Document Number: P1443R0
Date: 2019-01-21
Authors: Michael Wong
Project: Programming Language C++, SG14 Games Dev/Low Latency/Financial Trading/Banking/Simulation/Embedded
Reply to: Michael Wong <michael@codeplay.com>

SG14: Low Latency Meeting Minutes 2018/07/11-2019/01/09

Contents

Minutes for 2018/07/11 SG14 Conference Call

Agenda:

1. Opening and introductions

1.1 Roll call of participants

Michael Wong, Andreas Weis, Andreas Fertig, Ben Craig, , Hubert Tong, , Joseph Loser, Kevin Boissonneault, Dan Kalowsky, Herb Sutter, Lei Hou, Paul Bendixen, Mateusz Pusz, Rene Rivera, Vinnie Falco, Arthur O dwyer, John MacFarlane, Ben Saks , Dalton Woodard

1.2 Adopt agenda

Yes

1.3 Approve minutes from previous meeting, and approve publishing previously approved minutes to ISOCPP.org

Yes

1.4 Action items from previous meetings

2. Main issues (125 min)

2.1 General logistics

Review last call discussions

RAP C++ Std meeting updates
P709 was reviewed at LEWG, just 2 sections, that SG14 approved, changing precondiitons in std lib from exceptions to contracts, already doing this since Walter Brown's papers
also reviewed out of memory treat like other errors, LEWG liked it a lot, path needs to be worked out, and needs braoder opininon and review,
Post meeting mailing of paper, change the defaul new handler from throwing bad alloc to terminate
the other 2 parts was not part of LEWG, opt in light weigh exceptions, and try catch sugar, these needs to go to EWG, likely SAN, or Kona

Paul B: ranges in embedded domains, seems to progressing
seems to blow up your executable , seems to confuse the optimizer, fn calls do not get foled away, code size
is it V3, or V2 (Concept enabled)< send direct to casey carter, or Eric Niebler

AI Paul Bendixen: was looking at original Eric Niebler version, will connect with Casey, Eric, and Craig

CPpCON SG14: please sign up at SG14, only 50 seats left
FYI: same room as last time, same number of paid ticket 40 last year

Matt Bentley: std: list changes for that for std2; std2 is no more
Titus on cppcast mentions making chanegs without separate namespace, multirevision changes
OK this could work well, this is what I am looking for, remove slice, and change complexity requirements, opens up std:list to be implemented more efficiently
will rewrite the paper

2.2 Paper reviews

2. 2.1 papers by Ben Craig:
I would like to discuss P1105R0 "Leaving no room for a lower-level language: A C++ Subset"

Pre mailing draft discussion here...
https://groups.google.com/a/isocpp.org/forum/m/#!topic/sg14/xC1QeOyMDho

RTTI, etc (all in red box) none will be required in a free standing impl of C++, use feature test macro to test for it
exceptions are not required
Global init and tear down
constructor running before main, none is required in C++98 for freestanding, all impl defined,

dont want to make it required as it is not free

new stuff:
new dynamic exceptions,
throw is UB: allows people to translate exception from exception to error code,
have polls for this
if const expr false branch still has to look OK, can be just token soup, keeping exception hierarchy will allow such code to exists
Hubert: show focus on code gen here for if const expr false
fno-exception on IBM platforms need to be aware of this,

Need to find a different way to phrase this, if const expr false is close but may not be exactly right

Vinnie: what about function level? thread safe statics,
if u have a constructor then it is ill-formed,
all this is done before fn runs, so impl defined.

if exception is turned off, noexcept returns false most times, but
if u explicitly said noexcept false, we will respect that otherwise, true
this will leave the door open for static exceptions

Arthur: a bit nervous about this
want freestanding as a strict subset

Hubert: signature changing, its the tip of the iceberg, requirement on program is such that u really cant mix object files, really different world, no binary compat between hosted and frestanding

No mention of ABI compatibility, have to pick one for mixing object files, crossing library boundaries
This last bit was Ben's response. The answer was that mixing object files will indeed run into snags, but shared libraries (HT: with proper encapsulation) could work.
Herb: do u want comment on this now,
as the type system is not part of this, vast majority of std lib is pre conditions, you are doing this preemptively, and lib will catch up
if u throw bad alloc it will terminate
changing code to noexcept, chanegs it to a fn that throws pretends liek it never fails
A: I dodged the problem in freestanding, this builds on P0829 free standing, I include only things that do not throw and do not allocate, I dont propose alternative error handling just avoid them
OK you ban fns that allocate, could allow some of those,
A: provide legal extensions

slide 5: these features without an OS dont build, mostly linker warnings
pull in heavier weigh floating library that bloats your code

if you use within an OS kernel, same things will likely blow up
thread safe statics and floating point will do the wrong thing

if you use these in a C++ Signal handler, same things lead to UB

intent to have free standing get the nearly full language, allows EA, Bloomberg to just use the free standing bits that work

slide 14/15 from D & E

Herb: may have a chance for a subset, freestanding is mostly ignored,
subset is an issue with Bjarne, dont compare it with embedded C++
subset problems: abi compat, say what STl do, etc, freestanding is ignored and underspecified on purpose
provide substitute, maybe Arthur or Herb to do alternative for RTTI

Hubert: too much choice, is it for committee to choose, or options to th standard, turn off exceptions, turn off the heap separately, hard to integrate into a unified free standing, dancing around a reduced library for the language
My point is not about the C++ standard library, but the runtime support library that the core language requires. What I said was that the implications of needing a possibly different runtime support library should be made clear up front.
may be mention it near the top,
people costs
I was saying something along the lines that it would help the audience to see the costs of a solution, not just the end effects and motivation.

subset is not preferred whereas Bjarne prefers to have supersets first, before subsetting

want as much C++ that will technically work, not be a fork of what is there right now
this is not banning feature, going with optional; C++17 need almost all of the language , library needs less to be free standing
trying to aim for greater library

Ben: why nearly maximal
A: enum for execution policy,

Poll 1: get rid of free standing
SF/WF/N/WA/SA
0/1/2/9/11
Poll 2: modify along the paper, encouragement for further work, agree with most of it
SF/WF/N/WA/SA
5/13/4/0/0

does not try to block the zero overhead paper for polls 2, 3,4,5,6
please give us these features

Poll 3,4,5,6 done in reflector

2.2.2 Papers by Arthur O'Dwyer:
I have started working on a draft paper on "trivially relocatable," a.k.a. my attempt to obsolesce Niall's P1029 [[move_relocates]].
But I mention it here only for general interest, because it sounds like there will be plenty to talk about on this telecon without rushing my paper.
I am about to send a draft to Niall for his comments.
If anyone else's interest in the topic of "trivially relocatable" / P1029 is so great that they want a coauthorship and/or to help with a Clang implementation and/or to present my paper in San Diego, please shoot me an email!

