WG14 72nd meeting, Graz

Countries attending:
Austria, CA, Fr, DE?, US, UK, ~~SA~~, Sp, It, PL, Sweden

1.1 comments: convenor may switch btw rCs & AB
some questions cut number of meeting, ISO official no.
time here is to ask & answer, not present / hard sell / read; time for polls
questions will be cut off! flex upon reason

visitors: Postgres, Peter Eis. (sp?) from DE, SC32 WG3 participant
Linux Mag, Alejandro
SC1 Ryan Cord (sp?)

1.2 introduce individuals [see sheet]

1.3 straw polls as usual: one vote for each attendee
rCs determines consensus, tries to stick to model
able to use discretion
no voting by NB chat is a side channel only

1.4 required reading: CoC presented but not discussed
ISO guidance & process
IEC CoC, CoC
SC1 n2613

1.5 Approval of previous minutes N3372
AB moves, ABch second, no objections

1.6 Actions & Resolutions
Bollman: (list of retro changes) DONE
Svoboda: resubmit DONE

1.7 Approval of Agenda N3430 RB objects to changing agenda during meetings.
CB moves, DP seconds, ~~no objections~~
FW too

rCs count agenda to reflect actual discussion i.e papers moved up into section
↳ we can do but approve the sent out one, reapprove 'actual' at end

2 Liaisons

2.1 ISO, IEC, etc - SC22 meeting before Pittsburgh, no other news
2.2 NB news: INCITS meet in 2 wks, mandatory, 3 ballots sched
2.3 WG21 - C++26 feature complete, ish; pipeline may contain things not making it in - no profiles, patterns, maybe not reflection
2.4 WG23 nothing
2.5 MISRA A. Banks - [embargoed content]
more stuff for 'next' version
- [embargoed content post 2]
- misra/cert mapping updated, one in process inc. has to say collaborating more widely w/ other groups inc. MITRE
rCs: naming conventions - progress last week in discussion
recommend WG14 make no actions r/n
'we elected a fascist' - science changes affect Evolving Security FD group which was attacking C & C++
hasn't met since 2024; no large US govt opposition to C
DT: the documents do still exist at the moment on the prev. admin backup & CISA has tech docs
however much of the orgs have been disbanded
BSI in middle of 2 move
2.6 Ada in Group - nobody
2.7 Unicode - no
2.8 others none

3 Study Groups

3.1 CFP - [comments in email]
rCs: wg14 embedded TR - defect reports, but no way to maintain - consider folding TR into C2Y

are part of the TR was fixpoint math - does CFP cover this?
RB: do not cover in CFP ; no expertise in group, no benefit to CFP doing
LLVM new visitor was about to add hugg ; did want to work on TR
RB & CB may have expertise on this
Me: no customer interest RB: IBM impl it, but ... TR is still obsolete ↳ not following TR
PK: have arch support ; users mostly use third party libs, no req. for compiler support ↳ not following TR

AB: C2614 appreciates CFP's work
CP proj. still in service uses fixpoint for determinism

3.2 MoM SG
got the TS through ISO! thanks to reviewers
Sean SG hasn't met recently (Peter is chair)
↳ prefer to disband
FW here because of MoM being a mess in C ; clarifying (not adding features) is best use of C2614 ; will chair if needed
Sean will participate but not chair
see if others interested
HK original was very math heavy - can we publish as a whitepaper to not lose?
↳ no rules! anyone can write, committee will review
SC22 vote
Sean: could easily turn this into a white paper not sure what purpose served
MV: also want to continue work, point out overlap w/ URSG
FW: the TS content is the same as the math, rephrased to English 'for normal people'
TS was intended for merging to IS but needs work yet
RCs: usually changes to IS reflect impl exp ; sometimes we don't have TS golden it, sometimes good & sometimes not ; does not imply content will be merged directly ; write has total latitude to modify TS content after exp

AB TS is stable, good for impls to point at ; tried to get LLVM to do Intel stacked, out of resources; ptr diff missing but SW impl no problem LLVM maint. comment that small ptr hacks have huge downstream ramifications; 'will try their best' but if ultimately non conforming, existing uses come first & cannot regress

RB FP TS are examples of getting in - 18Z in, 3 is Ax, 4 is SIMD

FCs Annex K was a mistake to add in, C11 broke it & too late to fix can be too cavalier. TS

Sens so LLVM won't provide feedback? failed effort!
↳ GCC might get ...

RB IBM were going to, but now use LLVM

MV want to fix the int→ptr in LLVM

AB poss orders allow dropping prov. information - need to ensure propagated

Sens GCC status? → SUSE know, Rich R

3.3 C++ compl
Nina has been running, it's working

RB CFP looked at midpoint interp q. - not good added to bug, doesn't work w/ Dec vs Hex FP
not yet communicated to SG, this is the reply to LWG14

AB any liaison from CFP to numerics-SG?
Hans shows up ; open to it but no other C++ people

3.4 VBSG
VB examples doc 3455 published for review
series of Cathy Demon papers to see today
education doc not on agenda atm, work continues, likely to become a whitepaper not a TR

3.5 defer (deferred)

IS 24767

(only rCs was there)

3.6 TR 18037 - as above, Utilities published but retired. WG14 now responsible for doc. DRs & no mechanism to repair, so pull in all parts of value. HW IO & addr spaces

RB have addr spaces but don't follow

PK do follow the TR for intrinsic but not use defined

ARch missing portable use defined addr spaces - not useful to users maybe some doesn't exist

Banks echo speakers - impls not complete, don't support integration as is maybe need a part 2 w/ optional features need to re-formalize

↳ part of FS requirements? - becomes req. for all impls

RB we do ears in FS but not hosted

AB can always put in an Ax

SM can be optional in body

RB sounds like no real impls exist - need to refresh the TR? as TS? now

Me no consensus on impls at this time → TS gathered info

AB need buy-in; even on a syntactic level

RB hardware doesn't reflect TR content at all

rCs would anyone work on pursuing this?

AB, Banks, Bach; 4 1/2 people maybe
Roberto mild interest & uncertain

3.7 Infrastructure

SM would people like issue tracking? → 'yes please'

rCs asked Herb to share C++ info, who said yes & introduced 'web guy' would like C++ to pay for it - no sense of cost given, non zero WG14 has no funding or budget; will share cost when est. given

AB WG21 has info for P-docs system; licensing concern though (maybe resolved) - WG21 just uses GitHub for issues, no info to it SM has a system we can use, need something

rCs so C++ hasn't got an issue system?

↳ paper system would have value

SM also started a paper tracker -
rCs so, take the issue tracker for now
SM 'why should we have things that require money'
CB fear tying CWG/C into other tools or committees e.g. SIRA
SM using SAM's principle of 'everything into Git'
could.

Opinion - adopt SM's issue tracking sysem? 17+4/0 12+6 (Yes)
21/0/18

DT is 21 stuff open source?
↳ AB yes, don't know if all scripts public

SM need to know where to host the issue tracker
AB Todd can get you a login
rCs will continue to gather data

4 Future meetings
4.1 Pittsburgh - Dan. small city, meeting at CMU (new location)
student campus location (~~coffee~~ ~~shop~~ ~~ten~~ centre)
document shows several recc. hotels joined Cotton
haven't tested the AV
looks like a delightful location
Dan does not identify an issue w/ trans people attending

nothing after Pitts.

Me: standing invitation to Bracknell

DT attending VS/nav? → no longer a requirement, was for Doug G to not lose voting rights; retired now also only an INCITS req.

rCs wants Bangkok
AB: mae tel camp is good
Sens: meeting is useful at least once

Me can we have more, shorter? → possible but still has admin
ABakr don't need full admin w/ fixed agendas
↳ may result in reduced attendance → easier to schedule
rCs many tasks don't officially need a meeting i.e paper review can be done offline, with the will
CB 'this is a very expensive hobby for me' - financial constraints do apply

FW Nijmegen is possible, need to ask
Sen welcome back in Strasbourg
AC will try Valencia
DT will investigate Apple facility

rCs: want to lineup through 2027
SM: precise month shouldn't matter prefer off-season

5 Document Review

Ballot comments on DTS 18661-4,5 & DTS 6010

18661-4: two comments, both ed, CFP approve, ISO ed. already did
GB-001+2: unanimous consent
no objection
WG14 approves.

18661-5: more involved
GB: potentially non-ed, but unintended text change is fixed here
editor has agreed b/c mistake, considered nontechnical
just an example affected
unanimous consent
no objection
WG14 approves.

6010: four comments, three editorial
HK recommends accept all

fourth comment is Jens, queuing on doc. existence
GB-001-3 unanimous consent
no objection WG14 accept.
FR-004 (no change recommended)
unanimous consent
'comment has been noted'

(ISO have already applied recommended changes)

[resuming 3.5] n3462 defer
only need direction on a few details, mostly ready
both N-doc & TS; aim to publish TS after this meeting, get out ASAP not at 3yr limit
don't intend to start 36 - not enough open questions, no panic/rec ready for implementer feedback
will discuss paper content later on.

N3363 stdarg.h (H26)
previously seen as N3359, some precise wording tweaks
adds specific restriction to fn. compound
RB last change in 2.2 - drop footnote intentional? → don't think so
WG14 vote adopt N3363 into C2Y? 14+10, 0+0, 3+0 24 / 0 / 3 (Yes)
Decision restoring ~~keeping~~ footnote 297
DB 'variadic' word? do we introduce it now?
↳ yes as of most recent draft, can just use the word
CB 7.8 tables mixed up, interleaved text in diffs from separate changes
Jens read this as changing line items only - weird but makes sense
RB same interp. as Jens - if so, agree w/ paper
SM Jens is correct
↳ and example 3 last line? → Yes, intended to affect example
↳ so formatting is mangled
(poll here)

AC: dos b/c this pape was hard to understand; diff was hard to read

N2998 size_t suffix (N3845)
originally C++, to add via UDL feature; has z + zu literals
much more substantial benefits in C++ b/c deduction, overloads, templates
but compt is good

Opinion: could sth along lines of N? in C2Y? / / (Yes)

N3472 auto
seen several times; wording SG
AB C++ deducing arrays limits? → init_list happy to diverge
MW ordering vs. array pape? → pob establish an order so don't clash
either could go first
Jens have SG merge papers ↔ wording only

Opinion: would LG14 like sth along lines of N3472 w/o even array inference 17+6 / 4+2 18 / 0 / 6 (Yes)

Opinion: would LG14 like sth. ~~array deduction~~ with array inference, req [] in decl 11+6 / 1 / 5+3 17 / 1 / 8 (Yes)

CB like inference from init but want sq brackets
MW C++ doesn't do arrays here
↳ multi-decl? → no objections

N3484 demons V → without any objections before Pit

Decision: would LG14 adopt ~~disp~~ N3484 into C2Y? 18+7 / 0 / 1+6 25 / 0 / 7 (Yes)

AB turning compt type issues from UB into CVios
moves some stuff out of sema
↳ yes, was a definition
no non-UB code changes; no UB. hardly possible

auto a[] = { }

→ stop calling them that in future

YN clamor series)

MW only apparent - no real UB here but removing 'shall'

RB meaning of bold text? → editing mistake

vote to ed. not to adopt before PA

not a change worth adding to the retcon list

N3459 int arith & casts (+N3503)

mostly involves changing claim that 'result of' an expr is any old of expr

RB why doesn't digraph change?

