SG14

J Guy Davidson
Coding Manager
Creative Assembly

@hatcat01

Summary

WG21/SG14

Don’t pay for what you don’t use

Containers and algorithms

Parallelism and vectorisation

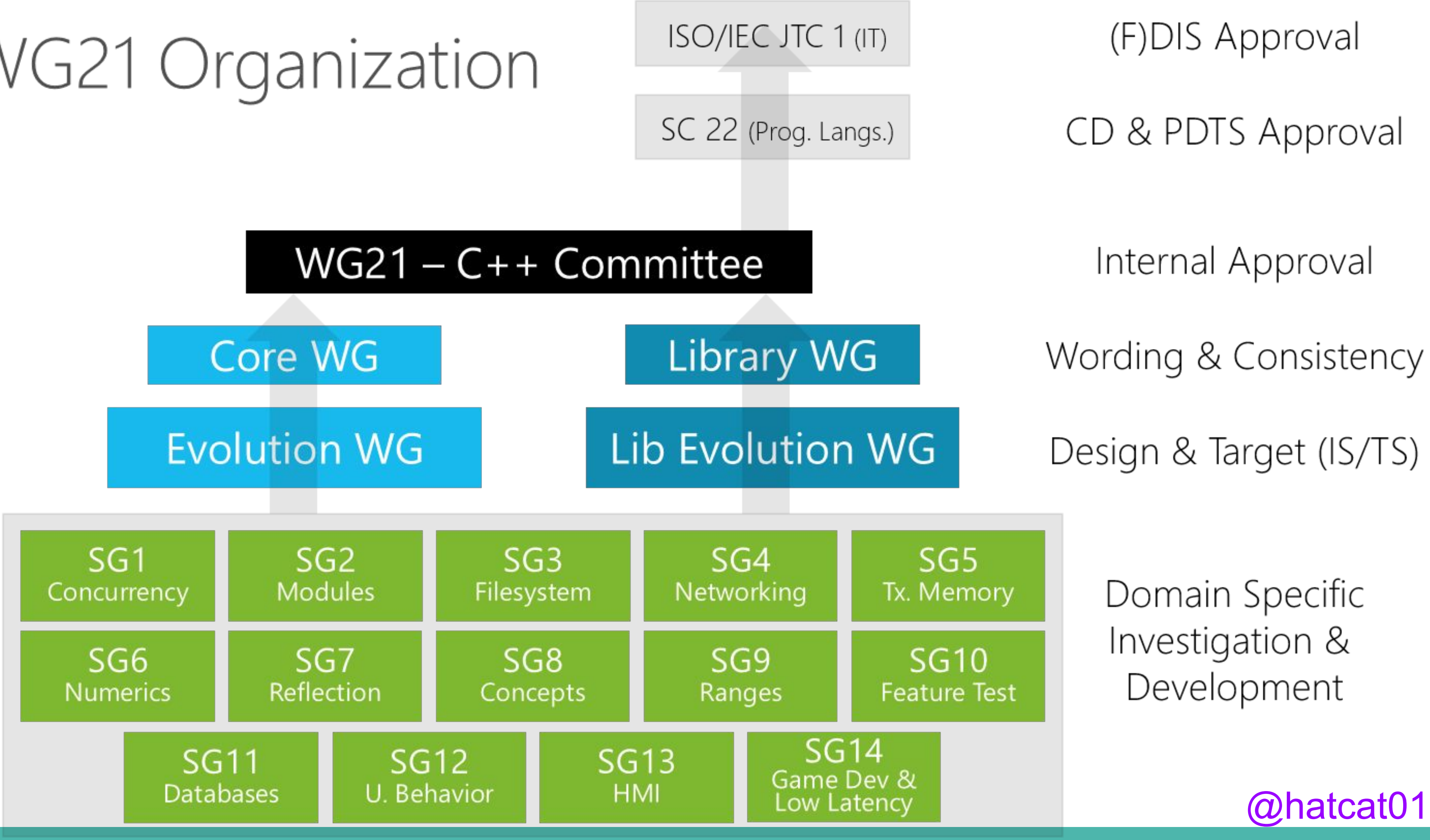
WG21/SG14

CppCon 2014

Low latency, real time

Games, simulations, financial trading, embedded systems

WG21 Organization



@hatcat01

WG21/SG14

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

Papers

GitHub: https://github.com/WG21-SG14/SG14

Telecons

VAMPIRE COUNTS TRAILER
TOTAL WAR
WARHAMMER

Some tricks

Run the world at 10Hz

Specify two cores

Use a GPU

Sound

Further constraints

CPU, RAM, GPU

Broad hardware range/single hardware specification

All x86-64 CPUs, Nvidia/ATI/Intel graphics parts

Don’t pay for what you don’t use

Exceptions

RTTI

The Standard Library

Memory constraints

Function calls

Exception costs

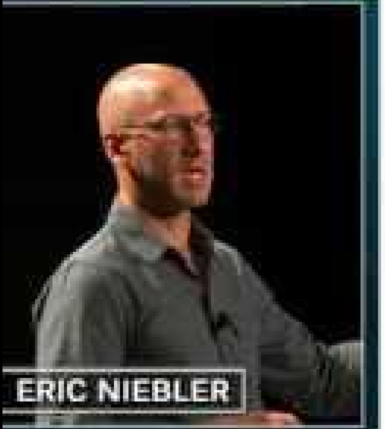
Deterministic destruction

Two ways out of a function

Two ways of creating the unwinding code

Patrice Roi:
http://h-deb.clg.qc.ca/Sujets/Developpement/Exceptions-Costs.html

cppcon

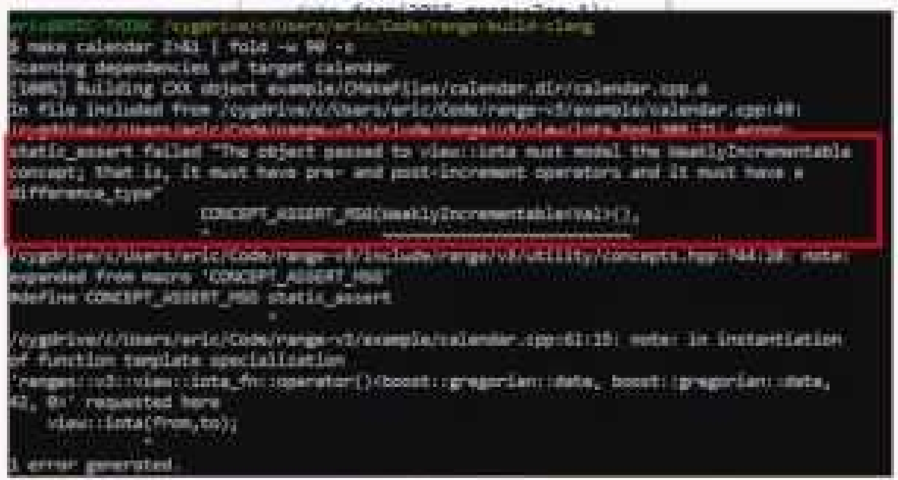

ERIC NIEBLER

Ranges for the Standard Library

www.CppCon.org

Range of dates = ☹

```
int main()
{