Please email Arthur.

2.2.3 papers by Niall Douglas

D1095R0/N2xxx draft 3: A C Either metatype (for P0709 Zero-overhead deterministic exceptions)

Draft 2 of D1095/N2259 C _Either(A, B) proposal paper

D1027R0 draft 3: SG14 design guidelines for latency preserving (standard) libraries

Are people finding this paper more palatable then the original paper?

D1028R0 draft 2: SG14 status_code and standard error object for P0709 Zero-overhead deterministic exceptions

Working with Wg14 C to see what they can do about deterministic exceptions into C
_Fails as a new eyword
errno propagates to caller
C might add this
WG14 has a counter proposal
might need people to attend, Michael could help

Hubert: posix has fns that tries to change errno, abi implications of not changing errno is big scary without a side channel
A: backwards compat is big, and current proposal will be backwards compat binary wise too
this means a new linkage signature?
A: associate things in object file for failed functions, else mangle it
which version will get call? acosf are type generics, redispatch to acosf_fail, Jens(from C)

_Fail will take a type normally, except with errno, the fn that calls _Fail needs to set errno before it exits, Dalton Woodard likes the special magic here
earlier discussion on green threads in TLS, from his discussion on this
also wants to introduce contracts to C,

AI: michael to contact Daniel Garcia

2.2.4 papers by John McFarlane:

There are two numerics paper revisions I may not yet have inflicted on SG14:

- P1050R0 - fractional (presented in Rapperswil)
aim to avoid precision loss
interface concerns?
2/4 is not reduced to 1/2 unless you ask for it
LEWG feedback?
ratio is taken
rational and irrational
Ben: fixed point can be used in the emedded world, without floating point co proc,
not sure I see same wide usage of fractional number system

struct not a class because there are no invariants

Vinnie suggested fract,
Michael: fraction
Jan Wilmans : provide more code examaples

- P0828R1 - elastic_integer (new revision of paper I *think* we discussed in February)

2.2.2 any other proposal for reviews?

2.3 Domain-specific discussions

I would like to nominate Ben Craig as additional domain chair for Embedded

2.3.1 Embedded domain discussions: Wooter and Odin Holmes,

2.3.3 Games Domain: John McFarlane, Guy Davidson and Paul Hampson
2.3.4 Finance Domain: Carl Cooke, Neal Horlock, Mateusz Pusz and Clay Trychta

2.4 Other Papers and proposals

2.5 Future F2F meetings:

SG14 at CPPCON:

https://groups.google.com/a/isocpp.org/forum/#!topic/sg14/EbqoO03Z7JQ

I'm still looking for talks and contributions to Meeting Embedded, so maybe Codeplay is interested: https://meetingembedded.com/2018/
Update on Meeting Embedded: right now I have 5confirmed talks, with a 6th one pending. Going to make the decisions on the other talks by end of July. Maybe there is only room for one or two more talks...

2.6 future C++ Standard meetings:

https://isocpp.org/std/meetings-and-participation/upcoming-meetings

- (not a WG21 meeting, limited agenda, library processing) **2018-08-20 to 24: Batavia, IL, USA**
- (not a WG21 meeting, limited agenda, modules) **2018-09-20 to 21: Seattle, WA, USA;** Microsoft
- (not a WG21 meeting, limited agenda, executors) **2018-09-22 to 23: Seattle, WA, USA;** Standard C++ Foundation, CppCon
- (SG14 meeting) **2018-09-26: Seattle, WA, USA**; Standard C++ Foundation, CppCon
- **2018-11-05 to 10: San Diego, CA, USA;** Qualcomm
- **2019-02-18 to 23: Kona, HI, USA;** Standard C++ Foundation, NVIDIA, Plum Hall, Jens Maurer
- **2019-07-15 to 20: Cologne, Germany;** Nicolai Josuttis
- **2019-11-04 to 09: Belfast, Northern Ireland;** Archer Yates

3. Any other business
Reflector
https://groups.google.com/a/isocpp.org/forum/?fromgroups=#!forum/sg14
As well as look through papers marked "SG14" in recent standards committee paper mailings:
http://open-std.org/jtc1/sc22/wg21/docs/papers/2015/
http://open-std.org/jtc1/sc22/wg21/docs/papers/2016/

Code and proposal Staging area
https://github.com/WG21-SG14/SG14
4. Review

4.1 Review and approve resolutions and issues [e.g., changes to SG's working draft]

4.2 Review action items (5 min)

5. Closing process

5.1 Establish next agenda
Aug 15

Niall's papers + others

5.2 Future meeting

July 11: this meeting
Aug 8: Michael away; moved to Aug 15: cppcon Sg14 meeting planning
Sept 12: CPPCon planning
Sep 26: CPPCON SG14 F2F

Minutes for 2018/08/15 SG14 Conference Call

Meeting minutes by Michael

bbMichael Wong, Ben Craig, Billy Baker, Brett Searle, Guy Davidson, Paul Bendixen, Ronan Keryell, Ronen Friedman, Hubert tong, Staffan Tjornstrom, John McFarlane, Ben Saks, Niall Douglas, Jan Wlmans, Athur O'Dwyer, Dalton Woodard, Rene Riviera, Dan Kalowsky, Dan Cholland, Odin Holmes, Andreas Weis

1.2 Adopt agenda

Approve.

1.3 Approve minutes from previous meeting, and approve publishing previously approved minutes to ISOCPP.org

Approve.

1.4 Action items from previous meetings

D1028R0 draft 2: SG14 status_code and standard error object for P0709 Zero-overhead deterministic exceptions: Michael to connect Daniel Garcia on Contracts. Done and awaiting reply, Cancelled.

May have come from John,
push back from Wg14 on Deterministic exceptions
working with Herb to expand his paper, have a series of static exception
Sept 17 is the deadline for papers,to send to John McFarlane, and posted to SG14 reflector, John will maintain the list
Dxxx papers for new paper
Pxxx papers for previously reviewed (in C++ Std Committee meeting) papers

2. Main issues (125 min)

2.1 General logistics

Review last call discussions

SG14 at CPPCON logistics (John McFarlane)

https://groups.google.com/a/isocpp.org/forum/#!topic/sg14/EbqoO03Z7JQ

Doodle poll: https://doodle.com/poll/t4n2ctzqf73hri9m

also eventbrite registration, we are over 50, will get ISO tag on badge

Papers so far:

P1144 "Object relocation in terms of move plus destroy,"Arthur O'Dwyer

Linear Algebra, Guy Davidson

P1105R0 "Leaving no room for a lower-level language: A C++ Subset", will need a proxy

Sept 17 is the deadline anywhere in the world midnight

2.2 Paper reviews

2. 2.1 papers by Ben Craig:
I would like to discuss P1105R0 "Leaving no room for a lower-level language: A C++ Subset"