↳ JM it always is, would be redundant here; can't use a VLA here to make result dynamic

Decision cont adopt first change from N3459 in C2Y? 12 8 / 0 / 0 / 6+1 7 (Yes)

MW structure of sentence?

DS word 'which' here indicates preceding subject

CB S-ICE? → not yet

second is just simplification of wording

Decision cont adopt 2nd change from N3459 in C2Y? 10+4 14 / 0 / 0 / 8+4 12 (Yes)

RB also b/c change doesn't describe diff adequately; changes no semantics
↳ JM considers editorial

AC trust the change but don't fully understand it ↳ ask that future papers have clearer diffs

FW 'shall' → carto next change for that

switching to N3503 for third change

RB would like to limit this one b/c late submission

JM only diff from prev is 'operand of a cast'

N3464

no matter the interpretation, for/class it is wrong ; another paper will handle this better ('value discarded') & CICE not needed
current defn. doesn't specify for e.g. Generic
Sens helpful to have changelogs inside papers
AB don't want to add the acronyms ; CB agrees

no poll on change I

N3377 named loops
addresses the scope issue ; also the implied meaning of a label as a jump target & consider this to harm readability
↳ prefer to force that the label cannot be a goto target
not tied to the specific syntax, want to make clear what a label is
DB 'avoiding goto' is a valid concern to avoid confusion but mostly comes from asm & Fortran allowing even more wild jumps ; MISRA allows forward & outwards → not enough justif. to add?
↳ to be clear, loop names are in
↳ don't pitch as avoiding a feature that exists ; % combination w/ defer which would not compose
AB want to focus on readability here
Jens does shadowing apply here? → yes, intend for it to apply
SL favour, syntax could improve slightly ; prefer to other one by not changing goto at all
JS author of comefrom proposal brought to LWG21 in Hagenberg (p ... r0)
AB previously wanted to put the label elsewhere — we could adapt this now to allow that syntax?
↳ two different people proposing syntaxes
JS covers a number of issues in N3377 - breaks widespread precedent does require repetition if do want both
LWG21 is concerned about contextual keywords
doesn't compose symmetrically w/ macros for loop headers that take loop body operands

label shows substantial precedent in other languages - CLang liked that
CXCT has strong consensus for semantic
AB thanks for context KLs - EK & I not tied to syntax, just to clarify
would be happy w/ any other label syntax
other languages do have distinct syntax
concerned that N3355 syntax doesn't say what the label is for in addition
RB summarize the C++ issues? → captured by Jan's paper
wording needs work too
CB originally supported 3355 b/c no novel syntax - want to be conservative
adding syntax - this looks like a macro, or a definition, or sth that
will highlight weird
labels are used in asm often to describe happens-before & misleads
actually find this usage ideal, to name things
break & cont already require a search for jump point
JL from Jan's - Rust has apostrophe sigil for labels, allows breaking
any block, not just loops
JS can have style guides that force naming conventions like caps for loops
& lc for goto
JM placing the label after for breaks macros - then the only option is 3355 pos
↳ we could also change the label syntax eg. :NAME:
↳ don't really think this is confusing after seeing a few times
APch only compare to other languages that do actually have gotos
favor in clx
could CXCT like to see a paper changing loop name syntax
Opinion at a future meeting? 5+1 / 7+4 / 5+4
6 / 11 / 9 (Abs)

N3493 → objection to the submission
only terminology changes since N3380
aims to split the ptr & array subscript operators
ignores index[array] for now, only deprecating it, not breaking it

weirdly changes so that [] does not convert to * ; adds the constraint to add
note that 'in-the-element' was already present positive index
fine w/ changing english ordinal words
CB cannot redefine eng words ; personally would use N-1
JM two consecutive sentences count in totally different ways
Me string the CC means it's not adequately clear
PCS prefer 'the subscript' to 'the latter'
FW editorial example issues
AB sort array subscript cvio - does this make inconsistency w/ parameters
that are declared using array syntax?
negative subscript into multi-dimensional arrays is used in practice
→ had this discussion before
Jens: we agreed this was OK to do to pointer-typed operands
constraint is on operand type, not underlying storage
this was always UB
AB: code does use this UB
so you can use a negative index on an 'array' param → could fix
Jens we already have many such problems/incons. w/ params - sizeof, void*
↳ voted for it in MN!
DT was this used much?
↳ not much for array types, widely for ptrs, which still work
RB temperature changes often
we have reverted votes before, esp w/ new information
negative indices in multidim arrays are commonplace & will break
↳ never seen any compiler do the wrong thing
CB don't think should worry bc part that continues to work
MW UBsan will diagnose this stuff already
would be good to expand rule to params
count cardinality
Decision adopt N3493 into CCY, subject to any editorial changes?
8+3/6+0/4+4
11/6/8 (Yes)
(delay per objection) ↳ not adequate consensus

N3464 'discarded'
builds on N3382, w/o repetition
relies on some extended definition of 'not evaluated'
appears in both exps and type names, hence two terms - types 'resolve', exps 'evaluate'
changed approach after SM feedback - 'discarded relative to' for shorteric exps & temp; only operator based are handled
stmts are a possible extension
discarded doesn't map to 'not evaluated' b/c always have the split
in wording for sizeof & handful missing words, updated by N3504
for exps c/in types c/in sizeof

Decision want ~~adopt N3504 as is into CW?~~ / / ( )

[delay adoption pending objection]

AB vote 'time name' typo → editorial
SM want a new version & only vote along the lines was
would 8th along lines of N3504 in CW? Opinion 12+0/0 / 12/0 / 5+6/11 (Yes)

AC this is not adequately clear, needs a lot of polish
RBS would have been no - not adequate improvement

N3465 prepo exps
'any questions?'
RCB looks like high chance of breaking use code - uses write surprising things
AB this would forbid builtins in preprocessor? may not result in expected
Me impl detail that stuff exists after prepo, it's an extension anyway
RB people use extensions all the time IRL
CB whose problem is this?
SM: for definition of 'discarded', avoid complexity

AB agree that this is a theoretical problem
count

Decision: adopt N3465 in C2Y? 10 (9+1) / 2 (1+1) / 12 (7+5) (Abs)

Jens not clear that this comes after macro replacement & undefined into 0?
SM in context, this comes after all replacements have occurred
conv ppdirto tok is already in Std
Myps ppdir only goes to one tok, or ill-formed

RCS high Abs → AB doesn't care
RB this is weak support in favour
impl exp? → all three opinions

N3473 if Ref. 28625
AB storage class - prefer to disallow
CB can use ude for c/ & w/o Constraints?

Opinion: w/ constraints on if(decl)? 9 (7+2) / 7 (5+2) / 12 (7+5) (Abs)
↳ direction to retain Constraint for now

Q from Glenn: what about skipped ppes?
↳ will try to accom bler in work; not all authors may know

Tuesday

N3390 I
MU favour total removal
AC remove from macro ns - UB for use to #define?
↳ yes if has linkage
RB CPP - rel n3452, fixes example cproj
needs fi suffix, not i
#define I _CI should be in Std as example but happy to remove I

Opinion: remove as macro + add example? 17 (14+3) / 1 (1+0) / 9 (5+4) (Yes)

Homework w/ wording for this

N3432 composite types
originated from the size/length debate - led into array composition
a const array of NCE is VLA itself; any non-VLA has fixed size
seeks to eliminate a UB and changes normative
↳ unevaluated array size UB
adds rules to clarify fall through of steps, missing UB stages
clarifies that unspecified and unknown are distinct

SM these examples are incorrect b/c use func decls - need to use definitions
or else the size is unspecified ↳ & void return

Jens changes priority of resolution only, not even case branches

MU only in this first case w/ UB - 1 other only clarifying

RCS (new change? → RB see next first)
cond op comes from MU directly, makes UB a CWG
↳ only short edge case where the composite of unevaluated VLA types
appears in ?: — compilers don't even handle this, crash
(where type needed from unevaluated side)
worse where both sides are VLA

AB 'below' not 'all'!

RB text says 'paired with' — not used elsewhere, composites don't generally pair - use a diff't term

RCS also a second version that doesn't fix the UB

AB ex 15 - what happens when 1 typeof(condition)?
SM - typeof evaluates VLA typed arguments

XM can already have this w/ sizeof, same handling

AB isn't this currently UB unless the same?
↳ UB if not both evaluated, UB if a VLA branch requires the other one to be evaluated

MU could make more explicit in example

SM can't use 'below' in ISO wording

CB so what is the decl type on the left? → depends, same as size rules resolve to

so it's the size of the active branch Jens: note number of evaluations too

MV clear eval only happens in active branch

CB can the compiler notice a type mismatch if the cond is different?

MV that's UB because of a runtime mismatch of VLA sizes & types

AC agreeing that we use active branch in second & uncond. constant in first?
↳ yes, not great but consistent.

AB would be better to be a CWio here?

MV cannot easily determine statically whether they're the same
see dep type paper, could do much more work

Me want a CWio here - but fixing UB is an uncontroversial improvement

PK so second is UB? → only if mismatched
cf. in N3432

Opinion would like to replace the UB with a CWio? 18+2 / 20, 1+2 / 3, 3+1 / 4 (Yes)
as described by sec. 3.1

AC not happy about diff. beh if const or not

MV if not eval, there is nothing to define

AB can see users running into UB if they delete a constexpr to add portability
↳ right now code is just broken here

AC so 'active branch'? → propose picking the evaluated size
so this can be changed by unrelated changes to size expr

Jens r/n that line is UB and broken

AC at least UB can be an error - don't want a defined confusing behavior
users can turn warnings off - why not make CWio & define bte?

MV will think more about this example line

AB yes like Jens & they can catch this right now

Me users do turn off warnings if they can justify it

RB Clang doesn't diag this; IBM crashes on these examples

PK don't think we'll impl this right anyway so UB helps us - making these defined is effort for no reward
MV - small effort in GCC

→ prev N3414

N3495 array declarator terms

'size of an array' is ambiguous wording

RCB new paper? → difference is only change marks, non-normative

proposed normative changes should not affect real behaviours

names of terms only (accuracy)

size goes to 'number of elements' ; bullet points became explicit

accuracy to the grammar in the text

mainly a text cleanup : 'length expr' is also an option

RCS in MN we used 'length' - didn't like that but eventually conceded

we have flexibility now, agreement not to use 'length' here makes it

clearer in local ctx ; 'assignment expr' is confusing in the text b/c

looks like an index expr ; used those terms consistently in prev paper

CB agree that asgn expr is confusing ; some syntax rules are just renames

for clarity - useful to reorder here

AB object b/c spelling out grammar in prose is unreadable - use font choice

to clarify vs. Eng. nouns. editorial change

↳ big project. agree on use of asgn expr - good to clarify & may not

achieve this goal. Too much for a 'HW' item

RCB agree on all that ; in 3.3, pb - drop 'for' ; what is 'it' in 'it shall

be the same' → clarify this

AC like this, doesn't conflate changes. minimal

Opinion — would sth along lines of N3495 in C2Y? 19+3, 0+0, 3+3 / 22 / 0 / 6 (Yes)

N3432 again

Decision — count changes in N3432 sec 3.2 in C2Y? / /

RCS VB shift needs more work, 3.1 not ready

SM both versions need work

RCS ok no more direction c/n

NO POLL

[N3485 revision]

N2998 resumed 'you didn't survive the break'

RCS go straight to Q&A, resume yesterday

AB in 3.2.2, can we just say size_t here, instead of 'the exp... type?'
↳ much prefer that

RBJ C++ went through & ok, but what is the real problem C users are hitting?

