
2025
STATE OF CLANG
with Aaron Ballman

C FEATURES

CLANG 20

C23 enumeration support

- non-`int` underlying types, including `long long`
- fixed underlying types

Slayed some Earthly Demons

CLANG 21

C23 Improved tag compatibility

C2y `_Countof`

C2y `0o` Octal Literals

Slayed more Earthly Demons

CLANG 22

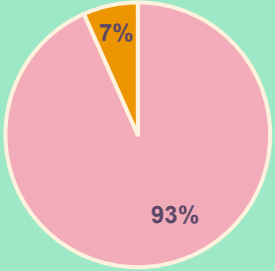
C2y Named loops

`defer` TS (in progress)

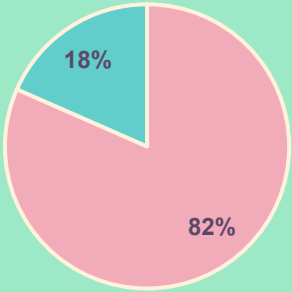
Likely to slay even more Demons

C FEATURE CONFORMANCE

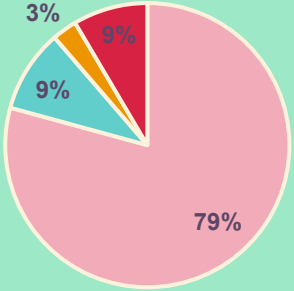
C99 As of Oct 2024
Yes: 93%
Partial: 7%



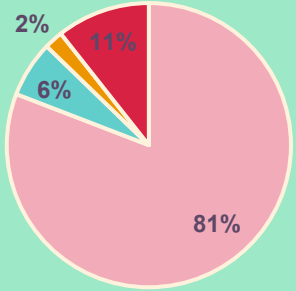
C11 As of Oct 2024
Yes: 76%
No: 3%
Unknown: 21%



C23 As of Oct 2024
Yes: 76%
No: 12%
Unknown: 9%
Partial: 3%



C2y As of Oct 2024
Yes: 56%
No: 16%
Unknown: 24%
Partial: 4%



Yes No Partial Unknown Pie charts show 2025 data

C++ FEATURES

CLANG 20

C++23 Lifetime extension in `for`

C++23 Using unknown pointers and references in `constexpr`

C++26 constexpr placement new

C++26 Variadic friends

CLANG 21

C++26 Structured binding packs

C++26 Structured binding conditions

C++26 Trivial relocatability

CLANG 22

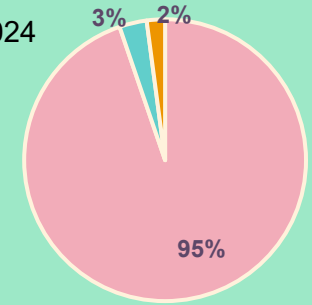
Lots of bug fixing from Clang 20 and Clang 21

Contracts (experiment in fork, likely upstreaming)

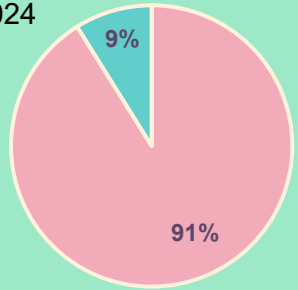
Reflection (experiment in fork, likely upstreaming)

C++ FEATURE CONFORMANCE

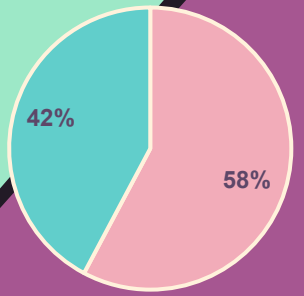
C++20 As of Oct 2024
Yes: 92%
No: 5%
Partial: 3%



