C++ standardization and Kona/Issaquah trip reports

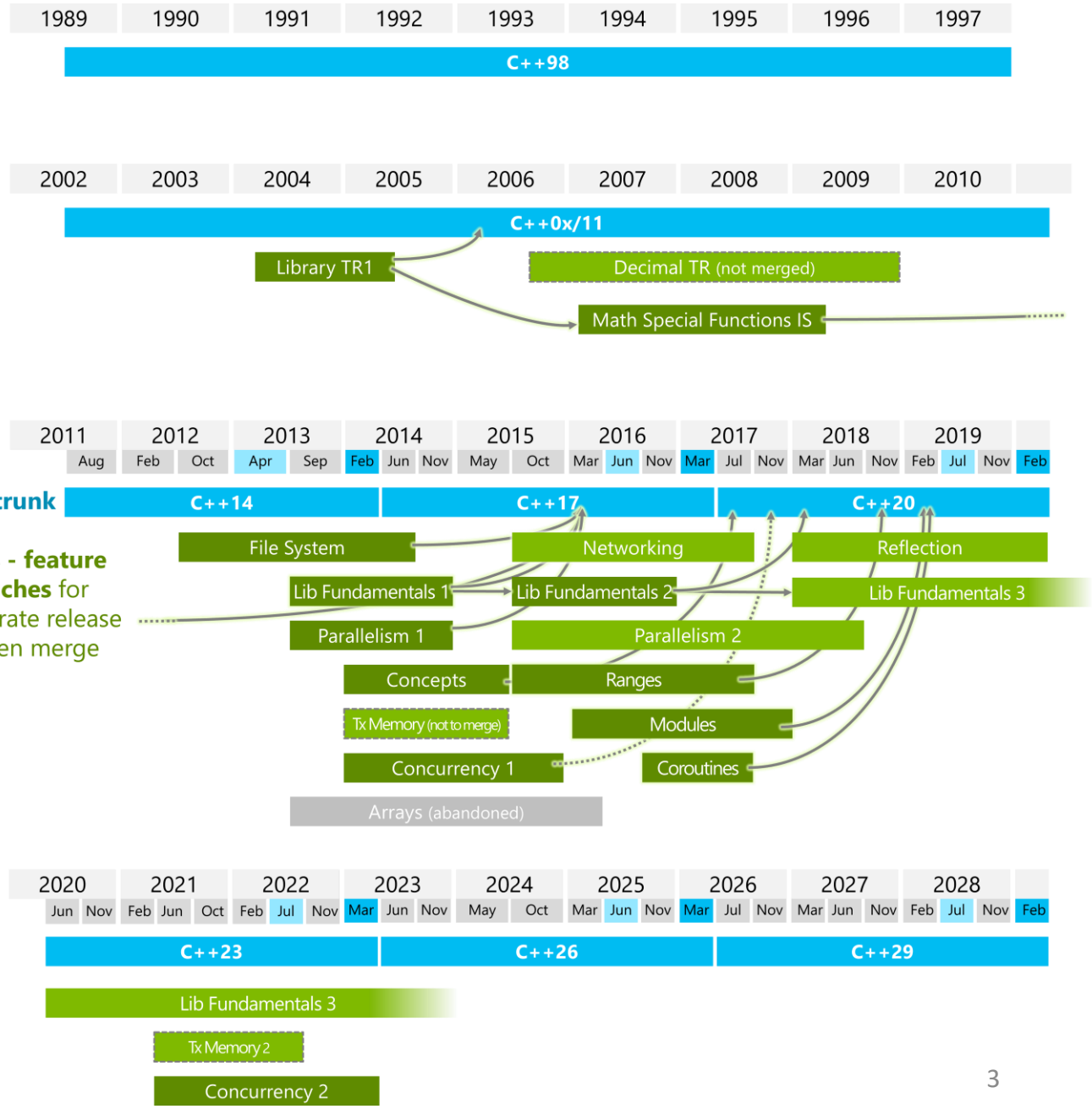
Bernhard Manfred Gruber

The committee

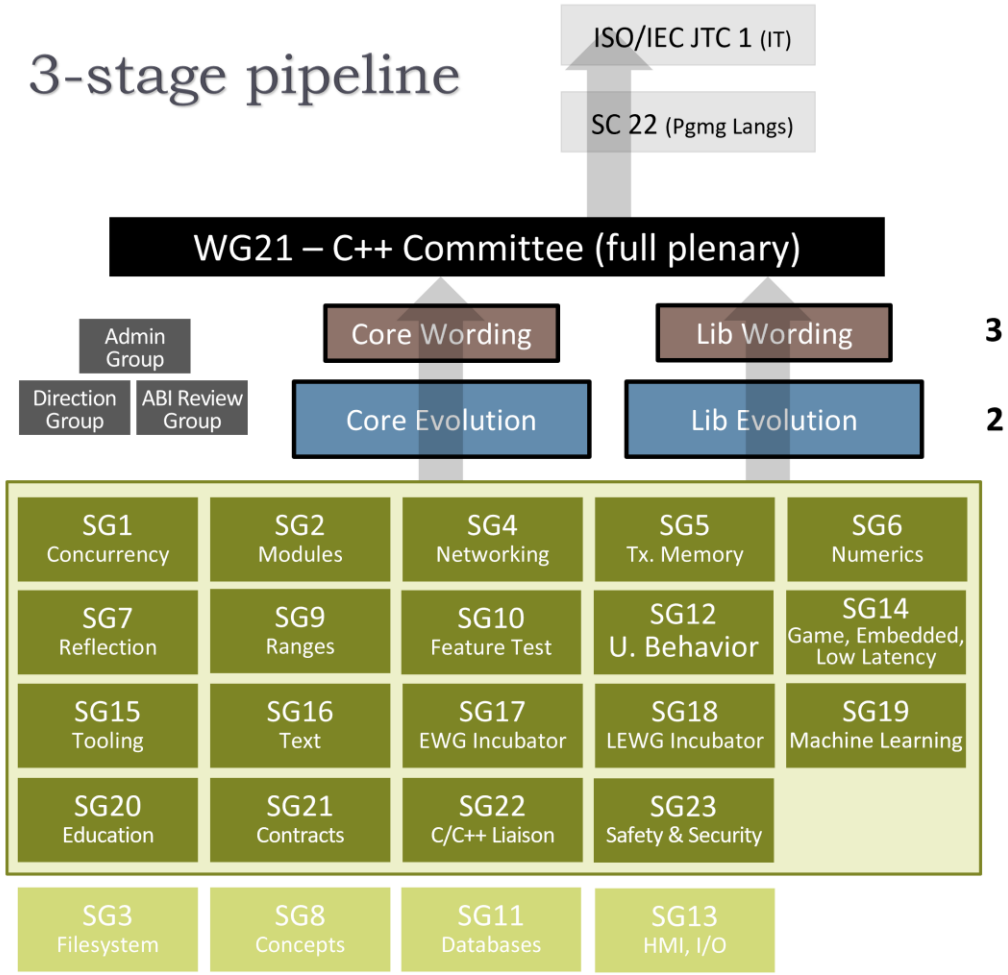
- Formally called ISO/IEC JTC1 / SC22 / **WG21**
- Consists of experts sent by national bodies (NB) of ISO members, currently 200 – 300
- Usually, 3 F2F meetings a year
- Lots of virtual ones since Covid
- News, updates, WG21 info: https://isocpp.org

Progress and status

- The train model: ship every 3 years
  - Several design meetings
  - Feature freeze
  - Committee draft (CD)
  - NB comment resolution
  - DIS (draft IS)
  - FDIS (final draft IS)
  - IS (international standard)
- Schedule: P1000
- C++ LaTeX draft: https://github.com/cplusplus/draft
  - Browsable: https://eel.is/c++draft



Study groups (SGs)



(F)DIS Approval

CD & PDTS Approval

Internal Approval

3 - Wording & Consistency

2 - Design & Target (IS/TS)

1 - Domain Specific Investigation & Incubation

Completed, inactive

Example: Issaquah agenda and rooms:

Day	Start	Break	Lunch	Break	End
February 7 Monday	9:00 AM **Plenary**	10:15 – 10:30 AM	12:00 – 1:00 PM	3:15 – 3:30 PM	5:30 PM