    date from(2015, greg::Jan, 1);
```



```
eric@ERIC-THINK /cygdrive/c/Users/eric/Code/range-build-clang
$ make calendar 2>&1 | fold -w 90 -s
Scanning dependencies of target calendar
[100%] Building CXX object example/CMakeFiles/calendar.dir/calendar.cpp.o
In file included from /cygdrive/c/Users/eric/Code/range-v3/example/calendar.cpp:40:
/cygdrive/c/Users/eric/Code/range-v3/include/range/v3/view/iota.hpp:300:71: error:
static_assert failed "The object passed to view::iota must model the WeaklyIncrementable
concept; that is, it must have pre- and post-increment operators and it must have a
difference_type"
                    CONCEPT_ASSERT_MSG(WeaklyIncrementable<Val>(),
                    ^                  ~~~~~~~~~~~~~~~~~~~~~~~~~~
/cygdrive/c/Users/eric/Code/range-v3/include/range/v3/utility/concepts.hpp:744:28: note:
expanded from macro 'CONCEPT_ASSERT_MSG'
#define CONCEPT_ASSERT_MSG static_assert
                           ^
/cygdrive/c/Users/eric/Code/range-v3/example/calendar.cpp:61:15: note: in instantiation
of function template specialization
'ranges::v3::view::iota_fn::operator()<boost::gregorian::date, boost::gregorian::date,
42, 0>' requested here
    view::iota(from,to);
          ^
1 error generated.
```