Pre mailing draft discussion here...
https://groups.google.com/a/isocpp.org/forum/m/#!topic/sg14/xC1QeOyMDho

Results from July 11:
Poll 1: get rid of free standing
SF/WF/N/WA/SA
0/1/2/9/11
Poll 2: modify along the paper, encouragement for further work, agree with most of it
SF/WF/N/WA/SA
5/13/4/0/0

does not try to block the zero overhead paper for polls 2, 3,4,5,6
please give us these features

Poll 3,4,5,6 done in reflector
Poll 3: noxcept should behave differently: 0/4/3/5/0, no consensus
4, 5,6 all on what to do with throw stmts when exception is not there
4: undefined: 0/3/6/3/1
5: ill-formed: 1/3/1/5/3
6: call std:terminate: 0/4/7/2/0 (least hated)

Ben Saks may be able to proxy this (AI: send ben the recording)
Jan: why not a customization pt? Will think about it

2.2.2 Papers by Arthur O'Dwyer:

P1144 "Object relocation in terms of move plus destroy,"Arthur O'Dwyer
https://groups.google.com/a/isocpp.org/forum/#!topic/sg14/6mAbZOTdVjk

Pre sent to get people to read it first

John: compare to memcpy, how?

A; smart compilers may be able to optimize this by coalescing, but no compiler can do all this
How this interact with object model and memory model may be a concern, may the OM should redirect to UB SG12
this is not Richard Smith's Bless P0593

Paul asks how this is not similar to object construction in Bless
A: need to look at the paper, but it could be related to implicit lifetime, when memory returns fro malloc, there is a moment when it is still without type;
implicit life time and trivially copyable types would be fine with Bless
Hubert: difference between reading and writing
A; when you make something to be a struct type, when you first access it, it will be a struct, but until then it still has undefined type, making it not work for network protocols, deserialiation
"undefined type" -> "indeterminate value"
AIUI:
When you receive untyped storage from some places (e.g. malloc), it'll be in this "blessed" superposition of states.
You can use that blessed storage like a struct (as long as the struct type is an "implicit-lifetime type"); if you do, then that effect ripples backwards in time and causes the implementation to ensure that there *really is* a struct there (notionally by inserting a call to the trivial default constructor, I think). But, if you were to use the storage like a *float* instead, then *that* effect would ripple backward and force the implementation to ensure that there really was a *float* there.
So any typed access causes the superposition of types to "collapse."
However, wherever you got this storage from, it had an indeterminate *value* — that hasn't changed. So even once the superposition of types collapses into a concrete *type*, it's still not safe for you to read the *value* right away, because it's indeterminate.
In other words,

```
struct S *p = malloc(sizeof *p);
p.x = 42;
```

is safe because the assignment operator has no preconditions. However,

```
struct S *p = malloc(sizeof *p);
int i = p.x;
```

remains undefined behavior because — although post-P0593 it would no longer violate strict aliasing by reading from the wrong *type* — nevertheless it's still reading an indeterminate *value*.

I just realized that P0593 would effectively make `calloc` into a synonym for `malloc` (`calloc` claims to return "zeroed storage"; this has never had a defined meaning but worked in practice; but P0593 would guarantee that `calloc` returns storage where, when you read it, you receive an

indeterminate value). Hubert, is that your understanding of how it would work? (And I'm sure this is off-topic for SG14... :p)

The question is with regards to the timing of when the lifetime begins, and the notional memcpy from a source of all-zero bytes to the memory associated with the object. P0593 does not rule out the possibility that the lifetime begins before the notional memcpy. This seems to indicate that realloc should probably also have memmove-like behaviour. I think the operator new and operator new[] cases need more thinking in terms of whether the "old memory values" may be read in the body of the allocation function and whether the new object may be written to within said body; there would need to be a clear demarcation between the lifetime of the old and the new objects.

Will be on cppchat podcast to talk about this

2.2.3 papers by Niall Douglas

D1095R0/N2xxx draft 3: A C Either metatype (for P0709 Zero-overhead deterministic exceptions)

Draft 2 of D1095/N2259 C _Either(A, B) proposal paper

D1028R0 draft 2: SG14 status_code and standard error object for P0709 Zero-overhead deterministic exceptions

Reviewed Jul 11

D1027R0 draft 4: SG14 design guidelines for latency preserving (standard) libraries

https://groups.google.com/a/isocpp.org/forum/#!topic/sg14/g8_C2ZV5pxA

get design points, blocking me
low vs fixed latency
fileI/O divided by equivalent memcpy
latency preserving: not all Std lib, not including barriers
Brett: preloading stage is not deterministic, then get a hot path once in cache,
A: this is consistent with fixed latency
Arthur: section 2 could make a great talk, offer a guideline/review checklist for people to use when building/designing library
John: as a by product of just having more efficient API will yield lower latency, similar to the idea of wide and narrow contracts benefits
Ronen: paper combines many different things of different importance, 2.3/4 are just nice to have A; for SIMD 2.3/4 is very important , depends on your domain

hence they are guidelines
Ben C: where to draw th line between strong vs weak guidelines, and when to apply or not; need a better framework?
2.1 due to how dynamic memory works, hard to get them to free memory and remain determinstic
need to pre push to cold path
instead of using the heap, can use the stack for super hot memory
caution here depending on if alloca is OK to use
Brett: to preserve fixed latency, it needs to limited
Ben C: dont want to rule out great STL containers just because of how memory is allocated
if not in hot path, not that we dont care
Paul: from embedded side, can we still use std: allocator in cold path
it is up to design of library to decide if it take no allocation guarantee seriously
Odin: if I can get what Library fns can give that guarantee
A: python is an example of entirely deterministic approach,
Niall:
Have the user supply memory to you, dont allocate it yourself

Should this guideline be published,
write this paper like core guidelines, to be added there
using motivation, examples, and alternatives, exceptions,
Can we bring this into the core guidelines

future paper shoudl restructure to look like CG on a github

2.2.4 papers by John McFarlane:

There are two numerics paper revisions I may not yet have inflicted on SG14:
- P1050R0 - fractional (presented in Rapperswil)
Results from July 11:
reflector bikeshed:
https://groups.google.com/a/isocpp.org/forum/#!topic/sg14/z2oaLOBvYDY

- P0828R1 - elastic_integer (new revision of paper I *think* we discussed in February)
P0828 elastic_integer last
time: https://github.com/johnmcfarlane/papers/blob/master/wg21/p0828r1.md

2.2.2 any other proposal for reviews?