February 8 Tuesday	8:30 AM				
February 9 Wednesday					
February 10 Thursday					
February 11 Friday					
February 12 Saturday	8:30 AM **Plenary**		**No further breaks**		no later than 2:00 PM

group	time	room	ppl	Zoom link
Plenary	Mon am + Sat am	Catterall + Bergsma	85	https://iso.zc
LEWG	Mon-Fri, Tue eve	Barlow	24 (needed: 35)	https://iso.zc
EWG	Mon-Fri	Catterall	62	https://iso.zc
LWG	Mon-Sat	Rowley	12	https://iso.zc
CWG	Mon-Sat	Denton	17	https://iso.zc
SG1 Concurrency	Mon-Fri (mornings only)	Bergsma	24	https://iso.zc
SG4 Networking	Wed-Thu (evenings only)	Barlow	12	https://iso.zc
SG6 Numerics	Tue evening	Bergsma	12	https://iso.zc
SG6 Numerics	Thu evening	Denton	16	https://iso.zc
SG9 Ranges	Mon pm + evening	Bergsma	12	https://iso.zc
SG15 Tooling	Thu-Fri (evenings only)	Rowley	12	https://iso.zc
SG16 Unicode	---	---	8	https://iso.zc
SG17 EWG-I	Wed-Fri (evenings only)	Bergsma	12	https://iso.zc
SG17 EWG-I	Fri pm	Bergsma	12	https://iso.zc
SG21 Contracts	Tue-Wed pm	Bergsma	25	https://iso.zc
SG23 Safety and Security	Thu pm	Bergsma	X	https://iso.zc