AC - run into this comparing constants to return values; allows not having a cast

AB useful for variadics too

A.Bch don't have suffixes for other types, inconsistent

CB not convinced C++ rat. applies to C; ideally avoid casts w/ literals but prefer not to care what the type is; never had prob. w 1UL

RCS really 2-r about programming; want my init to have the type of the object

Me such has impl burden it will outweigh

CB just don't want to have to write 1z etc - compiler knows the value

RCS getting a literal wrong in a header breaks uses forever

? useful for generic dispatch

[SL echo in the room]

AB missing lowercase u in table
↳ SHM this sort of thing is fixed in the revision, duo size_t, Ref comments etc - not the version in play but changes apply
did paper predates CZ3 - new wording has mixed suffixes that allow now
also the note about 'signed counterpart' - not naming the type though

AC think CB is right to not want lit. suffixes - maybe compilers could be less overjealous about literal comparisons

SHM const val only applies to storage - conversions must still apply in exprs
storing a literal is fairly harmless, stranger behaviours where other types & operations apply

Me MISRA 4 sys.

CB GCC / Clang allow much of this, eg. =-1 which fits into any type

Sens favour adding. Want vote

Decision: ~~adopt N2998 subj to ed. chgs from N3485 in CTY?~~ want

Opinion: could sth along lines of N3485 in CTY? 14+1 15 / 1+2 3 / 6+5 11 (Yes)
'majority support'

N3337 – Record
solves a prob. from last cycle w/ tag compatibility; unfinished px, still don't have fully generic DSes / dg types - still have repetition too to avoid redefs
solves the N3037 prob & also the more general improvement to the type sys proposed wording not considered ready to integrate

CB: core type variance paper - didn't worry about structs, would need to apply
don't much like syntax; header file sharing is a super thin justification, not against compt, but messy that might add unintentional compats
semantics come from types, this exes

SHM: more that so many redefs of a type intended to be 'the same' opt redef but it is intentionally opt-in, communicates intent
KW applies to structs or unions, isn't a third kind of tag/redefs
↳ fear that people will misuse

A. Bch: worthwhile goal - don't want to add grammar, want similar types to just be interchangeable

RB: opposite POV - don't want that, use gives tags for diff reasons
like the opt-in nature, allows not breaking existing code, very impt & reasoning for new KW is clear, b/c opt-in

MV: think this is too complex - feels like adding complexity back in
want to wait for uses to experiment more

AB: agree cat. complexity - don't think it scales, can't keep reb of types separate from one another, need added parts to subset it

Ryan: agree w/ cmts, would have been nice to have for vectors & cplx

CS: backwards - people already do this, cast stuff & treat structs as offsets
hard to explain to uses already doing it that way
do agree w/ Rajan - would prefer a KW to get more

Me: [support RB] [+ CB]

CB 'Rich's suggestion terrifies me' - structs are the answer for type safety
YN what happens to diff't attrs on Record fields - would they compose?
SHM think they would need to be same but wording may not make CWG at moment - intend to require exact match
AC why not an attribute? can't ignore packed
AB [vendor attrs aren't ignorable, but Std ones are ignorable
↳ why not make not ignorable? ↳ important.
SHM tried that in C++ - broke between GCC/MSVC [[no_unique_address]] caused much trouble by materially changing behavior of code, breaking model
a KW can do that & is explicitly opt-in; w/ attr it becomes disdechiffrable
AC so KW would make things compatible; ignoring the attr would be like not having the KW
AB a KW forces conformance vs. nonconformance
strongly oppose adding non-ignorable attrs
DI purpose of attrs is to assume they can be ignored, hard to reconcile w/ C++
CB worried about C parsing & attrs make it worse
FW is it UB to cast btw struct types now? → no meaning rn
RB pointer casting may not be valid but can be done
SHM have meaningful feedback - AB point that this is viral, needs to address subsetting; people seem to want diff't spelling
AB would like poll, get sentiment

N3451 anon structs & unions
standardizes behavior already impl. by GCC et al
want direction for which way this should settle
RB thought N2038 fixed this, CCS has this integrated
↳ think this is a distinct issue when anons were added - covers bgapt, not init
↳ 6.7.3.2 p15 changes do seem to address this?
Hlee explicitly defines as members of the containing obj. - considers the language to exclude unnamed mbrs

SH not sure that existing wording reflects an intentional decision - dates to C99 & would have referred to bitfields, not structs

Sens want PK opinion

PK don't do frontend work; SOCC is different & users don't rely on current beh. can easily change to Std & add a warning

RB Clang does diff't things on the provided examples - this means existing code would change
↳ intend to match existing behaviour
↳ doesn't appear to actually describe current behaviour

CB really like this! well researched

Ryan improves portability, good change

AB confirm that Clang doesn't behave as described, while does GCC
↳ 'behaviour B' - don't believe this is conforming w/ C23
so there's a difference described btw observed & std behaviour

Decision — vote: adopt N3451 in C2Y? 17+4 / 1 +0 / 6+3 (Yes) — 21 / 1 / 9

RCS Apple object → no, just weren't sure

N3409 demons X, void expr
'extern void x' is well defined, somehow

PK want to allow returning of void exprs by void fn
↳ proposed by TCE TS

SM convinced against in the description of the return stmt

DS posted example - is this the same UB? - possibly not

Decision — vote: adopt N3409 into C2Y? 17+5 / 0 +0 / 3 +1 (Yes) — 22 / 0 / 4

CB is this in the Annex? → maintained by editor, changes are implicit

genericsinc.info

NZ698 Mels parametric generics
[presenting slides] - unfinished with macros & also w/ void*
- principal diff from VMS is monomorphization
- Any is fundamentally sim. to bind type
↳ overall goal is type safety
↳ except b/c monomorphization, it can be a value type not just pointer type
inst requests & pending requests ↳ slightly better integration w/ structs than
distinct from C++ instantiation, better design for absolute clarity, not 2 phase
examples are accompanied by questions - do we want to be accepted
phased instantiation - use vs. deferred (distinct from VMS which has no phases)
↳ iterated substitution is significantly more complicated
in particular it is possible to write C++ style exps that do not substitute correctly e.g. member decl of an Any()* might not exist
↳ this is the same as C++ templates issue!
constraints applied in body not at point of inference
'not 100% formalizable'? has been prototyped though it looks above
↳ subset of C
compatibility is of the monomorphized program, not of the polymorphic one

DT does impl require code cloning? do you have to copy code
↳ yes, you get monomorphic code back [this is VMS]
↳ did you consider constraining to only incomplete types ↘ it can be done
↳ no. didn't know has to type check this
PK mention Jens's autos - what are adv/dis of this vs. that, which is simpler?
↳ do not remember, need to see... ~ cited in Sec 5
↳ specific to same form of genericity & all have smaller scope, tightly scoped & not broad kind of generics
did think MU's Any was related but achieved diff't goals - runtime version instead
CB excited by idea, not sure abt it - don't know about way of binding struct members, what's with the _; does this work w/ init lists
can this coexist w/ prepro? is this a core lang. change

Melo could n't see ctx in which types of parametric fields are ok
CB say two-mem struct - difficult to say both mbrs have same type, need to bind two names distinctly
↳ Melo: either unified or not; if they have to be the same
(↳ so, this works like C++?)
could give names & express relation; not expressed in syntax now
n3415
N3499 reproducible exprs (extends n3597)
considering context of VLA evaluations could be today: no problem to evaluate a type expr w/ no side effects; usually such effects are an error
sequencing makes many such exprs erroneous or badly defined
use unfriendly to simply make such effects UB
new category in btw constant & non-constant exprs; can be a syntactic category to ban fn calls ↳ add before consts & derive syntactically partly syntactic & partly statically semantic (repr. calls only)
↳ option to allow these forms inside sizeof etc but do not want to b/c less understandable
↳ subtle use of grammar rewrites for clarity
RB seen plenty of code that uses the side effect, common one is debug info breaks a real use case
↳ debugging does involve a lot of UB; question is really if the value expr does it on purpose in a way that could not be hoisted
RB never impossible to hoist, but make less clear. example of values that are off by one → see for subscript, sure for dcl?
↳ not sure if it's used in both
CB main mixing (share RB's) - although tries to make C safer, people do this; are we sure there is no utility? e.g. fn calls as args are generally allowed qualified ptr args are not sufficient & prefer not to have mention here
Sens want constraint as-if decl. w/ [[repr]] - restrict & const are req. by this
↳ side effects not restricted to this?
↳ many more restrictions than these ↘ not sufficient, relevant
SM disallowing forms within a sizeof that is ICE - no need to inspect such exprs

unnecessary additional complexity → we will have to poll this, good Q
costs on XM's 'discarded'

AB attrs are ignorable; not requiring [[repr]], good; but are req. semantics?
↳ user restriction, not compile

SHM like the idea, restricting all uses will restrict useful actions though
prefer to say if reproducible, the following well-defined properties hold

Sens don't fully understand... will investigate

AB any impls that disagree this? → maybe?

RCB same prior art Q., agree w/ SHM's idea, carrot vs stick, easier to support that following regs gives guarantees
↳ don't agree that the analysis could be difficult, though it's work

AB hard sell

Me PSE's, not much user ext

Opinion could 6+3, 0+3, 11+0
sth along lines of N3499 in C2Y? 9 / 3 / 11 (Abs)
work direction

N3498 known const size (extends n3416)
no intended semantic change, although normative; clarity, avoid neg. positive lang clearer (n.b. MV's Any)
some wording changes wrt. 'assignment expr' i.e. ctx of prev papers

ABch could have VLA w/ constant size? → yes but didn't invent the term directly in the document

XM: 'known' is def & slightly diff't

Me support, works w/ MV's Any b/c pos lang

PK so wording allows VLAs in structs? → why do more wording work

Ryan also support lang

YN 'asgn expr'? → will fix to use same length words change as before

Opinion ~~sth along lines N3498 in C2Y?~~ could no poll, no opposition

N3394 pass/fwd decls v4
this is sim. to GCC feature, not repeating the introduction
group had diff. prefs for specific syntax; Consider Apple ideas too
RCB is the additional change optional? → Yes
↳ don't like the optional change
Y/N summarize Apple objection: position on Reflector & in chat. Author of
Wkly implemented bounds safety extension, significant experience
this doesn't apply to non-parameter contexts like return & members
prefer a more general solution; describing size is usually other parameter
& commonly needs fwd decl - therefore inherently error prone
RCS can we see a full presentation of this?
RCB could voting on this be a problem?
Y/N just need the door to not be closed AB: no rush to approve
Reflector 78634 describes
allow fwd references b/c more natural
MV we did discuss & lend against fwd ref
AB new information is the weight of impl experience
RB don't want to wait - can do sth similar to make fwd progress
Y/N general safety approach presented in GroupLLVM - users have seen
will prepare slides after this
Jens if we don't want two ways - are the approaches incompatible?
could we do both?
(MV) ↳ pointless to do both, esp if not preserving sema
Ryan forces more typing for a decl on same line
A. Rich only to retrofit → Jens: this is for that purpose
MV not in a rush to get this in
AC these should be compatible, even to pass fwd decl in reverse
Y/N seems to be redundant
JM really don't need three approaches