C++23 As of Oct 2024
Yes: 89%
No: 11%



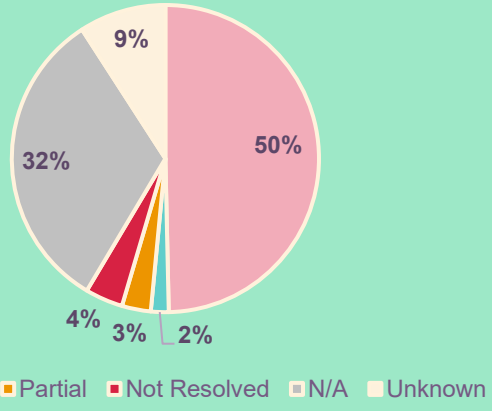
C++26 As of Oct 2024
Yes: 83%
No: 17%



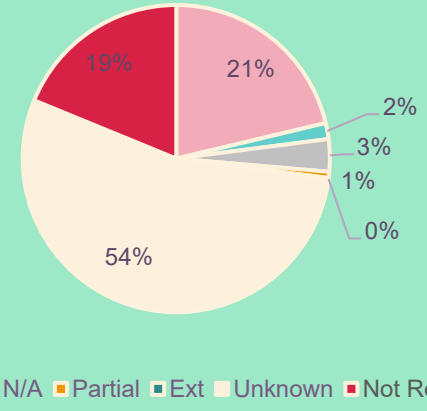
Pie charts show 2025 data

DEFECT REPORT CONFORMANCE

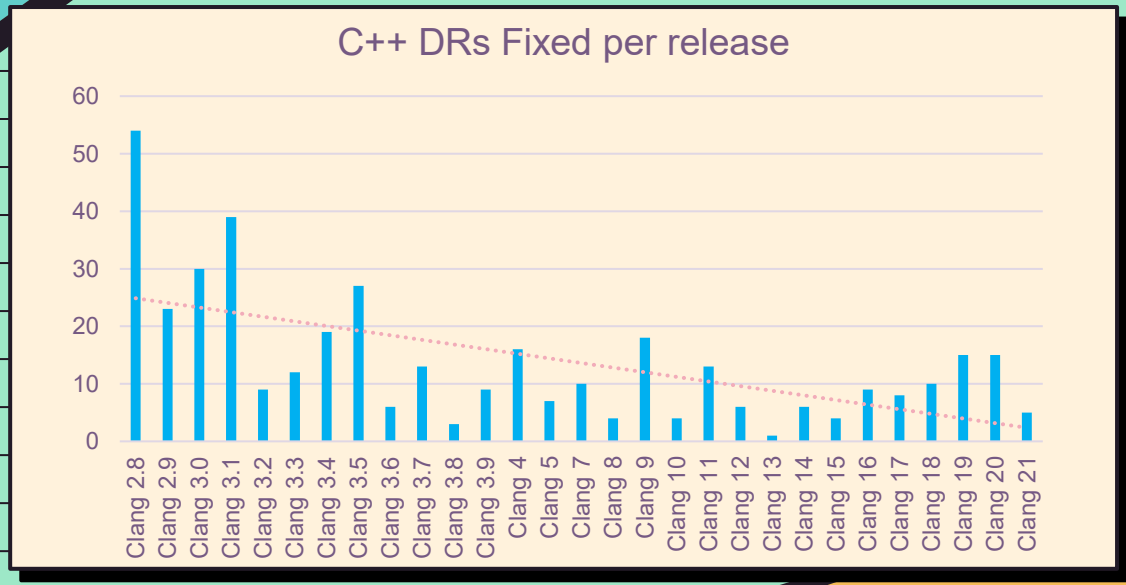
C DR Support