Papers

- Are the main means to provide input into C++ standardization
- Nxxxx papers are official ISO documents
  - meeting announcements, official minutes, working drafts, editor's reports …
- PxxxxRy papers are for proposals and have revisions
- Are grouped/released in "mailings" once per month
  - Previously, mailings were sent before and after each meeting
- Papers: https://www.open-std.org/jtc1/sc22/wg21/docs/papers
  - GitHub tracker: https://github.com/cplusplus/papers/issues
- Redirect service: https://wg21.link
  - Get paper: wg21.link/pXXXX
  - Get corresponding issue on GitHub: wg21.link/pXXXX/github

Recent meetings

- The 2022 Kona autumn meeting
  - 7-12 Nov 2022, Kona, Hawaii, US (UTC-10)
  - First ever hybrid meeting, ~160 total, >100 in-person
  - Objective: resolve C++23 NB comments (137)
    - Then: progress on papers for C++26
- The 2023 Issaquah winter meeting
  - 6-11 Feb 2023, Issaquah, Washington, US (UTC-8)
  - Hybrid meeting, 160 total, >80 in-person
  - First time: official evening sessions (because of too little rooms)
  - Objective: resolve C++23 NB comments, finish DIS
    - All done, editor is preparing DIS
    - Then: progress on papers for C++26

Previously:

- (virtual) 2022-07-25: Zoom virtual plenary meeting
- (virtual) 2022-02-07: Zoom virtual plenary meeting
- (virtual) 2021-10-04: Zoom virtual plenary meeting
- (virtual) 2021-06-07: Zoom virtual plenary meeting
- (virtual) 2021-02-22: Zoom virtual plenary meeting
- (virtual) 2020-11-09: Zoom virtual plenary meeting
- **2020-02-10 to 15: Prague, Czech Republic**; Avast Software

Ready for C++26 plenary

- Structured bindings
  - introducing a pack
    - auto [a, …bs] = tupleOrStruct;
  - can be constexpr
  - can have attributes
- static_assert(false)
- std::function_ref
- Static and SBO vectors
- #embed
- std::submdspan

- More constexpr <cmath>
- std::breakpoint
- Linear algebra (C++ BLAS) after a final review
- New SI prefixes
- std::bitset(std::string_view)
- Library fundamentals TS v3
  - propagate_const, scope_exit, observer_ptr, resource_adapter

C++26 progress

- New C++ Ecosystem IS
  - building modules/header units, build system interop, portable diagnostics (SARIF) and CLI
- Many TMP improvements, inspired by Circle
  - repeated argument against: reflection will be able to do that uniformly
- No discussion on any reflection facilities
- Packs, pack indexing and language tuples
  - as a replacement for std::tuple
- std::simd: Entire day in Issaquah for implementer/user feedback (mostly Intel)
  - Plan: forward it to LWG at next meeting for C++26
- std::simd: Pave the way for (via a few ADL fixes/restrictions):
  - non-member operator[]
  - overloadable operator:?
- constexpr reinterpret_cast from void*
  - to allow constexpr std::format, std::function, std::any
- Non-transient constexpr allocation
  - moving CT heap memory to RT
- Generation of messages for static_assert
  - also: general messages during compilation
- Pattern matching
- Contracts
  - settled on MVP for C++26 and discussed potential TS
- Ignorability of standard attributes
  - standard attributes must be syntax checked
  - a non-zero __has_cpp_attribute means the compiler is "pretty gosh darn sure" to implement the recommended practice
  - paper pending to synchronize with C

C++26 progress

- Talk: The Val object model, Dave Abrahams
  - Chris Lattner: C++ has value semantics, but nobody uses it
  - Fortran is often claimed to be faster than C++, because it does not have aliasing
- New alternative to std::error_code
- Senders & Receivers (std::execution) on track
- Networking effort
  - seems dead for now, previous authors do not want to continue based on S&R
  - however: new S&R based proposal by different author

- do statements, and do_yield
  - auto v = do { statements…; do_yield res; }
- constexpr_t
  - Generalization of std::integral_constant
- C23 compatibility
- Hazard pointers
- Read-copy-update (RCU)
- Synchronized_value (a T + mutex)
- Philox RNG engine
- Statistical functions
- fiber_context
  - stackful context switching