N3433 'Chris's version of this'
aims to address concerns w/ prev proposal
wish for syntax to follow semantics, not vice versa

no desire to add decls for other kinds of things - not supported by GCC
a fwd decl is a decl, despite not being exactly the same thing
main prior art is GCC ; semicolons are confusing, don't know which
element is 'real' until the) ; hard to learn exp. by example
GCC doesn't support empty → see LJJ, also for etc.
sequence point in between VM types? GCC treats the decls twice &.
even re-executes side effects - decls are not the same as subsequent ones
examination of C precedent as well as prior art - for loops have a predictable
number & believe so should this ↳ as do if & sw
identifies a number of errors encouraged by syntax
A.Bch think semicolon seq more readable if multiple
YN if multiple decls, how would you prefer to write to make portable?
↳ can impl macro however
MV thought semis mitigate technical risk by immediately indicating a fwd decl
AB is a redeclaration valid if it doesn't have identical fwd decls? can it inject the type?
MV as long as you construct the same type
RCS seq. points stopped being created after C99 - now have 'sequenced before'
RCB like this, seems like enhancement of MV's, solves issue created in C23
can't alternatives - only to 6.9.2 or whole? → don't care about seqpoint
CB urge undecided people to look at syntax
RCS resolve this vs. MV's approach
MV semicolon question? what other choices... empty list?
SM things that are not parameters
? neither compatible w/ C++ MV: both can be hidden by a macro
AB have either brought to SG22 to ask if C++ want it?
MV unsure CB no
↳ would you? this is the most common thing to have in header files
SM this is only relevant to VLAs, which aren't C++
AB attributes may take advantage of this ; not blocked by C++ though
ES what is this good for? allows declaring the length before the first param
but why not just swap the order, use a macro if need be

CB interface may need to be stable - perhaps this has blown up

RCS that bare semi is annoying - info, needed to indicate position, not here

Opinion does WG14 prefer N3433 to N3394? 11+4 / 2+1 / 8+2 (Yes)
Chris's Martin's 15 / 3 / 11

N3395 unsafe buffer usage
builds on static & dynamic memory safety modes introduced by TR24731
find pragmas useful for regions, vs attrs for whole block
examples of how this might look; quite restrictive atm
early want feedback on forms etc

PK if a trap is UB, doing nothing is conforming?
↳ could have to define this obviously to terminate or sth else, important for security profile that it stops running

AB pragmas - extern inline in a header, can have two different pragma states is that UB, or...?
↳ want to avoid UB & could add a rule to resolve the inconsistency
eg. static behaviour is always checkable, dynamic mode may not apply

MW don't envision as per-TU but for regions

YN more comfortable opposite way - bounds checking model statically & then work out ways to enable; don't want this to interfere w/ memory safety extensions; more comfortable if experience drives design

MW this does exist in Sun etc, tries to put together

RB can be done both ways around; wrt TUs, could be similar to existing pragmas affecting 'file' scope

AC GCC has some of the prior art, but would want to see deployment due to complexity → MW have a branch w/ feature as described

SM better bounds checking means more constructs in safe modes - feed into each other usefully

GC considered compile switches instead of pragmas? At whole TU level, same as optimization level; need to pin down applicable safety rules
a safe fn prob. shouldn't call an unsafe one; simplify the rules

interesting to see how much UB needs to be eliminated to make C even-safe
Checked-C uses pragmas ; would like overview of various approaches in
use by impls ; would like a dedicated SG for this
MV obv. need to elim. UB in safe mode - more we delete from Std, less
to worry about. Not a problem for this to be region-based
mostly the UB stuff (bounds checks) inserted at the frontend & opt. later
enables only looking at certain critical parts of code
AB strong support for sth along the lines - most user-requested feature
want to poll for starting a SG, aiming to produce a TS & get impl
buy-in, get forward motion
CB really like this - wrapping around changing mode inside a TU - extra
data needed for arrays? can this be done? could interleave/incremental
fixes? this also enables support by simple impls
MV basic proposal is simple to implement. safe stuff isn't that hard
to make really usable, does get more difficult
full bounds checking could evolve on top of this ; doesn't fully enable
yet but could incrementally get there
RCS similar to Ax L? do we want both? should stay aligned?
MV / meant to enable this but not adequate, could subsume
↳ Doug Walls retired before finishing the work AB: remove Ax L
RCS nothing using it, ok
DT like this, memory safety widely requested - need to refine 'trap' concept
to avoid adding UB?
↳ would need to clarify? Jens: this is defined stm
looking at bounds checked langs - mostly use dynamic checks. Hard to
prove statically - must integrate w/ dynamic checks ; need a way to
reason that code can't trap. raw ptrs also don't have enough info for this
MV first version disallows ptr arith for this reason - lot of existing code would
benefit if we refine the approach, but also this enables some safe code
SC nervous bc dynamic stuff, want to see implementation before deciding
MV similar to current gen Soc
YN q- first para - does this exclude bounds safety extensions making extra
checks?

MV tries to be consistent to prevent a wrong size - could not work w/ existing code

CA like & support; mem safety is biggest customer concern; want to migrate; helpful to have more clarity on terms; echo that 'trap' needs a tight definition that allows exits like exploit-detecting mechanisms many scenarios can result in this

RCS SG discussion deferred.

Wednesday

N3426 improved nullptr

nullptr as in C23 is incomplete as long as 0 exists

amusing Youtube video with much 'bro'ing & observing that we didn't solve the core type safety problem

two proposals, starting second: require typed NULL
make 0 deprecated
that pref is more radical & willing to break code
could force a diagnostic
no changed semantics expected to be 'wrong' as such

some suggestion to use prose rather than code for def

CB strongly support opt 1. in practice compilers do not treat cuts as error anyway; opt 2 might achieve less than nothing & affect no users
warning has to be mandatory

SL love opt 1 but concern that risk repeat of C99, lack of adoption, soft pref

DT break a lot of code & case resistance (not opt 2)
don't understand how to init array of pointers without 0
↳ dynamic array can't use const

MV copying a dynamic zero is already not guaranteed & this doesn't change it

AB can't support either. cannot reasonably do opt 1; massive sweeping changes risk introducing bugs, could result in not updating
for 50 years we promised this works
can't realistically clang in -pedantic either, will break a ton of users

benefit of impdef NULL alias nullptr or _null in the definition
experience of non-diagnosable cvios is that even after decades, tons of code breaks - implies a language fork & benefit does not justify it
MV users do exist who want this - not ones sitting on old code, but new users even if it takes decades, that's not bad, slow is better than frozen
↳ still refer to actively maintained projects
PK agree w/ AB, will give FPs on existing code; support that the macro should not be an int
AC qs for impls - would you add an opt-in diag? AB: hard to answer so few uses do, it's often not worth the effort → -Wextra in GCC
↳ want lowest possible FP rate, not convinced circles red beep
turning NULL to nullptr could break code conforming to POSIX
↳ this is already allowed, not defined
SM prefer obsolescence over removal, would allow zero-braces
don't want to mandate a token sequence; need to fix many Std examples
CB all uses I know would be abolished, zero concern except maybe zero-brace
nobody likely to force use of C2Y
Me MISRA, exp w/ prohibition, exc for zero brace
RB we would never do this either; Std out of sync
AB should be afraid of uses forking language ~~C++ is a terrible lang~~
↳ MV: C++ does this
DS: for alt 1, are you say that NULL exp to void*? not seeing in text → removed from possible null ptr consts
MV prefer to poll alt 2

Opinion ~~would do along lines of alt 2 in N3426 in C2Y?~~ / /

Opinion would like define NULL macro as having type void*? 8+1 / 5+1 / 4+3
9 / 6 / 7 (Yes)
not clear direction

Opinion would like disallow NULL macro from having integ type 11+4 / 1+1 / 6+1
15 / 2 / 7
direction

DT want to better understand impact on existing code

N3398 contains offsetof

widely used, relies on magic, good candidate for Std; some known coding issues; Linux has different spelling & const version

RCS what level of nested object will this find? → feel like this needs examples

CB strong support, everyone uses this, vital for common code esp subtype polym. debatable UB - putting in Std means it is definitively not UB. strong exp

RB other codebases, any uses w/ other meaning?
↳ only ever seen with this meaning & order of arguments

Peter good idea; q. on treatment of const, still want examples
↳ existing practice is a separate macro ↳ Linux var. doesn't do what name

AB favour, q. on design; couldn't make offsetof a macro today, compilers would use a builtin → RCB: macro allows user to use the ident for old version

Me exactly what stdlib should have; got-generic now possible

RCS introduce any UB? → yes because it can be misused

DS haven't encountered in wild - can't you build w/ offsetof?
Peter: naive version can be built that way

MW putting in Std gives guarantee of higher quality impl

DT means code used are known correct

A.Rch like portable macro impl, reduce entangling of comp/lib

Ryan like feature, want along lines

SL reservation about macro here, could reject code a KW would handle better

CB effectively a downcast - code variance of QoI in the wild, hope to see unified impls. Thank you.

AC could this open up having UB into a CWG? using wrong type operand?
↳ dynamic, cannot eliminate UB → can improve on existing

AB if Udy version goes in we can guarantee better behaviour & reduce the UB
↳ more complicated but can poll interest

RCS if the UB is the same as offsetof, we can only record it once
↳ no, it's a new one

Sens like idea, favour as KW if possible; more CWG over UB is better
moving offsetof into language would also be an option

Opinion could sth along lines N3598 in C2Y? 16+4, 0+0, 4+1
20/ 0/5 (Yes)
clear dir.

Opinion could the preferred feature to be a ~~value~~ keyword in C2Y? / / ( )

PK sth most users won't touch - core feature adds impl burden to SDCC, no benefit want to provide low-quality version. Every added is added work, pulls out devs
AC want to clarify a KW could use _Ugly sp. → not adding it now
DS if making long test, should do same for offsetof. not convinced UB can be eliminated by anything can cast to void* ; someone will have to kill the UB later prefer to not focus on UB here
GC ought to be consistent w/ offsetof ; don't understand implication
RCB swayed by PK this should be a lib feature, avoid bias bias. GCC etc
CB response to DS - doesn't matter if KW or lib, there is an unavoidable cost here, depends on what is passed in ; can't elim. UB from class hashing
ECS want to see this fact discussed in paper
RCB anyone against macro → A. Rch: against if impl is non-portable
↳ AFAIK no portability issue

Opinion ~~could~~ sth along

N3422_Opinion
revises version from Strasbourg ; slides from then were submitted & wording changes added ; rebuttals to common criticisms also added
- the type system is distinct from the object pointed by a ptr value
- common criticism of qual. referenced type addressed - less usable esp. for arith
- opt vs mand is key to understanding feature ; motivation from obj-orientation want the simple form to be the most correct form
- what this adds : assume-null vs current assume-valid ; regular w/ existing no optimization enabled here ; doesn't break defensive code branches but enables expressiveness. statically guaranteeing mand isn't possible
- cost is well-understood by users (similar history of non-universal acceptance)

dcl un of shstr used as example ; immutability as analogy
- _O qual on NNL as a direction for future ; analogies to fixing the type of string literals - gradual upgrade path
- introduces qualified fnptr types ; interop w/ enhanced variance proposal which could be expressed for better using qualifiers
- working implementation is up on Compile Explorer

Seas way forward?

