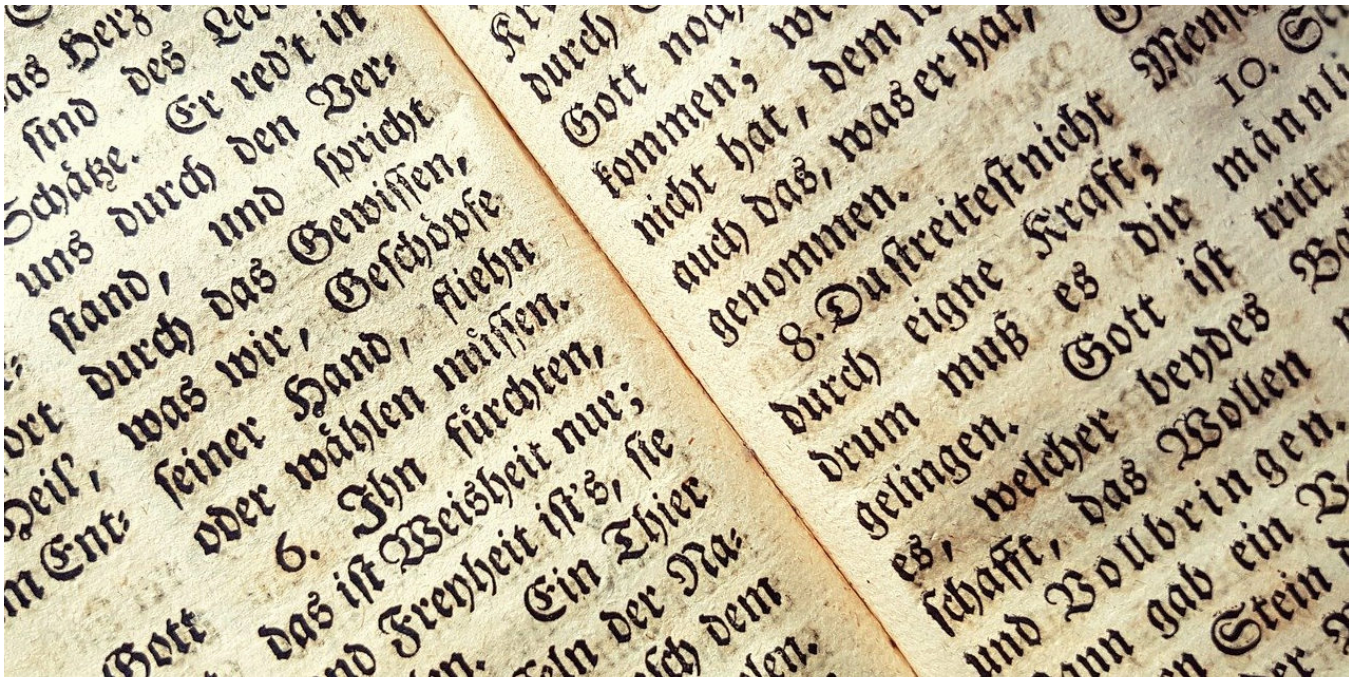

CURL AND LIBCURL

CONSIDERING C99 FOR CURL

NOVEMBER 17, 2022 | DANIEL STENBERG | 1 COMMENT

tldr: we stick to C89 for now.

The curl project builds on foundations that started in late 1996 with the tool named httpget.

ANSI C became known as C89

In 1996 there were not too many good alternatives for making a small and efficient command line tool for doing Internet transfers. I am not saying that C was the only available language, but for me the choice was easy and frankly I did not even think about any other languages when this journey started. We called the C flavor “ANSI C” back then, as compared to the *K&R* “old style” C. The ANSI C version would later be renamed to C89 (confusingly enough it is also sometimes known as C90).

In the year 2000 we introduced libcurl, the library that provides Internet transfer super powers to whoever wants it. This made the choice of using C even better. C made it possible for us to provide a stable API/ABI without problems – something not even C++ could offer at the time. It was also a reasonably portable language that made it possible for us to bring curl and libcurl to virtually all modern operating systems.