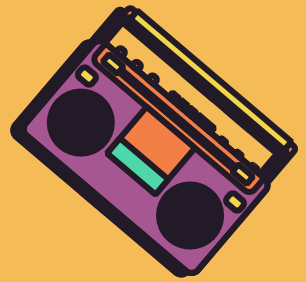


C++ DR Support

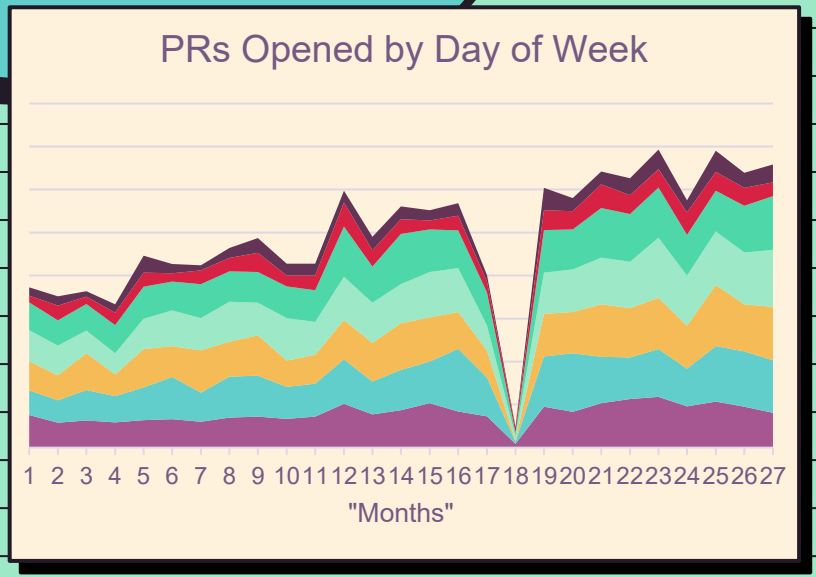


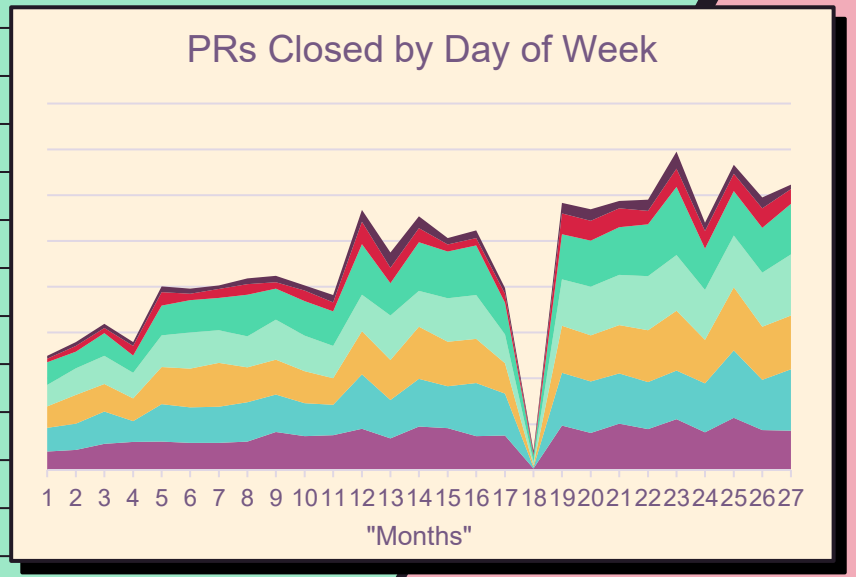
C++ DEFECT REPORTS FIXED PER RELEASE



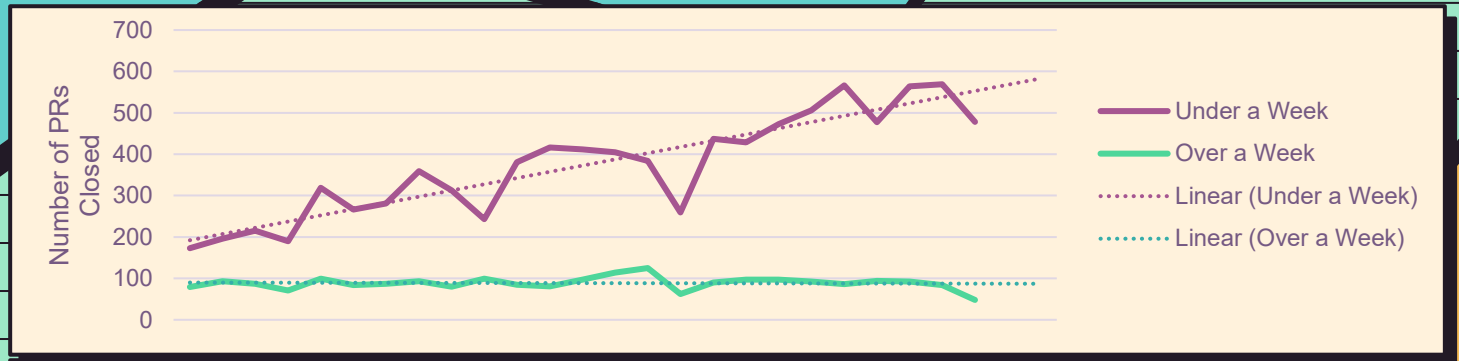


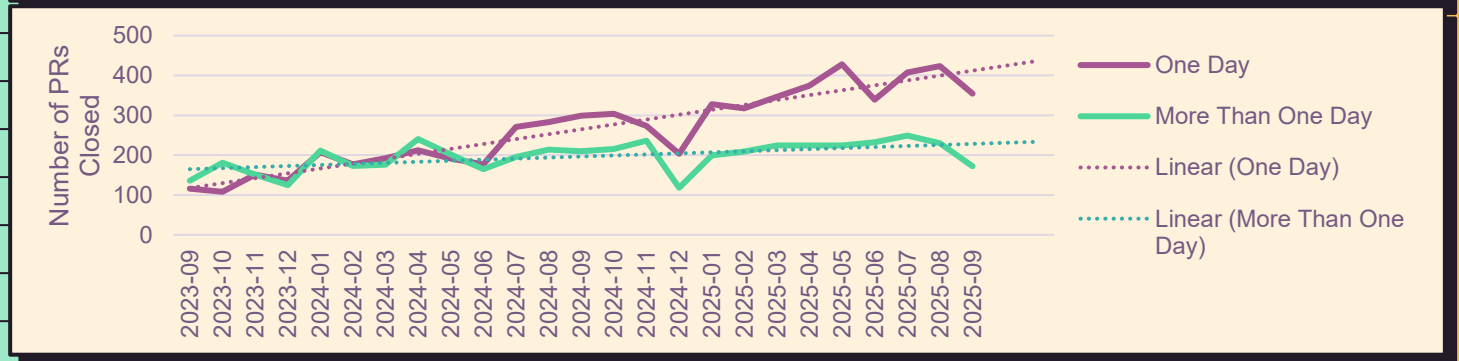
WHEN DO WE WORK?