CS need guidance - don't want for the debate on whether this is a good idea

GC linked & support - could this be overused in large codebases?
↳ use effort without benefit? intend not to make code more verbose & might result in lot of manual effort
↳ are there risks of FNs? what if a ptr should be and isn't, what if it shouldn't be and is? ↳ won't be protected from a UB; same as status quo
↳ Clang as-is can report checks w/o preceding null checks
works exactly like const in this respect ; not protected against mut

PK why should this be a qual & not an ignorable attribute?
↳ don't want it to be ignorable; semantics of attrs here are ill-defined

AB originally similar design rejected by Clang - did not like qual on pointee string around LLVM conf - uses always say the same thing, expect elsewhere having to repeat the explanation is a UX/design problem - need use experience
Google have researched the use experience issue internally - Getmyd & attrs didn't solve the problems ; nullability specifier worked best but both _O & _M inadequate for incremental adoption - need 'non-nullyfied' do not think this is the right design yet ; maybe a TS would help

DT important for security to guarantee no null checks ; get in the way of decent bounds checking ; having a TS would be worthwhile here

could desc in N support this syntax seems ~~exactly~~ to be presented in a TS? 8 +4 12 / 6 +0 6 / 6 +1 7 (Yes)

Opinion

RCS NB major members support this?
— four indicative

CP voted no b/c don't think TS is needed!

implement this anyway

N3471 printf UB

'embarrassing UB in printf' if a spec gives more than 4095 chars
minimal change to reality expected. not all long doubles fit in this - should
not be UB to do this. Also long strings ; adds zeros or spaces but not
expected to be serious impl burden, correct rounding well supported algorithmically
problem also affects any fn. defined in terms of printf
should be statically testable ; proposed name maybe misleading, change
in next rev. to avoid misuse

UB can be removed entirely for 'reasonable usage' - unspecify what is printed
but don't need UB generally

RCS count here is root of a format vulnerability - could returning unsigned help?
↳ neg. vals used for errors ; could saturate at INT_MAX instead though

PK idea is good to make IB not UB ; don't follow reason for inc. limit - huge
data types SOCC cannot implement (min 45 bits, 128 realistic) - Uses an
internal buffer & even have a lower limit than min. for this
not familiar w/ algorithm to elide buffer → only need for correct rounding
work into an have until the buffer could be

RCB don't want increase, currently valid for most use cases, impl allowed to be larger
doesn't help anyone become conforming
like the macros etc - help test & solve prob, but limit itself not helpful

AC why not just report error, no UB, return -1 only

Ryan like getting rid of UB ; like it as a use ; like the query, based fear

A. Rch is RCB ok w/ req. no UB for any fmt? → Yes, reasonable

CB worried about macro → will change the name. should have something here
for uses to query, useful to -S- on
↳ b/c people will use it as an array bound, don't want to encourage that
prefer returning a fail indicator

RCS is this a byte count? → char count, number of chars. should also apply
to wide w/ same macro name

A. Rch assumed this would be sth like INT_MAX/10 if characters can be up to 10
↳ isn't that bytes? → not the way it's defined atm. counts code units

Opinion — could like a limit more dangling exp in N3471? 11+2 / 13 / 2+1 / 3 / 7+1 / 8 (Yes)

DP — is LISL_MAX a typo? → should be LOISL_MAX, yes

Opinion — could like to inc. limit val from 4095 dangling in N3471? 6+0 / 6 / 5+2 / 7 / 9+2 / 11 (Abs)

dis. against

~~Opinion~~ — will amend as directed inc. removing the UB & make IB

RB — failure rather than trunc

N3466 nulls in library

response to impl q.s abt strncat, fread, fwrite ; clarifies wording wrt which args can be null

AC — does this silence some static analyses?

AB — not worse than before. cond. on passing zero

CB — think this was a mistake to allow const null here ; other langs don't allow ; can't be expressed as static constraint

Decision — cons adopt N3466 :- CZY? 12+2 / 14 / 0+1 / 1 / 4+2 / 6 (Yes)

N3410 Demars X1

intend & extend linkage in same TU - not aware of a possible use for this & widely rejected. second example pos. just confusing

Decision — cons adopt N3410 :- CZY? 16+4 / 20 / 0+0 / 0 / 1+0 / 1 (Yes)

AB — not opposed, not convinced of utility

CA: why could this happen? is there any use case?
↳ no known uses, easy undo

~~N3471 clemars X11, update 3411~~

going to N3411 demons XII first
simply deletes the UB.

Peter: is this real UB? → Yes because it's outside constraints
all the other 'shall' here are still UB

AB: currently - if file is not newline, it's UB? - yes. New version also adds a CVio b/c POSIX requires the newline

↓

N3477 adds the constraint JM: prefer new one

RCB: shows up in machine-generated code; have to disagree case

Me: artifact of 70's?

RCB: no, still applies to 'record based files' used today

Ryan: like deleting the UB

AC: spirit of Std prob. wanted this to be a CVio; complaints are from POSIX which already requires this

CB: say why oppose constraint

AM: our code does this, only diag in pedantic. don't want CVios for non-errors impl QoI to report against the POSIX req. sep.

Decision: want adopt N3411 in C2Y? 17+3 / 20 | 1+1 / 2 | 3+0 / 3 (Yes)

N3478 demons XIII → N3412 originally
partial comments at end of file - original was a non-issue, constrained partial prepo tokens, which don't exist & don't need CVio

RCB: no obj to new paper

AC: is a comment on the last line terminated? //

Jens: think this is a CVio b/c of how syntax rule is defined

AB: we should change the syntax, does not intended to fail, too pedantic

AC: a tococode wouldn't do that? → happens in human code too
→ to include EOT in 6.4.10 p2

MW: will write this as follow up

Decision: want adopt N3478 in C2Y? 19+4 / 23 | +0 / 0 | 1+0 / 1 (Yes)

N3418 demons XIV
caused divergence btw GCC & Clang ; resolve by pending to avio
SM do not consider a valid example - has backslash as 2 freestanding PP-tokens
no obj to change but link is misleading

Decision want adopt N3418 into CCY? 16 +4/ 20 | 0 +1/ 1 | 2 +1 / 3 (Yes)

RB so no diagnostic from output of standalone preprocessor?
Sens generally compiler not able to recognize things from earlier phases
AB does this diverge preprocessing? → don't believe so, UB/ill-formed subsumes avio
RCS saw long list of divergences → want to reduce list
AB looking for citation - this seems to be an existing diff btw. languages, UB beyond
↳ does allow line-splicing to form a UCN

N3480 demons XV updates N3419
has an identified wording issue not fully covering all IB cases
RCS this is an extension point though?
MV right but the dlls declarable types should be defined by the impl, not UB
RCS think UB is too broad - explicit extension points to gain own category? distinct because not existing UB
MV not sure this is usefully distinct → 'political value'
PK hard to do things like contained as an extension
AB 'or no definition' gives parse - this extension point is used on Windows unclear if current wording intends to allow this
DS POSIX allows the third env var argument ; IB would require documentation of the allowed extensions & forms
MV it already does this
RB other extensions allow entry points that are not called main widespread
AB if you have hosted+main, vs nothing named main, different UBs, little ouch
MV does define main as 'the' function called
AB ignores existing practice?

MV something else than a C program anyway?
AB 'or an impldef ident' solves prob
Me distinction between lib & prog meaningless, ignores reality even for hosted
Sens would like to clarify intent of this section
AB should generally allow impldef entry points + RCP on main
MV happy to so long as it's not UB ↳ strike out req for main
RCB so needs new paper? → yes, but try this text allowing no main at all
MV ok needs rework into new paper

NO POLL

N3481 denotes XVI updates N3420
only minimal change from '20. different approach from before, makes it obvious apparent unused extension point. updated paper makes the constraint to be generally applicable
Sens why not add to value conversion section? → no constraints there yet is all
RCS switch to positive wording? RCB widely ranging double negative, could fix all
RCB so this also adds a constraint section where one wasn't before
Sens rather have relevant things together, in 6.3.1.1 ; no normative difference
MV can we add it here? Stretching wickets

Decision count adopt N3481 as-is into CZY? 11+3 / 14 ; 0+0 / 0 ; 6+1 / 7 (Yes)

RCS request the double negative is removed editorially

N3482 denotes XVII updates N3421
resolves a UB that only applies to a CVio case of incompatible types, should be ok to just delete ; might have already adopted the optional change via H2O's paper
Sens don't even understand what deleted text could mean
Me seems added by N3363 → not in the form of a constraint
AB confused by diff btw 1a & 1b changes
MV alt is where to add the constraint - currently none in library

SM not a constraint MV so it's now UB, odd to add a change
Jens editor will have to change this
SM we clarify what the UB is, even w/o shall, addles a requirements
PK this was a normative change but doesn't affect what happens in new vs
Decision — want adopt core change in N3496 in CLY? 17+1 / 0+0 / 3+3 18 / 0 / 6 (Yes)
82 w/o add drop

SM believe this was about pre-Std mechanisms
RB equivalent to original pape only in this case

N3496 clarify width macros (updates N3413) (late)
resolves misunderstanding by vendors, clarifies where to look up, not all IB
RCS 'shall be replaced' is req on impl? → Yes (unchanged text)
unnecessary 'shall', no UB here → just 'are' then
following 'shall' should also be 'are', not programme actions
RCS footnote is new, 'many' → 'each' between versions
AB requested b/c got BOOL_WIDTH wrong in Clang on first reading
Decision — want adopt N3496 in CLY? 18+2 / 0+0 / 2+2 20 / 0 / 4 (Yes)

DS width vs prec - prec bits aren't part of the numeric value, checking this takes into account
Jens only diff for signed - that's not precision, that's another value

RCS editorially replace the 'shall be' to 'are' in the ~~Jens~~ affected locations in 5.5.5.1

~~N3447 defer TS~~ N3329 transparent aliases
discussion, not voting wording in; consensus last in Strasbourg
addresses issues raised then eg dlsym. ABI-stable upgrade paths.
entirely follows existing practice (asm labels etc), no runtime property
ABsch what does __func__ resolve to? → not replaced inside the definition
uses the defining name
MV still don't understand how this helps - if nothing at the linker level & does

what a macro does, which is insufficient to change ABI; w/o exception for redecl, by use w/o header, don't see it help

SHM if inside the Stdlib, you count the one declared as extern in definition - redecl is of originally wanted fn, but lib is allowed to redirect
so redeclaring strftime in macos would call the wrong function - will link to old ABI, not updated version; in existing practice the declared name doesn't correlate w/ extern link name

AB so... try to understand - given an existing binary, some name continues to link to old defs; new code links to new defs

MW this can't be solved w/ macros because of redecl. reason; if require the header, we don't need this feature

SHM want to solve the ABI issue w/o trampling identifiers - want to allow NS & shadowing
also doesn't solve struct layout being incorrect for old definitions
does solve prob of being able to leave code alone, by copying ex. proc

RCB reiterate. like this & solves the real problem - stdlib is not the whole problem & this makes good progress by adding portable labels

MW only captures the least interesting part of what asm does, not the useful part

AB major linkers can do this; is it implementable in other environments

RCB because these are regular identifiers they don't have that problem

SHM libs use manual mangling w/ C etc - didn't copy 'strongly typed' version b/c limits to what's usable & if C is our target, this will too - no dependence on any linker technology beyond extern itself - no need to parse .def files etc, can be scoped which allows localized usage (like -Op?), no link input at all

SM think this solves tagoth issues as well - cannot be defined to call by name b/c can be shadowed & call wrong thing (every impl gets this wrong)

CB considered typedef syntax? implying assignable

SHM did consider, don't mind much; tried to look like C# using
POSIX proposes allowing 'using' to do exactly this - freedom to look like whatever but aiming for compatibility - many candidate names, don't care that much

CB don't like resemblance to references

Sean extent of feature - what happens to attributes? keep [[dep]]?
↳ yes, answered on screen

does this also work for variables?
SHM not accidental - existing practice supports variables too b/c linkable names
users requested not to restrict to fns artificially
alias don't inherit - alias isn't deprecated b/c underlying name uses
diagnose through alias

Opinion ~~would sth analogous N3529 in C++?~~ / / ~~(Yes)~~

Opinion could pref. use from tclef like syntax for alias? 7 +1 / 5 +1 / 11 +2 (No)
no direction

N3442 defer, core feature
confusion over jumps from deferred blocks - not allowed in Strasbg ver,
added b/c requests from out of group; met w/ disdain but wanted
opinion from LWG. same prior art in GCC & MSVC
block-scoped labels can be captured by GNU nested fns
MS allows goto out of __finally (main issue is w/ catch)
also return out of __finally w/ value override
this is well-defined, just strange 'definitely arrives'
Sens dislike these constructs; current version wording should say 'unlabeled
statement' not 'secondary block' for clarity of semantics
SHM is that different from what 'if' etc do?
↳ not a statement you can jump into, fundamentally different
DB what's __label__ for?
SHM demonstrating a vendor extension. plain label is not captured by nested fn
not part of proposed feature
AB any evidence that the practice is intended? what's the use case?
SHM user wanted to implement a retry from error check in defer
'why can't you do this normally', not actually compelling but user
stood for it. behavior is deterministic at least ↳ 'i hate it a lot'
SH generally support but issue in example, can appear as whole body of 'if'?
useless & not needed - prefer to tie to compound & 'try acquire'
with conditionally registered blocks

could prefer a ban here b/c useless

SHM think that would be worse - could require state & allocation [like Go]
could ban but not hoist. even if looks silly.