Kona evening session: Future of C++

- Background: NIST minimum standards for software verification: “Some **languages, such as C and C++, are not memory-safe**”
- Safety is a property of operations and composes, safe/unsafe operations could be clearly defined, only safe ones allowed in some parts of a program
- Path to safety: detect precondition violations and either stop the program or throw. Or lift preconditions and define result for nonsense input.
- In any save sub-language, some programs will be suboptimal
  - E.g.: Rust needs unsafe to build an optimal doubly-linked list. Cannot have safety in C++ without sacrificing some performance.
- Defining the right line between safe/unsafe features important. Safe part must not be too complex.
- Any unsafe code can still be validated and then be declared safe
  - Kani: a Rust model checker for unsafe parts: https://github.com/model-checking/kani
- Idea: run with UBSan turned on in production, otherwise you are unsafe
  - Sony tried UBSan in production for "The Last of Us": ~5FPS
  - Most vulnerabilities not in the domain of UBSan
  - Google: running all sanitizers together in production: 50% overhead
    - Hardware acceleration can get this overhead very low
    - ARM Memory Tagging Extension
  - Running ASan in production increases attack surface
- 7% of all C++ users come from Unreal engine
  - Rust was considered for game development and abandoned

Issaquah: SG23 Safety and Security

- Prelude (mailing list): NSA officially recommends organizations to shift to memory safe languages (C#, Go, Java, Ruby, Swift) over C and C++
- Big discussions on safety profiles for C++
  - Generalizing the safe/unsafe distinction of languages like Rust
  - We want something, but we don't know what yet :)
- MS deployed some software in Rust. Main takeaway: Rust makes you rethink how you design code, once you grasp that, you can just as well write (safer) C++ again.
- Encourage more work on C++ safety profiles and features, à la [[trusted]], [[invalidating]], [[not_null]], [[check(range)]], [[check(type_safety)]], ...
- Zero-initialize objects of automatic storage duration P2723: SG23/EWG consensus
- Later (mailing list)
  - US National Cybersecurity Strategy (fact sheet): shifting liability for software products and services to promote secure development practices
  - EU Cyber Resilience Act (article): will impose general regulations on most software, based on a standard to be written, huge fines, carve-out for OSS, but explicitly calls out Linux, Chrome, Firefox, …

What's next

- C++23 is done
  - Going through final ISO ballot now
  - Implementation status: g++/clang++ almost feature complete for language changes, halfway for standard library
- Work on C++26 has begun, first items are ready for plenary in Varna
- Next meetings
  - Varna, Bulgaria in June 2023
  - Kona, Hawaii in November 2023
  - Tentative for 2024: Japan, Stockholm (Sweden), Wrocław (Poland)

So, what's in C++23? – Language

- if consteval
- Explicit object parameter/deducing this
- #elifdef, #elifndef
- Multidimensional operator[]
- #warning
- More stuff allowed in constexpr
- Mandated declaration order layout for class types
- Attributes on lambdas
- More constexpr <cmath> functions
- static operator() and operator[]
- CTAD for inherited constructors
- [[assume(expr)]]
- Better Unicode handling
- Lifetime extension of temporaries in range-for head
- Decay copy: auto(x)
- UTF-8 support as source file enc.
- …

Complete list: https://en.cppreference.com/w/cpp/23

So, what's in C++23? – Library

- std::stacktrace
- std::string::contains(…)
- std::out_ptr, std::inout_ptr
- constexpr std::optional, std::variant, std::unique_ptr, std::bitset
- Monadic functions for std::optional and std::expected
- Ranges/Views: zip, starts/ends_with, to, iota, shift_left/right, chunk, slide, stride, contains, cartesian_product, repeat, enumerate, …
- Formatting: ranges, thread::id, stacktrace
- std::format compile time format string check

- std::mdspan
- std::flat_[multi]set
- std::flat_[multi]map
- std::print
- std::byteswap
- std::expected
- std::unreachable
- std::generator
- import std
- extended FP types (std::float16, …, std::float128, std::bfloat16)
- std::move_only_function

Thanks

- Questions?

- Other trip reports:
  - July 2022 ISO C++ committee virtual meeting report, Timur Doumler
  - Autumn ISO C++ standards meeting (Kona), Herb Sutter, after Kona
  - C++23 “Pandemic Edition” is complete, Herb Sutter, after Issaquah

Backup slides

Can I participate?

- Rules changed recently
  - Previously: everybody was free to join as “observer” multiple times
  - Now: this can only be done once and needs prior notice (ISO requirement)
- Observers can fully participate and vote in SGs, but not in plenary
  - Only registered NB members can vote in plenary
- Regular participation requires joining a NB
  - Either the NB where you or your employer is situated (for CERN: SNV)
    - May be subject to fees, full voting rights
  - As alternate representative of the Standard C++ Foundation
    - For free, but without plenary voting rights
- Lots of discussion also on mailing lists (= “reflectors”), some are public: https://lists.isocpp.org
- https://isocpp.org/std/meetings-and-participation