Copyright Eric Niebler 2015

Exception costs

Error handling comes with a cost

Non-determinism is VERY limited

Standard library has many exception-safe components

Having said all that...

RTTI

typeid(), dynamic_cast<>

Runtime cost

Not wanted on voyage

The consumer won...

...or did it?

The Standard Library

Exception safe

-fno_exceptions does not mean “No exception code”

Try-catch blocks are unwelcome

_HAS_EXCEPTIONS = 0

namespace foo nothrow { ... }

The Standard Library

Thread safe

Implemented for maintainability

Debug configuration can be slow

Roll your own containers

Memory constraints

Heap allocation is a headache

Assign budgets to systems

Fragmentation

Partition your allocations with allocators

Memory constraints

64 bit address space

Standard library objects

std::function<bool, int, size_t> func;

Rolling your own std::function is a fun hobby

Function calls

Inline depth

Virtual dispatch

Calling virtual functions on containers of pointers

Tradeoffs

Library extensions

Ring

Flat map and flat set

Uninitialised memory

Fixed point numbers

Ring

By your presenter and Arthur O'Dwyer

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2016/p0059r1.pdf

An ancient structure

A common structure

Parkinson's law of triviality

Asynchronous processing, history buffer

Contiguous

Ring

Started after ACCU May 2015

Presented to SG14 at CppCon September 2015

Presented to committee at Kona, Hawaii October 2015

Acquired a collaborator

Presented to committee at Jacksonville, Florida March 2016

Presented to SG14 at GDC March 2016

Ready for Oulu?

Flat map and set

By Sean Middleditch

https://github.com/seanmiddleditch/CPlusPlus/blob/flatmap-wording/flat_containers_redux.md

Cache-friendly

Interface decisions

Element storage

Design-complete

Uninitialised memory algorithms

Brittany Friedman

http://open-std.org/JTC1/SC22/WG21/docs/papers/2016/p0040r1.html

uninitialized_copy and uninitialized_copy_n

uninitialized_fill and uninitialized_fill_n

get_temporary_buffer and return_temporary_buffer

raw_storage_iterator

Uninitialised memory algorithms

destroy

uninitialized_move and uninitialized_move_n

uninitialized_value_construct

uninitialized_default_construct

Uninitialised memory algorithms

P0040	Dinkumware	libstdc++	libc++	EASTL
uninitialized_move	_Uninitialized_move	__uninitialized_move_a		uninitialized_move
uninitialized_move_n				
uninitialized_value_construct	_Uninit_def_fill_n (n-variant)	__uninitialized_default	see vector::__construct_at_end	uses uninitialised_fill
uninitialized_default_construct				
destroy	_Destroy_range	_Destroy	see vector::__destruct_at_end	destruct

Uninitialised memory algorithms

Exception handling

move_iterator + uninitialized_copy = uninitialized_move?