ND why don't we do like scoped_exit which allows canceling a run, assignable
could like humbles to support feature

SHM prev. disc - if it's an object needs an object model, which C doesn't have
no lifetime model to control destruction. other issues w/ class also apply
none of the wording to support if exists in C
nested fns etc are in the works but separately

N3497 'simpler defer' updates N3474
aiming to see what we can do w/ existing extensions
simplified syntax, & adds a header which could be shallow
there is impl. experience - simple to do w/ nested fns + COUNTER, or
blocks + block helper. Even easier in C++
differs by not adding as a usual stmt, only block item (makes the semi etc)
previously didn't define weird jumps so much - was restricting to exact
reverse order; UB in place of jumpouts
easy to analyze; easier to relax restrictions

AB what is 'jumping out' - a return early? break?
↳ every jump out is allowed, launches reached defers & skips later
jump-out is same wording? (no it isn't) same idea as jumping over
a VLA

AC allow the horrible jumps from before? → no, restricted same way

Me not sure about skirting limitations of existing practice instead of
agree?

Jens few drawbacks, impls can allow no-braces as an extension, not hard restriction
do actually want the connection to a compound stmt

PK want to be as simple & restrictive as possible for new features - don't
want Lexe feature (→ MSVC has it) → very optional
do want the UB for longjmp

Ryan worry about the lack of cleanup-interface of longjmp
↳ want to avoid UB

want to either know or not know

SL dt libs exist, more magic means harder to develop portably
harder to maintain & develop independent stdlib - therefore don't want to force in lib interface

SM want design experience from a TS; need to vote away differences & come to combined proposal

Sen want direction on: assoc to compound, and on library approach

AB ship vehicle affects my vote, IS or TS

RCS strongly decided to TS, we have the work item

Opinion could like ~~as syntactic statement~~ defer to be defined as a block-item as in N3497? 12+5
~~7~~/ 4 / 17 (Abs)

RCS authors should determine

MW want an impl → we have one, it needs this
likes library option

Opinion could like else jumps out of deferred blocks? 0+1 15+2 8+1
1 / 17 / 9 (NO)
~~as~~ by jump statements

SHM got feedback from users that this should work like if etc
not arbitrary design

Opinion could like to allow defer to be provided as a library feature
by requiring a standard header name? 2+2 8+2 10+1 Yes
4 / 10 / 11 (~~Abs~~)
consensus No

SHM got adequate direction for TS here
how fast can we target IS after TS? → no limit, can even be faster

AB TS w/o experience is not useful, need user experience

RB why opposition to lib feature?

RCS constrains design space Pete: control flow should be core

O constexpr asserts?!

AC could still make a lot of this UB, support QoI
SM don't see benefit when needs to be backed by compiler anyway

Thursday

N3494 clarify wanchex attrs (updates 3456)
CCS new attrs missed features used in practice by pre/cont b/c assume return (you not documenting this) → non returning can affect other side effects in caller chx so reasonable requirement to impose
opts don't generally consider signals, abort, etc - cannot foresee this, & asserts open for discussion which operations to include in listed set, same obs. exceptions abort & Exit 'morally' stateless, unlike other exits eg. interact on atexit
- assert is special → Jens doesn't think we should restrict this, should allow
RCS what's a 'visible effect'? → do have a definition in Std already
Me support allowing assert
Peter is this saying abort/Exit can be optimized away?
↳ no, means that extend reordering could make the entry
PK if it's reordered behind a different endless loop or abort
↳ useful but words may be too open
MV could moving these be risky for security context? must trace when abort fires that is longjmp used for? → always jumps up
RCS so - VSE has to invoke an object & 'visibility to' sth - text talks abt exec state
Jens we map all side effects as-if obj does
SM not helpful to use sim. but subtly diff't wording - if you mean 'observable state of exec', use that term when writing
AC interested in use in security sensitive code to enable reasoning - must not be dangerous - longjmp is problematic
Jens trying to match to GCC which is poorly documented - welcome more info also how Clang handles these. impression is that they ignore it
Ryan improves clarity; what is long term plan for consolidation?
↳ no plan or direction, only a Cloister
MV don't think compilers care about any of this

why can't this inc longjmp, must end whole exp
AB Clang leaves all reasoning to use, UB 'don't do it'
↳ even asserts? ↳ prefer to not mix these w/ noreturn & specify in that to not combine with these
Sens don't always have attr, know on behaviour
Me way of overspecifying longjmp
AC all of these are declared noreturn - should be easy to diag?
Sens function may not be able to check, don't know if GCC checks
AB use tell LLVM to believe the annotation

Opinion could like to add? list of exception behaviors along lines of N3494 ~~that~~
4 +0, 3 +3, 9 +3
4 / 6 / 12 (Yes)

AC can have cvio instead of US
↳ not statically checkable → direct call could be CVio
AB US & UB are different here
Sens difficult to react to this feedback. welcome advice
RCS move last bullet, ship leading stems - Thx advice

N3427 unspec arrays
dynamic size information now passes through auto & typeof
star notation allows abstracting the count while specifying the elem type
similar to auto deducing only part of a type. forms a composite
PK don't think it's worth while; add sth to impl & learn for undersized feature; only saves typing length of
AB 'optionally enclosed in braces' - isn't that an array?
↳ scalar ptr to array can be en-braced
what about pz = p? → propagates as-if w/ auto everywhere
AC wrt. length of, repetition risks errors
MV redias for multiple dimensions
also opens up bounds for structs etc
AB do we have any experience? conforming extension to do now

could like experience

Me likely limited ... like declaration for communicating sameness of types

AB agree w/ PK that this is limited - auto already serves this

SHM so this allows the star to declare constants too? → Yes

RCB sympathize w/ PK - not everyone uses this, limited cost benefit of this data is good but we may struggle to gather

MW impl effort of this is very low imo

A.Bch not a strong arg - diff uses have different needs, eg. decimal, multidim

RCS weird to understand but sort of code people copy from books

Jens why the star, not empty?

MW star gives existing composite type rules to this, works w/ multidim

SHM solid favour - tried writing fns. w/ ptr to VLA for string lib; was unwieldy & awful, had to rewrite the size & needed macros to hide repetition; this makes a lot of that stuff easier to write. not for ret. type in project, gave up & used auto. helps writing VLAs usable when writing libraries

AB [godbolt] why not just make like I work?
↳ these are incomplete, not unspec

YN what bounds checking does this allow?

MW checking already works today, same w/ new syntax
↳ not about bounds checking

DT what impl do you have? → small GCC patch

Me implementability is not the problem w/ it

DT agree that making people retype things is a problem

MW not obscurity - still large comm. of uses

SM still some wording issues - revisit wording after finishing extended auto

Opinion would sth along lines of N3427 in C2Y? 8 +3, 3 +0, 6 +3 / 11 / 3 / 9 (Yes)

RCS could people impl this?

MW could try?

N3357 frexp double double
'give dbl something something'
model can be read as applying to C, IEEE or other; dbl/dbl is C model
AB not opposed but is this clear enough? word usage?
RCB this uses lang. from Std abt FP model ↳ editorial footnote is fine
not opposed to changing wording
Sens implies a term is introduced?
RCB only discuss C model, never go beyond that; impls follow model 99% of the time
AB so matches existing practice
Sens prefer more verbose formulation
DT seems editorial
DS feels like prescribing an extension, out of scope
RCB opposite of that, spec doesn't apply to exts
DS every FP number is part of the model in C?
RCB the model allows alle FP numbers outside of it - correct that this could be applied anywhere
Sens maybe better in the preamble instead then? → tried to be surgical
DS my extension could comply, then this statement would be wrong or incomplete
up to maintainer to decide if this is true
RCB just says if you follow it
RCS saying 'C model' would be more refined; verbose term gets old fast
Sens nothing in wdth gives this ctx - would be good for general introduction
RCB personally think this is a bad change, opens can of worms
why not put this text everywhere

Decision: Want adopt N3357 in C2Y as-is? 1+0 / 13+6 / 3+0 (No)

N3461 range errors
in C23 these stop applying to integer types - breaks some fns
CFP intent/understanding was not to have such errors, so obsolesce the ones that remain listed
keeps impls in sync & valid
RCS 'what is C18?' → C17 Sens: inc. fn names in future

Decision vote adopt N3461 in CZY? 14+6, 0+0, 2+0 20 / 0 / 2 (Yes)

Pete is this TC material? → it would be
AB put on the ideas list AC: can we add errno?
RCB: that's allowed by this
Opinion could ~~pages of interest list?~~ (Yes)

[no obj. to unanimous consent]

N3460 complex ops
moves content from A&G to body as asked for last meeting; seems for mixed domain operators, cover non-754 as well

Decision vote adopt N3460 into CZY? 15+6, 0+0, 2+0 21 / 0 / 2 (Yes)

Sean makes things clearer & cleaner, like it!

N3401 sigfpe
RCB note this is individual, not CFP group submission
FT handling & rehosting is UB; some impls will raise on non Std math fns not aware of a Std fn that raises it
RCB scope of options - 2 is tightest? → Yes. opt 1 is preferred
DT has dick you poll impls? embedded practice can be to just fault
↳ asked on Reflector & got no reply
↳ concern that not sufficient
Aksh not huge problem
GC trying to remove UB from a handler; makes SIGFPE explicit & opt-in through API but this undermines fenv. opt-in
fractures error handling as signal can no longer be relied upon
not ideal but used IRL for error handling. seen used & documented in Linux Prog Ifc books - people therefore are relying on it
DT fenv mode is a global processor mode
GC also get this for int div 0 in some platforms

DS glenn said most - good idea but restricts actual use cases & fund problematic

AB did CFP see? position? → RCB: yes but consider out of scope, no opin.