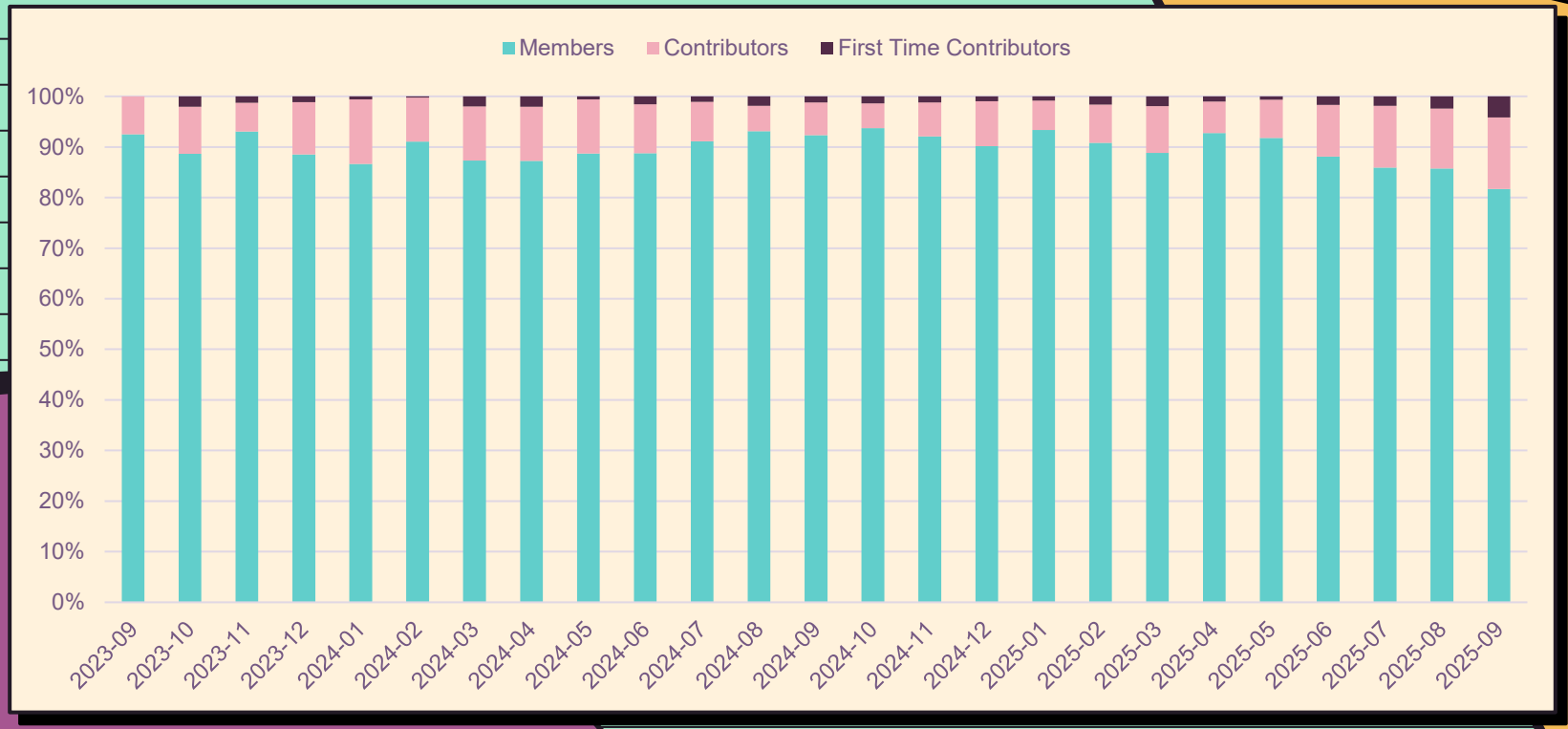


TIME TO CLOSE PULL REQUESTS

