I Just Wanted a Random Integer!

Cppcon 2016

Cheinan Marks

Spiral Genetics, Inc.

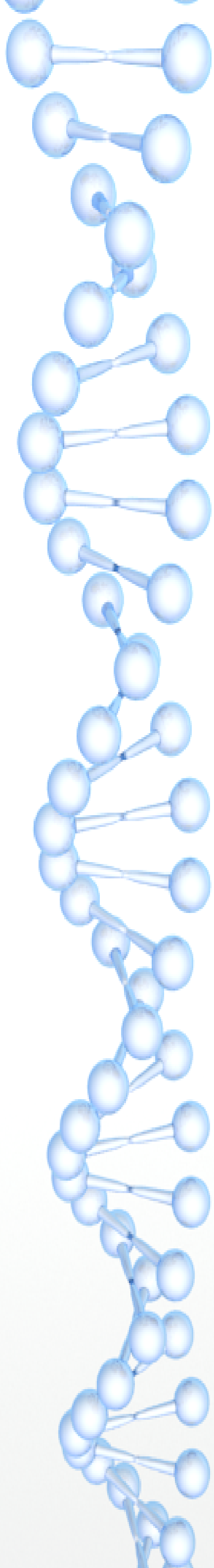
I Just Wanted a Random Integer!

ATTCTGTAGCGTGCATGCATGTCGAT

I Just Wanted a Random Integer!

ATTCTGTAGCGTGCATGCATGTCGAT

- Needed random sized DNA reads
- Wanted good coverage of size range
- Wanted to test edge cases



I Just Wanted a Random Integer!

```
#include <cstdlib>
auto r = std::rand() % 100;
```

I Just Wanted a Random Integer!

```
#include <cstdlib>

auto r = std::rand() % 100;
```

```
man -s3 rand
```

NOTES

The versions of rand() and srand() in the Linux C Library use the same random number generator as random(3) and srandom(3), so the lower-order bits should be as random as the higher-order bits.

However, on older rand() implementations, and on current implementations on different systems, the lower-order bits are much less random than the higher-order bits. Do not use this function in applications intended to be portable when good randomness is needed. (Use random(3) instead.)

I Just Wanted a Random Integer!

```
#include <cstdlib>
auto r = std::rand() % 100;
```

```
man -s3 rand
```

NOTES

The versions of rand() and srand() in the Linux C Library use the same random number generator as random(3) and srandom(3), so the lower-order bits should be as random as the higher-order bits.

However, on older rand() implementations, and on current implementations on different systems, **the lower-order bits are much less random than the higher-order bits. Do not use this function in applications intended to be portable when good randomness is needed. (Use random(3) instead.)**

I Just Wanted a Random Integer!

- It’s just a unit test
- std::rand should be good enough
- % should be good enough
- <random> is too complicated

I Just Wanted a Random Integer!

- It’s just a unit test
- ~~std::rand should be good enough~~ **NOPE**
- ~~% should be good enough~~ **NOPE**
- <random> is too complicated **Err, maybe**

I Just Wanted a Random Integer!

- It’s just a unit test
- ~~std::rand should be good enough~~ **NOPE**
- ~~% should be good enough~~ **NOPE**
- <random> is too complicated **Err, maybe**
- If you want even coverage of edge cases
- Then you **must** do better

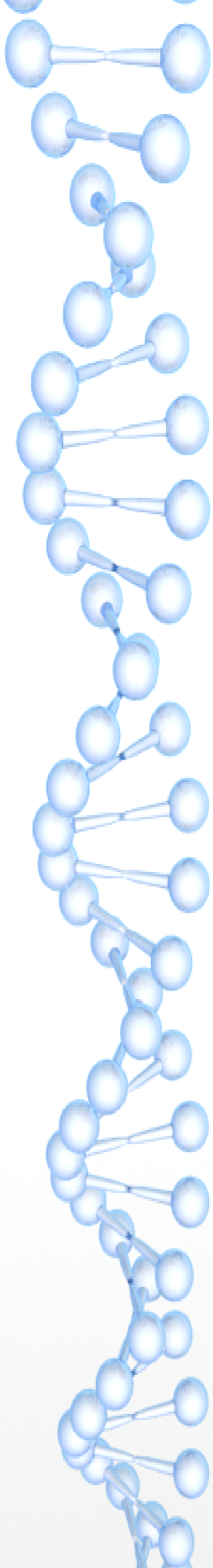
I Just Wanted a Random Integer!

1. Watch STL’s 2013 Going Native Talk
2. Use <random> and Mersenne Twister
3. Seed with std::random_device (entropy)
4. Use std::uniform_int_distribution
5. Profit

I Just Wanted a Random Integer!

Questions

- Entropy – what is it?
- What is std::random_device?
- Why avoid the stack for std::mt19937?
- Is std::uniform_int_distribution cheap to construct and use?



I Just Wanted a Random Integer!