Bidirectional iterator destruction order

Uninitialised memory algorithms

Specialised array-based containers are now possible

Array of unique_ptr

New type traits

Relocatable types

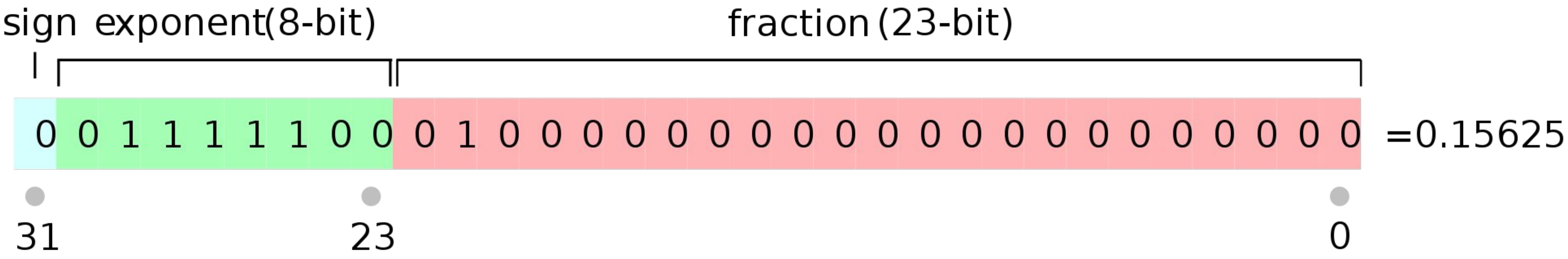
Fixed point numbers

Floating point numbers

binary32, binary 64

Not all processors offer native floating point registers

Uneven point distribution



sign exponent(8-bit) fraction (23-bit)

0 0 1 1 1 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 =0.15625

31 23 0

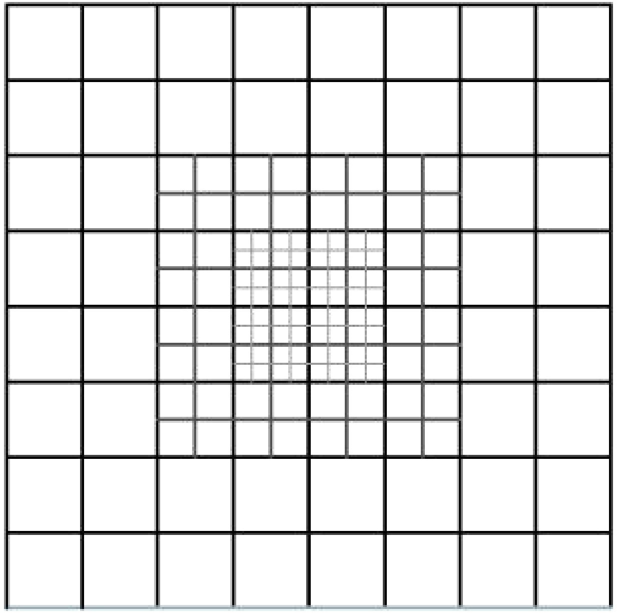
Floating point numbers

Five decimal orders of magnitude

Everything has a position

Combat = contact

Don’t fight at the edges



@hatcat01

Fixed point numbers

John Mcfarlane, Laurence Crowl

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2016/p0037r1.html
http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/p0106r0.html

SG6: Numerics

Library extension to <type_traits>

<fixed_point>

Fixed point numbers

```
template <class ReprType, int Exponent> class fixed_point;

template <unsigned IntegerDigits, unsigned FractionalDigits = 0,
          class Archetype = signed>
using make_fixed;

template <unsigned IntegerDigits, unsigned FractionalDigits = 0,
          class Archetype = unsigned>
using make_ufixed;

make_fixed<2,29> pi {3.141592653};

make_ufixed<4, 4> (0.006) == make_ufixed<4,4> (0)
```

Fixed point numbers

Promotion rules for operator overloads

If both arguments are fixed point:

Result type is the size of the larger type
Is signed if either input is signed
Has the maximum integer bits of the two inputs

Fixed point numbers

Promotion rules for operator overloads

If one argument is floating point type:

Result type is the smallest floating point type of equal or greater size than the inputs

Fixed point numbers

Promotion rules for operator overloads

If one argument is an integral type:

Result type is the other fixed point type

Fixed point numbers

For example:

```
make_ufixed<5, 3>{8} + make_ufixed<4, 4>{3} == make_ufixed<5, 3>{11};

make_ufixed<5, 3>{8} + 3 == make_ufixed<5, 3>{11};

make_ufixed<5, 3>{8} + float{3} == float{11};
```

Fixed point numbers

Overflow and underflow

```
make_fixed<4, 3>{15} + make_fixed<4, 3>{1}

make_fixed<6, 1>{15} / make_fixed<6, 1>{2}

make_fixed<7, 0>{15} / make_fixed<7, 0>{2}
```

Fixed point numbers

Leave it to the user. Caveat emptor.