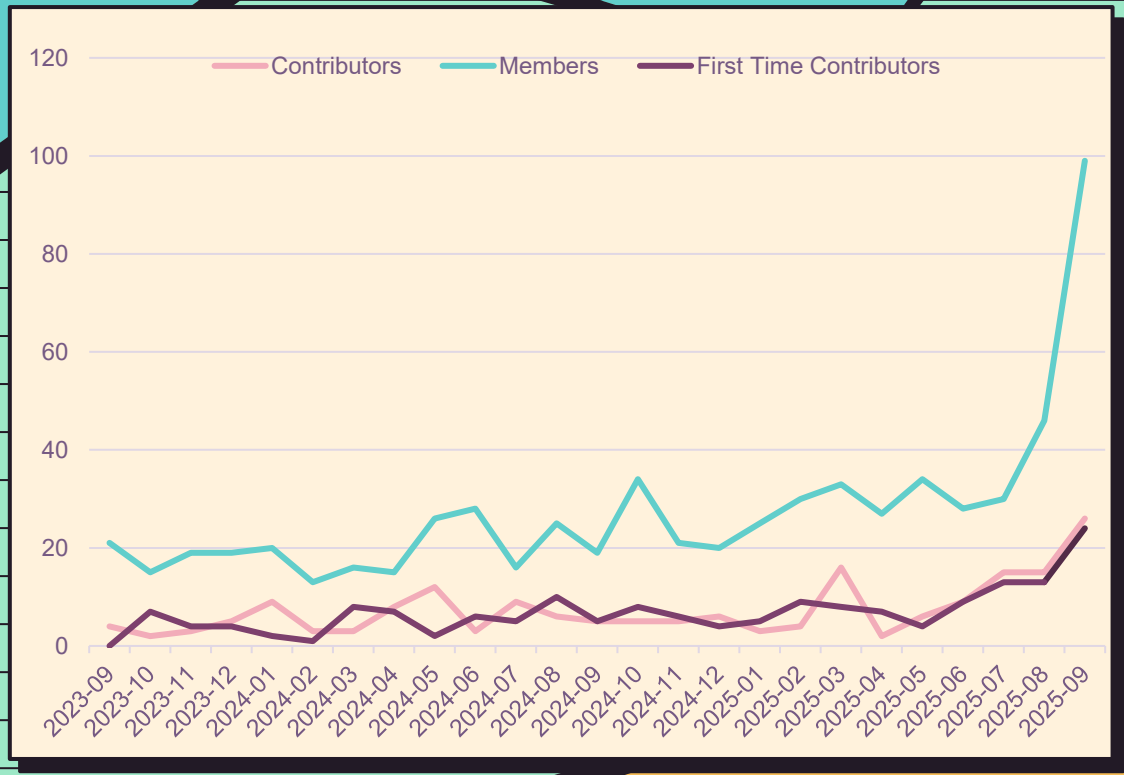




WHO DOES THE WORK?