2.3 Domain-specific discussions

I would like to nominate Ben Craig as additional domain chair for Embedded

2.3.1 Embedded domain discussions: Wooter and Odin Holmes
2.3.3 Games Domain: John McFarlane, Guy Davidson and Paul Hampson
2.3.4 Finance Domain: Carl Cooke, Neal Horlock, Mateusz Pusz and Clay Trychta

2.4 Other Papers and proposals

2.5 Future F2F meetings:

SG14 at CPPCON:

https://groups.google.com/a/isocpp.org/forum/#!topic/sg14/EbqoO03Z7JQ

Doodle poll: https://doodle.com/poll/t4n2ctzqf73hri9m

Meeting Embedded: https://meetingembedded.com/2018/

2.6 future C++ Standard meetings:

https://isocpp.org/std/meetings-and-participation/upcoming-meetings

- (not a WG21 meeting, limited agenda, library processing) **2018-08-20 to 24: Batavia, IL, USA**
- (not a WG21 meeting, limited agenda, modules) **2018-09-20 to 21: Seattle, WA, USA;** Microsoft
- (not a WG21 meeting, limited agenda, executors) **2018-09-22 to 23: Seattle, WA, USA;** Standard C++ Foundation, CppCon
- **2018-11-05 to 10: San Diego, CA, USA;** Qualcomm, Oct 8 is the mailing deadline.
- **2019-02-18 to 23: Kona, HI, USA;** Standard C++ Foundation, NVIDIA, Plum Hall, Jens Maurer
- **2019-07-15 to 20: Cologne, Germany;** Nicolai Josuttis
- **2019-11-04 to 09: Belfast, Northern Ireland;** Archer Yates

3. Any other business

Brett Serale: asking for CPPCON interviews
Michael: Bjarne presenting Embedded C++ at NDC Oslo.

Reflector
https://groups.google.com/a/isocpp.org/forum/?fromgroups=#!forum/sg14
As well as look through papers marked "SG14" in recent standards committee paper mailings:
http://open-std.org/jtc1/sc22/wg21/docs/papers/2015/
http://open-std.org/jtc1/sc22/wg21/docs/papers/2016/

Code and proposal Staging area
https://github.com/WG21-SG14/SG14
4. Review

4.1 Review and approve resolutions and issues [e.g., changes to SG's working draft]

4.2 Review action items (5 min)

5. Closing process

5.1 Establish next agenda
Sept 12

5.2 Future meeting

July 11: this meeting
Aug 8: Michael away; moved to Aug 15: cppcon Sg14 meeting planning
Sept 12: CPPCon planning
Sep 26: CPPCON SG14

Minutes for 2018/09/12 SG14 Conference Call

1.1 Roll call of participants

Michael Wong, John McFarlane, Andreas Weis, Ben Craig, Ben Saks, Brett Searles, Charles Bay, Rene Rivera, Staffan Tj, Hubert Tong, Ronen Friedman, Jan Wilmans

1.2 Adopt agenda

Approve

1.3 Approve minutes from previous meeting, and approve publishing previously approved minutes to ISOCPP.org

1.4 Action items from previous meetings

D1028R0 draft 2: SG14 status_code and standard error object for P0709 Zero-overhead deterministic exceptions: Michael to connect Daniel Garcia on Contracts

2. Main issues (125 min)

2.1 General logistics

Review last call discussions

SG14 at CPPCON logistics (John McFarlane)

https://groups.google.com/a/isocpp.org/forum/#!topic/sg14/EbqoO03Z7JQ

Doodle poll: https://doodle.com/poll/t4n2ctzqf73hri9m

Papers so far:

P1144 "Object relocation in terms of move plus destroy,"Arthur O'Dwyer

Linear Algebra, Guy Davidson

May not be ready, but ok to do discussion

Leaving no room for a lower-level language: A C++ Subset, Ben Craig (Ben Saks)

updated draft by next Monday Sept 17

P0468R1 (retain_ptr<T, R>), Isabella Muerte

Was this presented in previous cppcon?

std::byteswap, Isabella Muerte

she also mentioned some SG15 papers, but only if we have time

https://groups.google.com/a/isocpp.org/forum/#!topic/sg14/d807OXJ_gts

Deterministic exceptions
should schedule Nile and Herbs papers together

put Ben's paper before or after these

add John's numerics papers

Please get a D number for these papers

Eventbrite invites sent

2.2 Paper reviews

2.2.1 papers by Niall Douglas

Niall may not make this meeting.

P1095R0/N2289 draft 3: A C Either metatype (for P0709 Zero-overhead deterministic exceptions)

need to do this here as Niall will not be at cppcon
this is final consensus paper between wg14 and wg21, and reflectors and austin working group
_Either type is now gone, replaced by aggregate designated initializer (C)
C22 may be a target for this
C and Austin working group seems happy, SG14? if yes then there is pressure for C++23
fails call is now like an attribute
everyone wants errno function to be pure

is the compiler going to need to know special things abbout the error structure in order to return part of it through the carry flag? No dont need special knowledge

fail function is called either using try or catch... reduces complexity
fails(e) maps close to throws(e)
allows C and C++ code to interact

option of undef within CPP file to avoid conflicts

on C front, we dont have major collision concern,
if we want this to build in C++, then the fails may have bikeshedding, make them context sensitive will avoid that
consider modules as an example

paper is not published yet for SAN, but it is on WG14

what can C++ do with designated initializers?

Sorry I disconnected.

Draft 2 of D1095/N2259 C _Either(A, B) proposal paper

D1028R0 draft 2: SG14 status_code and standard error object for P0709 Zero-overhead deterministic exceptions

Reviewed Jul 11

D1027R0 draft 4: SG14 design guidelines for latency preserving (standard) libraries

https://groups.google.com/a/isocpp.org/forum/#!topic/sg14/g8_C2ZV5pxA

2.2.2 papers by John McFarlane:

There are two numerics paper revisions I may not yet have inflicted on SG14:
- P1050R0 - fractional (presented in Rapperswil)
Results from July 11:
reflector bikeshed:
https://groups.google.com/a/isocpp.org/forum/#!topic/sg14/z2oaLOBvYDY

been working on fractions more, mostly the same
lewg bikeshed may change the name from fractional
have a type that allows to express generating a fixed point number , and state how wide and how many digits it should have
also useful for performing math ops when you might do a bunch of stuff, then reduce that down to a single value, but looses precision along the way if you don't have fractional, especially with irrational numbers
do I want the bare minimum of operators or a fully fleshed out numeric type
this is mostly discussed last time

can you have complex, if you can have any numeric type?
makes sense if you define behaviour in terms of the actual implementation
is there an absolute value function, for a complex number requires hypotenuse-> what mathmatical concept can we put in here?
numerics types they are a ton of operators,

From the chat:
we have a use case for a 'to_base' free function, the idea is that we do fixed base fractional math, so we'd want to represent 1/2 as 32/64
Yes we look to support that for financial sector
Any auto handling of either num or denom overflowing their type?e.g. when defining the fraction as a pair of int16_t, should we have the option to have, e.g the mult operator decide to perform partial/full early divide if any of the numbers will overflow?
Yes this is in elastic integer paper
if they widen too far, exceed 64bit, get compiler error
i have to decide in the analysis whether to postpone the reduction, but this might cause me to lose precision
what I have is the laziest solution, so its up to you when to call the reduce function