JM if app calls ext fn like feenex it is using UB anyway
dislike implication that Std facilities could raise sigfpe b/c 'except as doc'

CB orig. thought 'ex as doc' was good way fund but convinced otherwise - this describes idealized system

Decision — count — adopt option 1 from N3401 into CD? 5+3 / 8+0 / 8+2 = 8 / 8 / 10 (Abs)
not consensus

Decision — count — adopt opt 3 ↑ ? 3+1 / 5+3 / 12+1 = 4 / 8 / 13 (Abs)
not consensus

Decision — count — opt 4 ? 6+5 / 4+0 / 10+0 = 11 / 4 / 10 (Abs)
consensus

RB why vote against RP? → no answer

Decision — count — opt 2 ? 5+0 / 5+1 / 9+4 = 5 / 6 / 13 (Abs)
no consensus

N3492 improved rounding errors [updates N3404]
improves specificity ; addresses all Ref concerns

Decision — count — adopt N3492 into CD? 10+5 / 2+0 / 7+1 = 15 / 2 / 8 (Yes)

RCS why 'one of' & not 'either'? → based on original text, purely editorial no technical change

DB weakening domain error into 'possible'?

RB makes more code & impls conforming due to divergence, impls took adv of C99 latitude ; cannot get domain error if model has no infinity

DB should still have it, errno as otherwise

RCS we're not going to make existing conforming impls nonconforming - natural process is towards softening

PK wording w/o 'may' is correct when i.f. isn't possible (pdl here)

note: change 'if one of x and y' to 'either x or y'

~~try to~~

N3405 error cands in math.h

similarly inconsistent on errors - resolve contradiction in current text, no intended impl change

Jens 'can' should be 'may'? → no the 'can' is correct here for sentence

Decision want adopt N3405 in CZY? 10+5 / 1+0 / 6+0 15 / 1 / 6 (Yes)

FT don't want to weaken the requirement, want opposite

N3452 complex lit wording

alleviate non-integral types w/ Std operations; not normative, clarification explanation to users only

Decision want adopt N3452 into CZY? 14+5 / 0+0 / 5+0 19 / 0 / 5 (Yes)

RCS 'since' should be avoided, prefer 'because' → possible Britishism

note: change the word

PK do these go on retain list? → error papers? RB: good idea

Decision want N3492, N3405 on retain list? / no obj to unanimous consent

Peter were these fixes for regressions?

↳ yes, intent.

AC reissue? → no, informal

[N3428 removed from discussion list]

N3437 VLA alloc control

two ~~dead~~ issues c/ VLAs: moral & technical. this is what got VLAs pulled from C11 - try to make this less underspecified. Always left implicit since 1990s; limited to numeric community, limited support features. big problem for portability; this doesn't overturn optional part

these cannot return null b/c obj must exist

can be pointers not for-designators only [whichever way around id is]

CB ambivalent! mostly use VLAs in test code - want to support strongly like instrumenting memory allocation; method of impl seems strange for sth provided normally by implementation; almost like overloading not intuitively user-defined

SHM wide variety of behaviors users want for this; example simd asks alloca call up to you if this feels like operator

CB not against fn call but would expect to call a registration fn, not use declaration

PK like to know user demand for this; can't use just allocate & deallocate (in defer) memory explicitly; think example 14 is outright wrong because of behavior of alloca & will leak

AB don't know how to deploy this - compiler works w/ foreign std lib with varying levels of completeness; the break from C23 to C2Y would break in practice

SHM there is no stdlib function - if not in scope it uses the C23 'magic' allocation as it always did, will not change behavior w/o opt-in. No exp. libsigsegv author did voice support & give impl advice

AB so user defines a reserved identifier we told them not to?

↳ unless we tell them to. can change name to sth else, must be a reserved id

AB registration functions would eliminate need for that

SHM wanted to avoid the overhead of making it dynamic - this binds to scope in the same way the VLA is bound

AC user providing & not handling errors would be UB? → Yep

using VLAs for passwords to stop from appearing on heap; security risk
can be provided at link time? → has to be visible in scope
why can't uses just use malloc? → needs defer for that...

SHM impl calls if it needs memory & frees 'somewhere' after lifetime - leaves existing optimizations open to not allocate or to hoist allocations & reuse same as current stack manipulation under as-if rule

SL would prefer separate operators like :new

Sean efficiency concern? can't do register inc/dec w/this; dislike pointer variable that can change alloc strategy - not thread-safe. prefer aliases & lite scopes, but not mutable ptrs or registration dynamically

DT echo security concern, allows use after free, interferes w/ register & sim. mitigation seems overly specific & would prefer to be policy-based

ND VLAs are fundamentally broken; don't know how they allocate, which you want in C; would like VLAs VT w/ injection across threads fully deterministic; been on direction but don't like VLAs

RCB some misunderstanding - impls do ~~use~~ need to change if they see this declaration. Adds use hooks into impl, don't consider good thing - user shouldn't control & undo impl strategies. Any header can define these & change use code that way

Opinion would 8th along lines N3457 for C2Y? 5+0, 7+5, 10+1 5 / 12 / 11 (Yes)
no dir to continue

[note: a study group is not a work item nor implies one]

N3423 TrapC
slides presentation: intro etc to other; referring back to White Hose paper & national security org. for safer languages; presenting TrapC as extended subset of C w/ some characteristics of C++; not in-lang subset & no exposure of an unsafe model except by linking to a CTU; adds others 'trap' is a kind of catch; 'alias' is a typesafe macro ('type overloading'?) 'castpoke' only templatize pointers; printf & scanf exist but are checked has a build system modeled on cargo, can be used for C as well

RCB 'decimal type'? → integer-based w/ decimal point
↳ scaled fixpoint? Yes

DS if ptrs are safe why no ubsan? → might not need restriction
but didn't want to prove at this time, 'don't use ubsan in new code'
on 9- 'cudze' files can be processed by 'cudze'? → Yes
- example of gets-based buffer overflow ; buffers are more like std::vector
terminate an error describes the issue ; does not proceed
DS what if I write this with a while loop & direct subscript?
↳ compiler retains the size of the memory region associated w/ pointer
'At in the compiler' → runtime check but not on every access
TS did the access happen? → usually yes up until the invalid one, but may optimize ahead to the UB. shouldn't get half a buffer
↳ for embedded HW, need to know if partial write occurred
which is the worse scenario depends on application
none of this example is compile time
Had it considered embedded example
GS newest C Std does resolve this w/ volatile - writes before the OOB may not fail (RB is that true?) ; do you see this as mostly runtime or compile time tool?
RR not safety certified yet but in SC, expect things to be thoroughly tested
'trap' keyword is original, not in C ; catches errors & handles
using the following block as a handle to invoke on signal in calls
contrast to defer ; mrp have an associated dtor which always runs
overloading printf ; 'also has member functions'
mostly aiming for minimalism similar to C

N3438 #embed sync
only aligning w/ C++ progress for C++26, requested small changes
resolves flaws in spec inc limit ; consistency w/ GCC behavior for parameters
can't really vote in as-is

Opinion could like apply changes along lines of in N3438? 8+3 / 7+0 / 8+3
11 / 2 / 11 (Yes)

AB use case for offset? → only loads a subset of file, reuse of limit
immediate user request

AB found these frustrating to implement, relatively limited utility; originally also embedding /dev/urandom - what do offset or limit mean? what is the order of operations? existing ones are troublesome

SHM understand all impls to have the same behavior - skip N elems, then limit L seems to be no divergence btw impls or intent

AB not agreement if limit left off infinite stream (crash vs 0); offsetting into device? offset can be done by build system, outside compiler

SHM did try to argue that this means discarding the specified no. elements - urandom is special, but this is consistent. limit is intended for devices & inf all this is in orig

AB Clang was heavily rewritten & introduced divergences

SM device files are really out of scope; parts of this should be relearn & offset is new proposal

Peter offset then limit seems very clear

Me media files, skip header w/ offset, alt file

CB see the use of offset, very handy for crash dumps
(poll here) (adequate direction)

N3469 array size survey
actual statistical analysis of responses! over 1K responses
nobody from Antarctica
result of long debate last session; graphs are in the blog post
surprising preference for non-Ugly KLV
large number of free text comments esp wrt 'length'; data from Google etc
strange expectations around null terminators, leading to bugs (in Google's code)
potentially informed preference; also showing weighted analysis for format
results surprising, expected to mirror Chris!
includes a vote to change KLV & add header if we want.

CB shouldn't matter, somehow does anyway... will result in divergence, scared

RCS 'if only someone had warned us in KN' - not surp, exactly as predicted
esp from industrial respondents; prefer count, prefer usual Ugly/<std>

SM don't like the kind of header; if we do, needs version; AxB, info, etc

AC on 'pref spelling' - length & count only similar in mild sentiment
difference is all in the strong sentiment, consider this weighted view
would impl countof even if lengthof voted in
CB cannot endorse that tactic but don't blame you
RCS like that this surveyed C users
Jens don't think this justifies the time spent, don't want to take Coute hostage
'hate' is not an appropriate terminology. cool down
JHM not intending to relitigate & don't care. Only wanted to settle the real data
despite survey, room doesn't seem to want LC KU..!
RCS do want a poll.

Decision ~~Opinion~~ ~~count~~ ~~hate~~ to rename lengthof to countof ~~strlen~~ as spec in N3469?
-C
[considered consensus] 6+3 / 3+0 / 10+2 9 / 3 / 12 (Yes)

Jens need to change my doc? → voted for a global change

Decision count add strlcountof header as spec in N3469?
6+4 / 3+0 / 10+1 10 / 3 / 11 (Yes)
[consensus.]
note redphotelize KU list

N3443 const ints
aims to counterbalance PT's paper mandating arrays are VLAs unless ICE
& later retitled by AB; aiming to take a new path with defining this
implicitly gives constexpr; some uses wanted extern to work too
this is not as aggressive as what GCC & Clang do; does fall within C++
rules for objects, also not going as far; covers the cases users care about
JM like the principle though there are wording issues
CB does this obsolete constexpr? → no, that's for any object; this is int only
& not everything; covers existing practice; cx has much broader utility
AB favour, matches practice

Opinion

would sth along lines of N3463 for C2Y? 18+5 / 1+0 / 3+2
21 / 1 / 5 (Yes)

N3505 #if exprs - revisit how
fixes wording to clarify builtins are OK
RB resolves my concern

Decision

Lout adopt N3505 in C2Y? 19+5 / 0+0 / 2+1
24 / 0 / 3 (Yes)

consensus

N3507 #dialect
this is a pitch not a concrete feature : slide presentation, no fixed syntax
noting cases where C23 + GCC-latest breaks compilation of old code
some proj like curl explicitly not willing to move away from C89
therefore, a make to communicate intended version
candidates suggested are directives
mental model is communicating metadata, not an instruction w/ effects
much precedent in other languages ; potentially improve build systems
improve diagnostics substantially ; could improve IDEs
also interacting w/ features like _Optional, 0-null
could work to fully change language e.g. btw C & C++
control extensions in use, e.g. asm
avoid conflict when an extension is adopted but not compatible
like 'trap' or diff't defined evil code?

Opinion

would sth along the lines of the ~~sheet~~ #dia directive prop by N3507 in C2Y?
6+3 / 10+2 / 6+1
9 / 12 / 7

ES proposed this a while back - nobody wanted it then ; want to just have
the one version tag in one place, no splitting of TUs
don't want TUs to be undecidable ; want '89 code to keep working
w/o mitigations to please recs unwanted semantics
hidden changes like 'auto' changing meaning ; not consider as a bug
feature, but as a bug, other direction

SC one of probs is focus on switching dialect mid-TU - difficult to implement sth like that: how can C89 .c use a .h w/ C++?