As I wanted curl and libcurl to be system level options and I aimed for the widest possible adoption, they could not be written in any of the higher level languages like Perl, Python or similar. That would make them too big and require too much “extra baggage”.

I am convinced that the use of (conservative) C for curl is a key factor to its success and its ability to get used “everywhere”.

C99

C99 was published in (surprise!) 1999 but the adoption in compilers took a long time and it remained a blocker for adoption for us. We want curl available “everywhere” so as long some of the major compilers did not support C99 we did not even consider switching C flavor, as it would risk hamper curl adoption.

The slowest of the “big compilers” to adopt C99 was the Microsoft Visual C++ compiler, which did not adopt it properly until 2015 and added more compliance in 2019. A large number of our users/developers are still stuck on older MSVC versions so not even all users of this compiler suite can build C99 programs even today, in late 2022.

C11, C17 and beyond

Meanwhile, the ISO C Working Group continue to crank out updates to the C language. C11 shipped, C17 came and now they are working on the C2x pending version, presumed to end up called C23.

Bump the requirement for curl?

We are aware that other widely popular C projects are moving forward and have raised their requirements to C99 or beyond. Like the Linux kernel, the git project and more.

The discussion about bumping C flavor has been brought up on the libcurl mailing list as well, in particular as we are already planning a version 8 release to happen in the spring of 2023 so in theory it could be a good moment to make some changes like this.

What C99 features would improve a project like curl? The most interesting parts of C99 that could impact curl code that I could think of are:

- `//` comments
- `__func__` predefined identifer
- boolean type in `<stdbool.h>`
- designated struct initializers
- empty macro arguments
- extended integer types in `<inttypes.h>` and `<stdint.h>`
- flexible array members (zero size arrays)
- inline functions
- integer constant type rules

- mixed declarations and code
- the `long long` type and library functions
- the `snprintf()` family of functions
- trailing comma allowed in enum declaration
- vararg macros
- variable-length arrays

So sure, there are lots of cool things we could use. But do we *need* them?

For several of the features above, we already have decent and functional replacements. Several of the features don't matter. The rest risk becoming distractions.

Opening up for C99 without conditions in curl code would risk opening the flood gates for people rewriting things, so we would have to go gently and open up for allowing new C99 features slowly. That is also how the git project does it. A challenge with that approach, is that it is hard to verify which features that are allowed vs used as existing tooling normally don't have that resolution.

The question has also been asked that if we consider bumping the requirement, should we then not bump it to C11 at once instead of staying at C99?

Not now

Ultimately, not a single person has yet been able to clearly articulate what benefits such a C flavor requirement bump would provide for the curl project. We mostly see a risk that we all get caught in rather irrelevant discussions and changes that perhaps will not actually bring the project forward very much. Neither in features nor in quality/security.

I think there are still much better things to do and much more worthwhile efforts to spend our energy on that could actually improve the project and bring it forward.

Like improving the test suite, increasing test coverage, making sure more code is exercised by the fuzzers.

A minor requirement change

We have decided that starting with curl 8, we will require that the compiler supports a 64 bit data type. This is not something that existed in the original C89 version but was introduced in C99. However, there is no longer any modern compiler around that does not support this.

This is a way to allow us to stop caring about those odd platforms and write code and checks for when the large types are not very large. It is hard to verify that code nowadays since virtually nobody actually uses such compilers/systems.

Maybe this is the way we can continue to adapt to and use specific post C89 features going forward. By cherry-picking them one by one and adapting to them slowly over time.

It is not a no to C99 forever

I am sure we will bring up this topic for discussion again in the future. We have not closed the door forever or written anything in stone. We have only decided that for the moment we have not been persuaded to switch. Maybe we will in a future.

Other languages

We do not consider switching or rewriting curl into any other language.

Discussion

See reddit and hacker news.

C99 CURL AND LIBCURL SOURCE CODE

ONE THOUGHT ON “CONSIDERING C99 FOR CURL”

pjmlp

NOVEMBER 17, 2022 AT 17:38

MSVC is mostly C11 and C17 compliant, with exception of atomics.

https://devblogs.microsoft.com/cppblog/c11-and-c17-standard-support-arriving-in-msvc/

They aren’t supporting any of the C99 features that became optional in C11.