Wanted to point out: handling the internals as floating point values brings up the whole mess of dealing with precision loss over the computations. Which seems to be contrary to the usual goals of fractional implementations
ok may be design space is too large

- P0828R1 - elastic_integer (new revision of paper I *think* we discussed in February)

may change is FAQ
presented in LEWG, there we re a few problems in synopsis
mult doubles bit, addition increases bits by 1

cheaper then checking a carry flag, you mean zero runtime cost, since the decided return type is a compiler time calculation right?
yes agreed

detect overflow on mult is expensive, because we use inverse and division is expensive
gcc and clang offer intrinsics in constant expressions

Towards a static_number

Minutes for 2018/10/10 SG14 Conference Call

Roll call of participants

Michael Wong, Andreas Fertig, Ben Craig, Billy Baker, Charley Bay, Guy Davidson,John Mcfarlane, Paul Bendixen, Ronen Friedman, Staffen TJ , Ronan Keryell, Herb Sutter

1.2 Adopt agenda

+Ben Craig, small subset of Free standing
Yes

1.3 Approve minutes from previous meeting, and approve publishing previously approved minutes to ISOCPP.org

Yes

1.4 Action items from previous meetings

2. Main issues (125 min)

2.1 General logistics

Review last call discussions

SG14 at CPPCON logistics (John McFarlane) and outcome

Logistics
SAN + KON is likely the last meeting to admit C++20 features,

Graphics:
RAP loss consensus on supporting Graphics TS
BSI wants to publish its own standard
status quo is to continue
submitted P1200 highnoon for graphics proposal on what to do next, given that some NBs want to have graphics
JF bastien has put forward response on Apple requirement, he is trying to reconnect
will not be in SAN
going ahead with plan B to implement the paper in pieces, starting with LA
Geometry next, then color, lines
there is a surprise where this was encouraged first, then there was objection so there is a disconnect

new information from new paper will ask LEWG to reconsider based on NBs request, so it can go several ways: back to SG13, back to LEWG, or ? ask to have some email reflector discussion first, to save plenary time

Will there be SG14 meeting in SAN:
No, not yet. We have trouble pulling people out of existing WG meetings especially during critical C++ 20 schedule deadlines. May be when we are out of crunch (post-Kona). then we can start more regular meeting, especially when we broadcast the expectation.
Having more SG parallel meetings will improve throughput for longer term future proposals. EWG can be focused on more immediate proposal, while SG14 can focus on more longer term proposal future proposal.

SG14 consumers are SG1 and LEWG mostly but also look at mostly long term future Future proposal.

2.2 Paper reviews

2.2.1 by Michael
Affinity in C++
need high granularity, need absolute control, dont want to rework our magic constants
code and unit test to completion our discovery algo, then the problem is solved.
compute farms slices and dices itself, loads vary

only static discovery for now, but later may be
dynamic discovery needed? resources die, or become overloaded? coming back online?

I just want one thread, and place it on one that is not overscribed
can we do this place one thread in the least contention possible? this may not be the point of this paper?
use scatter

Pipelines in C++
question about error handling
does coroutine help? Yes on fpga, different hw coexisting at the same time

big data scenario with pipeline helps testablitywith composability, can individually test pipeline stages
is used for cross simulation
can replace stage of pipelne with simulations

working on a project now where we do our own pipeline, similar to this proposal
1. assigning data
2. how do you make a scheduler,
should this be in a separate library? due to scheduling being a large piece of work

yes, this is where customization point, and how it relates to executors

tagging specific data items, through the pipeline
for HW error events, without discarding it,
suggest this for customization point

No objection to investigating further.

Ben Craig proposal
https://groups.google.com/a/isocpp.org/forum/#!topic/P1212R0: Modules and Freestandingsg14/tgMH77nm8eE
https://groups.google.com/a/isocpp.org/forum/#!topic/sg14/tgMH77nm8eE

P0581: how to layout the standard library when we have modules, forcing function for this paper
P0829 adds to the library
P1105 is on removals from core
going over options in 2
eliminating freestanding is strongly opposed by sg14
p0829+P1105 would expand the library but subset the language (opposed by some) but is recommended by Ben and SG14 poll
how to superset the language then subset
subset the language makes it incomplete
are there general facility we can add now, that once added then we can cleave, to create a clean subset

this assume proposal x, y ,z as a prerequisite
the free standing library being proposed does not require light-weight deterministic eh,
if we had light-weight deterministic eh, then we can use some of those facilities in embedded.

Look at Bjarne's recent Norway talk on embedded programming

2 basic issues not addressed for embedded C++:
1. being able to use threads (like thread attributes) help Patrice Roy
2. C++ mutexes, can't use those (unless we have priority and inheritance) get collaboration here

2.2.2 Linear Algebra immediately after this call:
https://nvmeet.webex.com/nvmeet/j.php?MTID=m1e71b8e56f2b64c972751941953ba265

Other papers

2.2.2 any other proposal for reviews?

2.3 Domain-specific discussions

2.3.1 Embedded domain discussions: Ben Craig, Wooter and Odin Holmes
2.3.3 Games Domain: John McFarlane, Guy Davidson and Paul Hampson
2.3.4 Finance Domain: Carl Cooke, Neal Horlock, Mateusz Pusz and Clay Trychta

2.4 Other Papers and proposals

2.5 Future F2F meetings:

SG14 at CPPCON:

https://groups.google.com/a/isocpp.org/forum/#!topic/sg14/EbqoO03Z7JQ

Doodle poll: https://doodle.com/poll/t4n2ctzqf73hri9m

Meeting Embedded: https://meetingembedded.com/2018/

2.6 future C++ Standard meetings:

https://isocpp.org/std/meetings-and-participation/upcoming-meetings

- **2018-11-05 to 10: San Diego, CA, USA;** Qualcomm
- **2019-02-18 to 23: Kona, HI, USA;** Standard C++ Foundation, NVIDIA, Plum Hall, Jens Maurer
- **2019-07-15 to 20: Cologne, Germany;** Nicolai Josuttis
- **2019-11-04 to 09: Belfast, Northern Ireland;** Archer Yates

3. Any other business

Lots of interest in machine learning group. Herb and I will coordinate.
What will help ML in C++
high level design reflection, differenetiation, optimizaiton, fusion

Reflector
https://groups.google.com/a/isocpp.org/forum/?fromgroups=#!forum/sg14
As well as look through papers marked "SG14" in recent standards committee paper mailings:
http://open-std.org/jtc1/sc22/wg21/docs/papers/2015/
http://open-std.org/jtc1/sc22/wg21/docs/papers/2016/