SL don't want to allow switching semantics, only to communicate

SC looks like choosing a version for a file; spelling c/in a version, or semantics?

CR important but agree that users will expect too much; get rid of the new directive & put under pragma, which users (think they) understand
good place to put under SG22; let people scatter pragmas throughout

RCS can see complexity but do like it; same simple things that can be done metadata about a file, in a file → users often don't know what dialect or version they are using; files can be copied losing info from build sys, if it's in the file can't lose that; things like use of no-strict-aliasing are also useful dialect warnings

ABch existing pragmas have support; would complicate the grace period for changes
stds try to reduce variation

RB would have been useful in the past, have this in own compiler as pragma (not in Clang) - after more, users encountered errors & were surprised

HK forever or as a migration tool?
↑ this one, users will not change

RB seems to contradict self w/ switching, experience is of only one mode
doesn't contradict - std mode, per whole TU
ignored in subsequent headers, uses first then warns on subsequent

Me comment vs effective directive
(poll here) direction against

Peter so people want sth but not this

N3389 TCE/constexpr

N3543 examples of UB
updates doc last seen in MN w/ examples for every UB, as a command that no example was possible to create
some come from places like CERT
currently requesting more reviewers to reduce chance of errors
(confusing example 6 ??)

RCB can examples be direct links? more useful
↳ plan to inline all examples, big C+P job
probably targeting LWP rather than TR or TS
↳ would pseudocode be a substitute when no real example exists?

RCS can we delete such UBs?

DS a lot of the demons are such examples, some may be worth keeping
also want the LWP to target C23

AB ex 6 is probably about numbers...?

Ryan would like one w/ Godbolt links? → test scale?
also like idea of pseudocode, seems useful

DS ex 195 is interesting b/c only affects hypothetical / future target

RCS would this go in Annex J?

DS asked before - in MN concluded should be LWP

AB expanding AxJ makes IS more expensive
↳ could extract AxJ to LWP

N3450 typeof in sigs
'this is weird' - typeof or typedef simplifies

SM prefer typedef names, in general

Opinion — in fn signatures for fn types

Opinion would like typedef names diagnoses of N3450? 5+3 / 2+2 / 10+0
8 / 4 / 10 (Yes)

CB this feels like bureaucracy, prefer typedefs

Decision count adopt N3450 in C2Y? 2+2 / 5+1 / 11+3
4 / 6 / 15 (Abs)

DK worried about the names, treading on user names
↳ suggest using POSIX names

Friday

N3408 branding, community, website
slide presentation not proposed. First, logo: traditional K&R logo pref. over modified isocpp logo ; colloquial names list for Std version
'C Cute' vs 'WG14' for group ; official group name: on code forge sites many taken by name squatting on GH ; codeberg requires proj to be FOSS
discussion of NP. alt group structures to simplify trademarks
same issue applies to domains ; examples of new home page demo-ed w/ various accessibility & themes

Jens -C ; opposed to GitHub as host
MW agree about GitHub ; naive of FP ops changes too fast → codeberg is NP & should inquire of it
DS website should link to C Stds ; open source has good documentation, what does this website do better? → it's more accessible to non-UG people
lot of info not useful to users, missing useful info?
CP how could ISO react to shared drafts easily accessible?
JM following SG1 Stnd Doc 23 → failing to plan is planning to fail
AB other people will look into fixing this.
JM can't name the lang. elec. page 'foundation'
RCS think this name should be made official. (who owns current one?
→ Ketel does, privately owned.
AB avoid breaking existing website, compliant w/ ISO as-is
BL will grab c-language.org

N3407 closing ghosts
found many non-requirement 'shall' - do not point to UBs, no way to trigger anything. proposal significantly reorders 6.6 w/o intending any normative change. Aims to substantially improve readability ; editorial improved by removing descriptive 'shall'
RCS is maint by editor explicitly & ignored by group
Jens Acc is not itself normative ; can advise editor to change text
→ transfer responsibility to the UBSG?
MW what does that mean? SG advises editor already
RCS put into scope of SG responsibility → already doing that, official or not

→ q for example → is correct

DS concern at replacing so much text → only reordering. DS can't to be sure this works
SM sent comments; includes unintended normative changes; already amended 6.6
& can't reasonably rewrite this way: numbers are already off by one
AB how unreadable would diffwords be? → horrible; agree unreviewable
Jens clear that should wait for shortlisted draft; will reword
SM principle is reasonable; get 'discarded' in first ↔ still in flux then
Opinion would sth change lives of N3447 in CCV? 11+5 / 0+0 / 4+1
16 / 0 / 5 (Yes)

RCS can clean up 5.2 in isolation MV: can we remove 'shall' simpler?
SM can remove word if not a req. RCS switch to 'is/are'
AB AxJ is editorial, no vote needed even, no papers just direction to editor
Decision want ~~remove redundant US from~~ update AxJ? as spec in N3447? +5, 0+0, 4+2
12 / 0 / 6 (Yes)

RB don't want out of sync w/ body → not out of sync if the assessment is true
Jens so 'constant expr' is a syntax term; e.g. an ICE cannot not have integer type
by its constructed definition; tautological 'shall'-reqs

N3448 ghosts II → officially never cross US
additional redundant uses of 'shall' b/c changes to model byte array have US value
Discussion RCS is this really US bdy? → odd meaning of words. The value is unspec, a value isn't a behaviour Jens: we can change the title of the section
AB unspec values are valid of the type - not known yet? correct doesn't change to unspec byte values - indet? → no narrative rep for bytes
Jens we can treat this as applying to character types but was couldn't understand
SM see padding bytes etc - already expressed here for value
ES allocations have no effective type; could remaining 'wobbly' wordings break anyone?
Jens no, already applied that change
DS traditionally the type of bytes is unsigned char AB: add a new value kind?
Jens adds ambiguity. This is for rep. bits, could refer to bits MV reuse wording
AB contradicts effective type rules
Jens - then same change to direct alloc, malloc. No indet. reps, again.
same potential for global change
MV access by type that does have trapreps? → diff't access, could be UB

RCS could accept 3.1 & leave 3.2 + 3.3 to editor → JM: vote on whole AC: →

Decision coat adopt dkgs spec in N3448 in CCY? 11 +4 / 0 +0 / 4+1 (Ver)

ES 'accesstype' can be added? JM: this is dst byte values

reserving N3447 - JM did we unintentionally add the SO to 3.2?
not sure this should be added & is clear requirement on impl

PK floods are guaranteed to represent zero Jens: compile & link environments can be different, hence adding this

note editor do not include newly added WB from N3447, or review & consult JM & CFP & Jens

PK please incorporate this part in the next version of this document

N3449 enhanced type variance
good ideas lead to & come from other good ideas. users can count away; partly b/c don't understand, partly b/c type sys isn't very good. Characterize a qualified pointed to as a subtype ↳ like addition to long. that won't fully thought essentially can't add qualification to argument pointed types c/o contravariants teachability is extremely poor. huge benefits to safety & usability

PK for this to work ptr to qual must have same representation but do not have coercion to enforce. assumption doesn't hold in general, only CUT-qual address space qual doesn't work this way at all; want to bring parts of address space TR into CCY to fix these kinds of issues. like idea but should list affected quals explicitly

CB thought that requirement existed

Me strong support; address spaces & atomic shouldn't be quals anyway

AC impl def qual list?

RR every impl has some of these non-qual qualifiers that would break by this

AB impl exp. CB: not impl before getting direction
couldn't be able to put this in w/ too much complexity but this would come straight out of Optional, to make usable

AB like research direction but not adopting a type sys change c/o exp

OT second this - type sys work before, need exp.
worry about breaking extensions for e.g. bounds safety

MV chicken-egg problem PK: slang lines indicates direction
DT get a GSoC paper or other way to prototype
AC lot of exp in GCC (and Clang) w/ -w-cost diagnostics
AB true about egg; uses start using features right away which punishes early adoption
less worried dbt prior art & more about changing extensions
CB not trying to force this in right now; respect impl concerns dbt this, right direction
reasonable request for formal proof
MV parts of this imitate C++ const conversion rules - could split out & adopt easily
RB if a customer demands this, no egg prob, we do it; haven't done yet
CB users have very low expectations of C, don't expect to get fixed
↳ 'ars do'
DT nice if we had list of lang. corner cases for people to look over [new website]

Opinion could sth slang lines N3449 in CLY? 11+4 / 0+1 / 7+1 (Yes)
15 / 1 / 8

looked right at you and said 'zero'
CB didn't like void ptr 9- AC: impldef?

Opinion could like another dir. extending pun subst to void*? 9+3 / 1+1 / 7+2 (Yes)
12 / 2 / 9

N3508 constant size objs
not considering count/sz/len! lot of syntax changes tho (using len), resolves
confusion btw expr & its value
RB same version as before? → yes

Decision adopt N3508 into CLY? count 14+6 / 1+0 / 2+0 (Yes)
20 / 1 / 2
pending objections b/w

RCS maybe delay everything? → would obstruct LSO generation
CB voice support - small & worthy improvement
AC 'arraylength' or 'element count' → lot of things could be better but

note change _Length of to _Countof in merge conflict

'Robert C Model'

N3357 Questions [WG14.29101]

response to negative vote: asking for feedback for improvement; tries to address comments RCS: 'model' is unclear, but repetition is annoying - try to coin a term like 'C model' PK so would in C Std
'C FP model' AB just opens weirdness - 'C integer model'? 'C ptr model'? etc. RCB - CFP mostly supports the change, 8-10 w/ 3-4 strong support & minor group oppose
RCS what abt section heading? RCB 7.12 would have a line introducing this in 'General' Seo: model itself is in 5.x
SHM shut that there are models outside the C model - put in Lib introduction that results of Lib fns can be unpredictable if doesn't follow core C model ie. if any exception, make it a blanket one
RCB so in 7 not 7.12 - core typedefs, float, etc as well
↳ direction for CFP

Array subs w/o decay [homework update]
make zero-indexed ordinals explicit in text - fine so long as consistent
RCS 'zeroth' comes from physics (no poll)

~~Decision~~ ~~adopt N... in CD?~~ (Yes)

Section 6 arrangements for future meetings
will consider options for Pittsburgh
strongly dependent on AV quality, shared mics etc
↳ not confirmed quite yet

8 Actions → reflector only
decisions reviewed online

Thanks

AB moves to adj, TS seconds

Adjourned!

Name	Organization	NB
Alex Celeste	Perforce Software	United States
Rajan Bhakta	IBM	" "
Henry Kleynhans	Bloomberg L.P.	" "
Javier Múgica		
Alejandro Colomar	Linux	Spain
Aaron ~~PETER~~ BACHMANN	~~ASTECKON~~ AG	ASI
Peter Eisentraut	EDB	Germany (guest)
David Tarditi	Apple	United States
Joseph Myers	Red Hat	UK
Yeoul Na	Apple	United States
Aaron Ballman	Intel	US
CHRIS BAZLEY	ARM	UK
Freek Wiedijk	Plum Hall	US
MARTIN UECKER	TU GRAZ	ASI
Philipp Krause	SDCC	Germany
Jean GUSTEDT	INRIA	AFNOR
DANIEL PLAKOSH	CMU/SEI	USA
Ryan Karl	CMU/SEI	USA
Robert Seacord	Woven	USA

PASS THAT WAY →