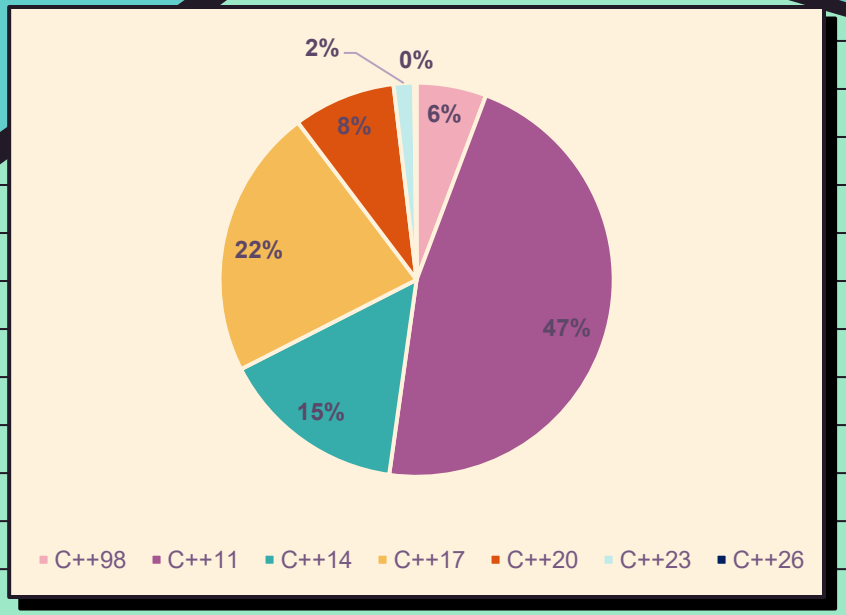


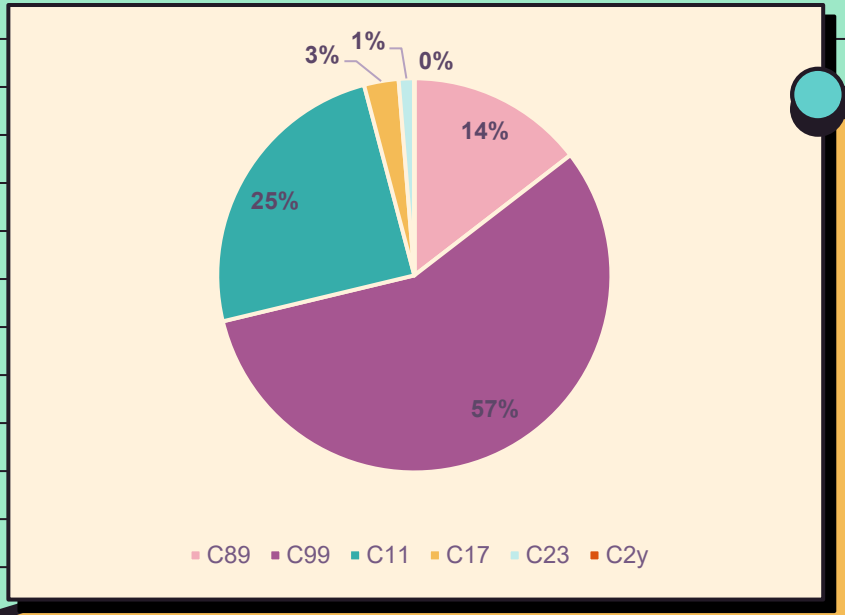
NUMBER OF STILL-OPEN PULL REQUESTS



- The last few months are showing still active reviews
- First time contributors looks like a small percentage, but compared to previous slide, **most of their PRs are still open**
- Overall trends are an increase in number of still-open PRs over time; we need more dedicated reviewers!

MOST PROJECTS USE OLD LANGUAGE MODES!





WORD AROUND TOWN

- The `defer` TS pull request has more user interactions than any other Clang PR (311 emojis!)
- Users are still excited by security improvements (bounds safety, lifetime safety, sanitizers)
- Lots of tire-kicking of newer standards (C23, C++23), however, **BY FAR** most folks are still sticking with older standards though (C99, C++11) so older code is critical to keep working

- Looking for community help with completing C++20 modules
- Newer C++ features (contracts, reflection) require significant investment
- Build times and poor documentation continue to be a significant user concern

THANKS!

Do you have any questions?

Email: **aaron@aaronballman.com**
Discord/Discourse/GitHub: **AaronBallman**