Code and proposal Staging area
https://github.com/WG21-SG14/SG14
4. Review

4.1 Review and approve resolutions and issues [e.g., changes to SG's working draft]

4.2 Review action items (5 min)

5. Closing process

5.1 Establish next agenda
Nov 14 cancelled due to post SAN and SC18

Dec 12

5.2 Future meeting

Oct 10: this meeting
Nov 14: Cancelled
Dec 12:

Minutes for 2018/12/12 SG14 Conference Call

Roll call of participants

Michael Wong, Ben Craig, Billy Baker, Andreas Fertig, Bjarne Stroustrup, Charley Bay, Carter Edwards, Guy Davidson, Jan Willmans, John McFarlane, Matthieu Brucker, Paul Bendixen, Ryan Petrie, Sergey, Steffan Tjerstrom, Thomas Feher, Arthur O'Dwyer, Kirk Shoop, Hubert Tong, Rene Rivera, Paul McKenney, Niall Douglas,

1.2 Adopt agenda

Yes

1.3 Approve minutes from previous meeting, and approve publishing previously approved minutes to ISOCPP.org

Yes

1.4 Action items from previous meetings

2. Main issues (125 min)

2.1 General logistics

Review last call discussions.

2 new SGs mailing lists for SG19 Machine Learning

https://groups.google.com/a/isocpp.org/forum/#!forum/sg19

and SG20 Education

https://groups.google.com/a/isocpp.org/forum/?fromgroups=#!forum/sg20

https://isocpp.org/std/forums

Any meeting rooms required for Kona?

EWG-I will meet Mon-Wed.
LEWG-I is penciled in for Mon-Thu, considering the outcome in San Diego.

SG12/WG23 will meet Wed-Fri - the last day being SG12-focused.

SG20 (education) would like to meet on Thursday (all day)

Suggest SG14 Friday Morning or afternoon.

SG14 Morning.

Who is coming?

2.2 Paper reviews

2.2.1 Embedded/freestanding vs hosted
Goal, paper for for Kona, or evening session

P13760:
Summary of freestanding evening session discussions
https://groups.google.com/a/isocpp.org/forum/?fromgroups#!topic/sg14/3B-Eg3cnETM

JanW : why type-id was removed from free-standing?
A: concerns about subsetting in general
PM: lot feel that they require program to run everywhere, can portability be limited?
HT: helpful if we look at video encoding standard
BC: Vulkan and OpenGL has feature set discussed
BS: main problem with subsetting is NOT because people wanted universal portability
freestanding is just one thing in people's mind and no single subset that can serve a large enough group, more dominant then any others
Situation when we cannot use alloc, due to performance, but in other cases, no floating point, or no exceptions,
So subsetting does not necessarily solve this
So how do we identify a subset. The problem seems to be multi-dimensional, and hard real-time constraints
JM: will post question

Ben Craig presents:
Slide 1:
Embdeded is not the right question
its about resources
2:
What is an OS is not useful either
some have scheduler,
3. quote from DE
what a programmer can rely on in general
4
expect tool chain to catch up

5. Hosted C++ the dark side
simd, shared memory, heterogeneous needs extensions
MW: as these things stablize we add them to standard
6. theory is everything is there
7. in practice, you have to finish the job of the tool chain
malloc, thread-local, exception, needs possibly a lot of work to make work
everyones dialect is freestanding - conforming features + varying extensions,
8. freestanding dark side
can do integer math, custom heap data structures
now 20+ years later, still no expectation anything else will be there
9. in the entire stack
who to blame, who should provide the core language C++ runtime support
GCC cannot know all
in practice, the customer application, has to fill in the blanks
10. freestanding goal
have something that means vendor can catchup to even if it is in future
11. back to D&E quote
12. problem features in priority order
highest is EH, lowest is floating point
EH needs thread local, rtti, and calls terminate when it goes bad
13. not meant to get rid of all these features, we can do shades for grey
status quo: accept non-compliance
change the feature: by not forking the language hopefully
14. suggested heuristics
3 rules that helps to guide future design
FS is signal/interrup safe, then needs to get rid of locked atomics, just use a spin lock
thread safe statics,
15. zero overhead rule from D&E

BS: what is this for? how you characterize the domain for which these things are useful or not
A: these are difficult to implement or use effectively and on GPUs
MS kernel driver does not support C++ exceptions, we know exceptions on x86 processors, is it too difficult or too expensive to implement that

type based exceptions dont know how to do it in a zero-overhead way, implementations have restrictions, like single inheritance
BS: how about using error codes
A: current environment, have to pay something for error handling, proportional to amount we use it, if there is a big call stack, those frames dont have overheads for error codes; OTOH, if we have eh, even if they dont throw all the way down, still have overhead
HT: call stack that only throws at the bottom?
A: still need exception table
JW: dont want to call it a subset, but it is a subset that we want to rely on in those environment, but BS saids it is not clear what that environment is
HT saids in terms of cost of implementation
JW/BC: prevent vendors from implementing the same stuff again for increasing exotic platforms

JW: not about defining a subset, but about what is standard implemented that any vendor can adopt
BC: most of these require knowing something about the operating environment
JW: can we specifiy better customization points?
BC: with op new delete, customization point is already there, terminate already has one, I want one at build time,
tls and thread safe statics, the protocol for a customization point is very constraining, some have per processor implementation
JW: having the customization point is not enough, need to say if the feature is there or not
BC: we say now you have to implement it, dont say how
are there any features additionally that should be on slide 12?
BS: embedded C++ in 90s failed and similar to this, again it was not well characterized what it was supposed to do, namespaces was banned
what is domain you are trying to solve, then risk falling into embedded C++ trap
BC: embedded was trying to reduce implementer burden
Strict subset, not a fork
BS: problem is when you take away feature, often you find you need it back
BC: have existing code that is operating with these constraints, none of these are operating in windows kernel, may be zero-cost exceptions from Herb
BS: MS has already 3 kinds of exceptions,
JW: new delete, global init teardown, attack these as low hanging fruits first, then locked atomics,
PB: 2 papers one adds to freestanding, and one that subtracts,
could this be a direction, before a final definition
BC: additive to standard library,
no lower level paper is subtracting features,in core language
from Niall to everyone:
Even in C, not all of C is provided by many embedded toolchains. A recent, slightly angry, debate on WG14 was over strict aliasing for example (some feel very strongly that it needs to be banned in future C standards).
Kirk Shoop presentation on Abstract machine.
heterogeneous machines accrediting overtime
slide 1:
perhaps a subsumption hierarchy
slide 2:
std library feature varies on multiple axis,
3.
similaryly we vary std library on input output traversal position
4. this is not subsetting, just capability dependent
5. then have diversity of discrete machines, we have papered over that
now have connected machines, sharing resources
6. single abstract machine have illusion of homogeneity
7. but as a collection of connected machine, then we can define each machine as an abstract machine
8. always start with algo