Allow the user to provide a custom type for ReprType

Promote the result to a larger type

Adjust the exponent of the result upward

```
c = a + b;
a += b;
assert(c == a); // may fail
```

Fixed point numbers

```
promote(make_fixed<5, 2>{15.5});

make_fixed<11, 4>{15.5};

demote(make_fixed<11, 4>{15.5});
```

Fixed point numbers

trunc_reciprocal, trunc_square, trunc_sqrt
promote_reciprocal, promote_square

trunc_add, trunc_subtract, trunc_multiply, trunc_divide
trunc_shift_left, trunc_shift_right
promote_add, promote_subtract, promote_multiply, promote_divide

Parallelism

SIMD - Single Instruction Multiple Data

1997: MMX

1998: 3DNow!

Parallelism

1999: SSE

2001: SSE2

2004: SSE3

2007: SSE4

2011: AVX

2013: AVX2

Parallelism

No standard!

Boost.SIMD

Mathias Gaunard

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2016/p0203r0.html

Parallelism

```
template<class T, class X = /*implementation-defined ABI tag*/>
int best_size_v = /*implementation-defined*/;

template<class T, int N = best_size_v<T>, class X = /*impl-defined ABI tag*/>
struct simd_vector;

template <class T, int N>
simd_vector<T, N*2> combine(simd_vector<T, N> rhs, simd_vector<T, N> lhs);

template <class T, int N>
array<simd_vector<T, N/2>, 2> slice(simd_vector<T, N> a);
```

Parallelism

```
simd_vector<T, N> a;
simd_vector<U, N> b = simd_cast<U>(a);

template<int... I, class T>
simd_vector<T, sizeof...(I)> shuffle(simd_vector<T, sizeof...(I)> a);
template<int... I, class T>
simd_vector<T, sizeof...(I)> shuffle(simd_vector<T, sizeof...(I)> a,
                                     simd_vector<T, sizeof...(I)> b);
```

Parallelism

Aliasing:

```
void foo(float* aligned_data)
{
  simd_vector<float>* my_vector_data =
    reinterpret_cast<simd_vector<float>*>(aligned_data);
  // ... do stuff
}
```

Parallelism

Aliasing:

```
simd_vector<float> v;
float* p = &v[0];
p[3] = 42.0f;
```

Parallelism

Calling conventions

Compiler support required?

Heterogeneous computing

Massive parallelism

Head start in games...

Graphics cards

Heterogeneous computing

Direct3D

Nvidia GeForce

ATI Radeon

Heterogeneous computing

Agency, Jared Hoberock and Michael Garland

https://github.com/jaredhoberock/agency

bulk_invoke, bulk_async, bulk_then

Policies for parameterising control structures

Agents which parameterise user lambdas

Executors which create execution agents

Heterogeneous computing

```
void saxpy(float a, float* x, float* y, size_t n)
{
  using namespace agency;
  bulk_invoke(par(n), [=](parallel_agent& self)
  {
    auto i = self.index();
    x[i] = a * x[i] + y[i];
  });
}
```

Heterogeneous computing

```
std::future<void> saxpy(float a, float* x, float* y, size_t n)
{
  using namespace agency;
  return bulk_async(par(n), [=](parallel_agent& self)
  {
    auto i = self.index();
    x[i] = a * x[i] + y[i];
  });
}
```

Heterogeneous computing

```
std::future<void> saxpy(std::future<void>& dep, float a, float* x, float* y, size_t n)
{
  using namespace agency;
  return bulk_then(par(n), [=](parallel_agent& self)
  {
    auto i = self.index();
    x[i] = a * x[i] + y[i];
  }, dep);
}
```

Heterogeneous computing

Heterogeneous C++ compiler

Parallelism APIs in HPX

SYCL

Next big frontier

Finally...

Join the subgroup

Join any subgroup

Improve the standard

Thank you!

@hatcat01