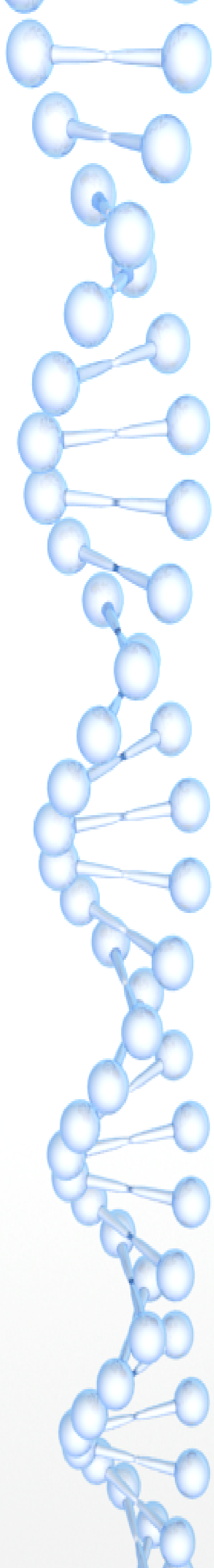
- Entropy?
- Comes from std::random_device
- Blocks when it runs out
- Wait, what?!?

I Just Wanted a Random Integer!

- Entropy is randomness
- Computers are not by nature random
- Computers can be pseudorandom
- Entropy is “true randomness”
- Hard to generate on a computer

I Just Wanted a Random Integer!

- Entropy?
- Comes from std::random_device
- Blocks when it runs out
- Wait, what?!?

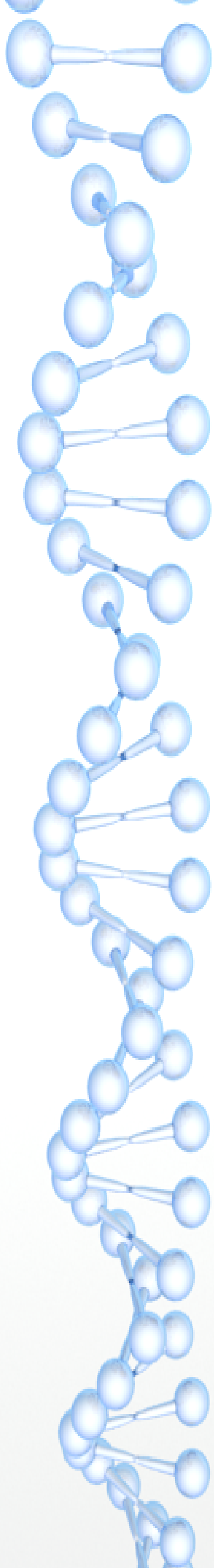


I Just Wanted a Random Integer!

- Entropy?
- Comes from std::random_device
- Blocks when it runs out
- Shrink person with die and whiteboard
- Nanoperson rolls die
- Nanoperson writes result on whiteboard
- Nano person returns number on demand

I Just Wanted a Random Integer!

- It takes finite time to roll die
- If demand > supply, entropy runs out
- Entropy generator might block until more entropy is available



I Just Wanted a Random Integer!

- std::random_device
- Can be used as a temporary
- Can use it as a generator
- operator() returns min() <=RN <= max()
- Hardware and implementation dependent
- CAUTION: Can throw
- Might be slow, might be pseudorandom
- Uses /dev/urandom on my machine

I Just Wanted a Random Integer!

- Generators
- Use std::mt19937 or std::mt19937_64
- Seed with std::random_device
- Pseudorandom
- Deterministic
- Fast

I Just Wanted a Random Integer!

Generate billion random ints

std::random_device	std::mt19937
44.3 seconds	3.6 seconds

I Just Wanted a Random Integer!

Generate billion random ints

std::random_device	std::mt19937
44.3 seconds	3.6 seconds

- std::random_device may be hardware
- Multithreading behavior unclear
- std::mt19937 can be thread local
- Initialize correctly!

I Just Wanted a Random Integer!

1. Watch STL’s 2013 Going Native Talk
2. Use Mersenne Twister in <random>
3. Seed with std::random_device (entropy)
4. Use std::uniform_distribution
5. ~~Profit~~ Think

I Just Wanted a Random Integer!

std::mt19937 is good, but...

I Just Wanted a Random Integer!

std::mt19937 is good, but…

5000 bytes on the stack

Slow to initialize

I Just Wanted a Random Integer!