for every library feature, define the hardware feature requireed to operate, and organize that into hierarchy of Concepts,
define abstract machine as some permutations of supported COncepts
implement library with tag-dispatch based on concepts
implement library with awareness of these connections
9. constexpr can be exposed as an eval method in Std::, constexpr defines and abstratc machine
distributed: a collection of machines with connections
HT: prefer to run it can, and give an error where it can't, people dont want the interfaces in library that throw eh,
KS: define eh as a subsumption hierarchy a fn can be eh transparent, another would depended eh to run, or in the presence of eh it would terminate
KS: overtime we would still need this funcitonality with fewer requirements
BS: like some of it, people have implemented STL using memory pools
BC: stuff that is in templates you can do a lot of this, some in std lib not in templates, harder to do this dispatching due to static assert firing
static if is dead, constexpr if is only inside templates
some concrete cases: 3 algorithms the traditional non-parallel algo headers, if you can get a different temp buffer, in palce merge, stable sort, stable partition, can consider what the dispatching is, this can change teh big O overhead with tag dispatching
in my freestanding lib proposal, std::variant if it is in eh by value, calls visit and it throws eh, if eh is not on, then you cant into that stage
JM: like to see some examples, interesting examples,
BC: containers unordered map what those with look like
BS: need concrete examples here
quote is quoting from Alex
PB: this type of Concept, based off the feature test macros? and those would be defined for every machine type,
BS: hope not, as it increases dialects
write my program to test for hundred different things
only mageable cause you ignore them most of the time
JM: useless for writing applications, writing a library, dont know all the features you need
KS: most library dont care about most of the features
JM: can add deduction guides if can test for deduction guides, if not, then the library dont
BS: macros foul out ides
use them deep in the library, and not the primary interafce

Does nothing lower then C++ apply to Library?
https://groups.google.com/a/isocpp.org/forum/?fromgroups#!topic/sg14/Azg-X-PfmrY
no lower level language in library side,
BC: zero overhead rule stick with library side, mostly we do, a few places where we have standard library globals, so I am paying for it if I accidentally included in that header
BS: foundation lib should follow zero-overhead principle, does not mean we could not have std library, a convenience lib, that is not interested in zero-overhead all the time
BC: as long as it is global, convenient, and I use it then its fine
BS: feel some pain when I cant do a simple output of histogram, no std graphics lib to allow me to do this

such lib would not be zero overhead for any expert
the point is I dont see why we could only have one kind of graphics, one that is universal, and one that is specialized taking advantage of the hardware
one is to make the next matrix movie, and one just to do histograms, anything available universally would not be up to making matrix movies
does not need every single line of code is optimal
BC: graphics globals are fine as long as in graphics specific header, but dont include them from type traits headers
Niall had a set of guidelines that preserve the same guarantees

Final comments:
cb: eminently practical way of going forward on ben's proposal slides
we have a free standing approach and its called C
execution discipline cannot be guaranteed
ND: WG14 and safe C dialect: if we can const expr to be able to run C code subset that is safe, better then current effort on C Safe secure
BC: floating point is the main contentious points, can do it with constexpr, but not with freestanding
BS: practical real world vs academics, remember when constexpr was unimplemtable, need to idealistic

MW: do we have enough for a summarizing paper for Kona
BS: perfer someone else lead write it as coming from me would seem the same
PB: can we combine some of what Kirk has but not use Feature test macros, urge Kirk to write a paper
HT: Concepts can be used in this way
MW: urge Kirk to get with a Concept expert to explore this further
No polls were asked for

MW: Ok, if I get time, I may write this up with Ben based on the notes.
- show quoted text -
Jan 21: mailing deadline

Minutes for 2019/01/09 SG14 Conference Call

Michael, Andreas Fertig, Andreas Weis, Ben Craig, Ben Saks, Bob Steagall, Charley Bay, Guy Davidson, Hubert Tong, Jan Wilmans, Odin Holmes, Paul Bendixen, Rene Rivera, Ronan Keryell, Ronen Friedman, Niall Douglas, Maikel, Arthur O'dwyer

1.2 Adopt agenda

Approved.

1.3 Approve minutes from previous meeting, and approve publishing previously approved minutes to ISOCPP.org

Approved.

1.4 Action items from previous meetings

2. Main issues (125 min)

2.1 General logistics

Review last call discussions.

Slowly to Zoom from Webex.

2 new SGs mailing lists for SG19 Machine Learning

https://groups.google.com/a/isocpp.org/forum/#!forum/sg19

and SG20 Education

https://groups.google.com/a/isocpp.org/forum/?fromgroups=#!forum/sg20

https://isocpp.org/std/forums

Any meeting rooms required for Kona?

EWG-I will meet Mon-Wed.
LEWG-I is penciled in for Mon-Thu, considering the outcome in San Diego.

SG12/WG23 will meet Wed-Fri - the last day being SG12-focused.

SG20 (education) would like to meet on Thursday (all day)

SG14 Friday Morning, SG19 Friday afternoon.

Who is coming? Michael, Ben Craig, Ronan, Bob, ben Saks,

2.2 Paper reviews

2.2.1 Embedded/freestanding vs hosted
P13760:
Summary of freestanding evening session discussions
https://groups.google.com/a/isocpp.org/forum/?fromgroups#!topic/sg14/3B-Eg3cnETM
Does nothing lower then C++ apply to Library?
https://groups.google.com/a/isocpp.org/forum/?fromgroups#!topic/sg14/Azg-X-PfmrY

Outcome from Dec call:
In the SG14 session, he mentioned 2 that he prefers
* Freestanding is signal / interrupt safe
* Freestanding requires no special dispensation from the operating environment above what freestanding C99 requires

But there are other possible directions
* Freestanding should be as small as possible
* Freestanding has all the same core language features as hosted

Ben C: 2 issues that come up most are
a. what is the intended target, is that clear or not: kernel developer, microcontroler, gpu, fpga? TPU ?
b. if there is no one subset for all, what can we do ? will provide implementable usable C++ subset, to minimize the gap, not necessarily what every domain wants
c. Pain points: is the gap between what toolchains provide and what is needed
a paper summary can be published by Ben

DG discussion, socializing here first:
all agreed there is no suitable definiton of embedded
none of this fits within the todays freestanding definition
Language: removing freestanding as a definition, if we have a minimal set, then that would wrongly encourage maximal portability
potentially heading in a direction where we can say if FP, atomics, EH, is supported, and what facilities is available,
And have a fallback path (compile fail)
Library in supporting a list similar to what Ben's has previously

No better then what we have today? we are encouraging full implementation and that there is no difference with host
now you just get a hard error, instead of wrong behaviour
parts of hosted just not implementable, need a subset that excludes this
its actually a matter of how expensive it is to implement it, may be everything is implementable

instead of minimal bar, is it vendor query? could this fragment into 8-9 version, even harder to keep to up to date

important to point out that somethings do not makes sense. Can we better say what we cannot support
no argument on library side
language subset is problem, so can we get guidelines on how libraries can be implmented, that can get more buyin
venn diagram, very large, negative defenition vs position definiton, harder to define what is not there
group prefers a list of things that are unlikely to be there
11/14 std library is std practice, embedded domains does not have std practice
Guideline for pain points for embeded as a standing document .

"optional features" vs "features that can't be implemented on every platform".
implement everything could be bloated
implemnting everything could lead to performance issues, so enable fallback path
even an inefficient implementation is ok

potential for more studies on costs, both compile and runtime.
Is it the DG's position that a freestanding library (along the lines of Bens p0829 I believe) is ok, but changes to the language (such as removing rtti, exeptions or such) is out? I believe that would be an ok compromise. Also, that library would be so bare bones that only the vendors who are actually the target would want to provide only that (Who'd want a a deskotop c++ compiler that doesn't support vector and string?)
Evening session Kona: Monday

as part of the freestanding / exception discussion, could we discuss std::expected? It was forwarded to LWG by LEWG over a year ago, and is merely waiting on wording review. I think the SG14 constituency care about this very much, and voicing their concerns would help move things along.
we can support adding it to freestanding library, but would have to omit .value function as they throw
very similar to std: optional which also has .value
freestanding does not have a good std library error type
differences between outcome and expected.
FAQ in outcome docs
1. outcome is targeting boost user, expected for general C++ user space
2. optional api vs variant is better, outcome duplicates variant model, expected is around a success oriented path, but outcome is neutral
3. typing: expected need more typing, outcomes does not
4. expected hardcoded an exception throw in .value, outcome does not have this exception, .value can just do undefined behaviour, or halt
should add variant as the fundamental design

why after writing expected, why can't I just report a failure, why explicit constructors, Vincent does not like the implicit model
outcome is much bigger
Niall: better then nothing
Arthur: ok as a type, dual api hazard, should resist redoing filesystems with adding this api
Ben: not strongly either way
though many feel std:expected is not muli-order better
Andreas weis: complaint and had to address this need, some kind is needed
if Herb's paper is approved, then a lot of use cases for expected and outcome goes away
like iterator, multiple error handling needs similar to reduce multiple nxn explosion
unlike iterator, we only have really 2 syntax
compile time load similar to optional or more expensive due to storage, Peter Dimov feels outcome has less compile time burden
sceptical: compile time of different variant implementations and sfinae of many different constructors, expected has fewer constructors
The "throwing" issue is why we've adopted Outcome in our code
sg14 is generally OK with std: expected

2.2.2 Linear Algebra update + notes from Dec 5 (first Monday every month)
https://groups.google.com/a/isocpp.org/forum/?fromgroups#!topic/sg14/JGKsRVrzkAY
Jan 2
https://11950069482448417429.googlegroups.com/attach/60d4bfb6a587d/02jan2019.txt?part=0.1&view=1&vt=ANaJVrEPWoxHPpZ54t6CiYjZtaaNNv6rM23380AXmdr0gyxB6X5rLNiYkVM65h1TPpvpjZoqeaUgjRnsRUTe3ZVwfPvQyBQl27VnYlaN-y4j8EEcf0jWCWY
Next call: Feb 6 E PM ET

Linear Algebra Layer
https://docs.google.com/document/d/1poXfr7mUPovJC9ZQ5SDVM_1Nb6oYAXlK_d0ljdUAtSQ/edit
Paper from Guy
Concerns about identifiers using another code point: can we use actual identifiers like bit, but it would be longer to type
else we can use subnamespace,
can we use overloads? yes its an option

a third paper is 1385 on initial LA design
what is the problem against subnamespace: Titus feels it encourages users to have a high level using directive, this can lead to ADL issues, but ranges has a subnamespace
http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2017/p0816r0.pdf

Linear Algebra History from Mark
https://groups.google.com/a/isocpp.org/forum/#!msg/sg14/BG8Es5UuywU/tV5EdHGtBwAJ

2.2.3: Any serious study on cost of Exception vs cost of Error Codes

DG is discussing proposals for study

2.2.2 any other proposal for reviews?

2.3 Domain-specific discussions

2.3.1 Embedded domain discussions: Ben Craig, Wooter and Odin Holmes

Destructor paper

2.3.3 Games Domain: John McFarlane, Guy Davidson and Paul Hampson

Tooling to fill in the gap of C++ ecosystem
USe Sg20 as a way to disseminate complex features
Games reaction possibly due to over cleverness of pythagorean triples
for previous C++, similar reactions happen as they want to move slower camp
compile times is always the major complaint
debugging and constexpr, and metaprogramming techniques are hard to debug
previously was template complexity, and the extra load it puts on tools, with sfinae
embedded is less worry about compile time, but debuggability is a hugh concern, shipping code with O0
consider why Unity is moving their code to C#
It's usually not the metprogramming that breaks debugging for us gamedevs. It's inlining and variable optimization ellision that kill debugability.
Ranges can be off putting, introduces a lot of new vocabulary, they stick with well known constructs,

2.3.4 Finance Domain: Carl Cooke, Neal Horlock, Mateusz Pusz and Clay Trychta

2.4 Other Papers and proposals

2.5 Future F2F meetings:

Embo++ https://www.embo.io/

March 14-17 Bochum

SG14 meeting in Embo++, uncontacted primitive tribesmen : siemen, etc, will join, all welcome please join

2.6 future C++ Standard meetings:

https://isocpp.org/std/meetings-and-participation/upcoming-meetings

- **2019-02-18 to 23: Kona, HI, USA;** Standard C++ Foundation, NVIDIA, Plum Hall, Jens Maurer
- **2019-07-15 to 20: Cologne, Germany;** Nicolai Josuttis
- **2019-11-04 to 09: Belfast, Northern Ireland;** Archer Yates

3. Any other business
Reflector
https://groups.google.com/a/isocpp.org/forum/?fromgroups=#!forum/sg14
As well as look through papers marked "SG14" in recent standards committee paper mailings:
http://open-std.org/jtc1/sc22/wg21/docs/papers/2015/
http://open-std.org/jtc1/sc22/wg21/docs/papers/2016/

Code and proposal Staging area
https://github.com/WG21-SG14/SG14
4. Review

4.1 Review and approve resolutions and issues [e.g., changes to SG's working draft]

4.2 Review action items (5 min)

5. Closing process

5.1 Establish next agenda

Feb 14 tentative

5.2 Future meeting

Dec 12: Freestanding. DONE
Jan 9: this meeting
Jan 21: mailing deadline
Feb 14: valentines day
Feb 18: C++ Std meeting Kona