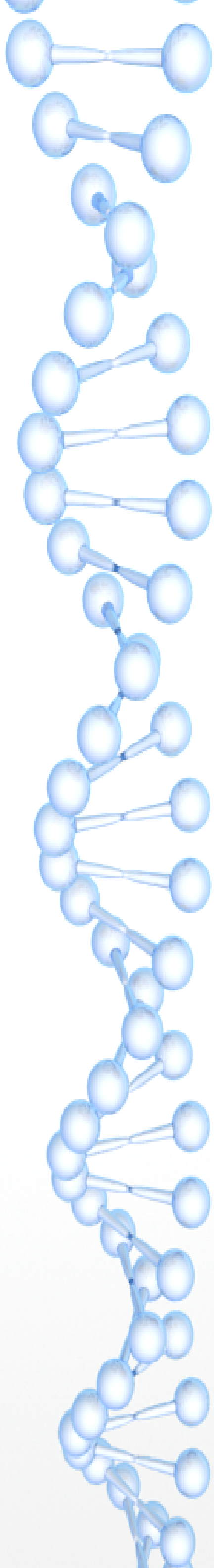
std::mt19937 is good, but…

5000 bytes on the stack

Slow to initialize

14.991 seconds for 1,000,000 inits

Slower than std::random_device!



I Just Wanted a Random Integer!

std::mt19937 is good, but…

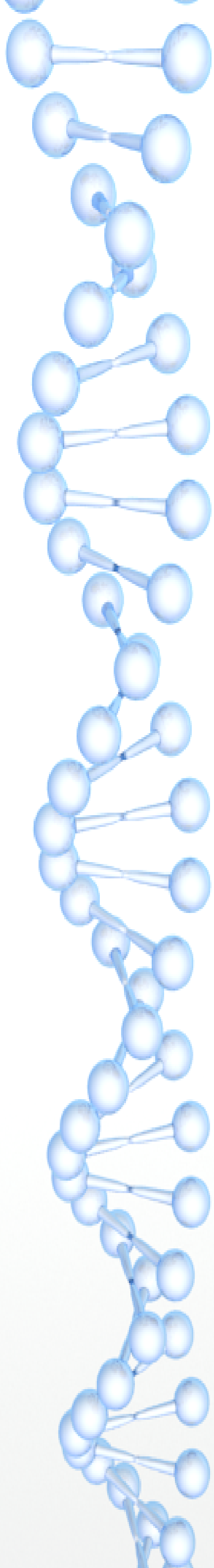
5000 (2504) bytes on the stack

Slow to initialize

15.0 seconds for 1,000,000 inits

```
void f() {
   std::mt19937 g(std::random_device{}());
   auto rn = g();
}
```

I Just Wanted a Random Integer!



std::mt19937 is good, but…

5000 (2504) bytes on the stack

Slow to initialize

15.0 seconds for 1,000,000 inits

```
void f() {
   static std::mt19937 g(std::random_device{}());
   auto rn = g();
}
```

I Just Wanted a Random Integer!

Guidelines Make Sense

- Use std::random_device to seed
- Keep std::mt19937 off stack

I Just Wanted a Random Integer!

Guidelines Make Sense

- Use std::random_device to seed
- Keep std::mt19937 off stack
- It can be static, thread local
- On stack, but beware construction

I Just Wanted a Random Integer!

Guidelines Make Sense

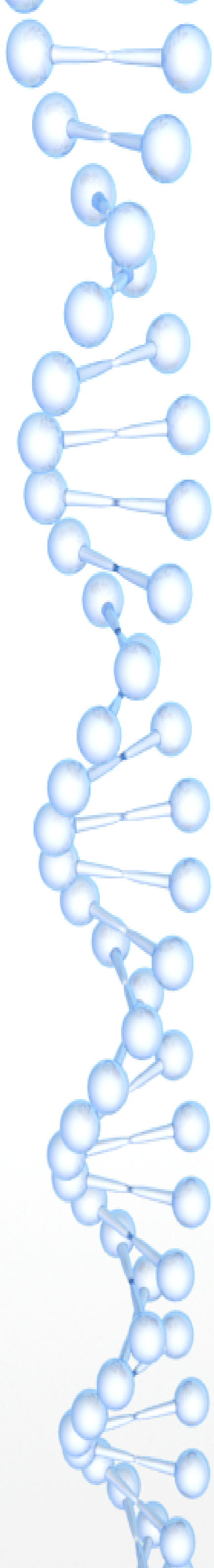
- Use std::random_device to seed
- Keep std::mt19937 off stack
- It can be static, thread local
- On stack, but beware construction
- std::minstd_rand is much faster
- std::minstd_rand cycle is smaller

I Just Wanted a Random Integer!

Guidelines Make Sense

- Use std::random_device to seed
- Keep std::mt19937 off stack
- It can be static, thread local
- On stack, but beware construction
- std::minstd_rand is much faster
- std::minstd_rand cycle is smaller

random_device	mt19937	minstd_rand
44.3 seconds	3.6 seconds	4.7 seconds



I Just Wanted a Random Integer!

Guidelines Make Sense

- Use std::random_device to seed
- Keep std::mt19937 off stack
- It can be static, thread local
- On stack, but beware construction
- ~~std::minstd_rand is much faster~~
- std::minstd_rand cycle is smaller

random_device	mt19937	minstd_rand
44.3 seconds	3.6 seconds	4.7 seconds

I Just Wanted a Random Integer!

Marin Mersenne 1588-1648

I Just Wanted a Random Integer!

$M_n = 2^n – 1$, prime n

$M_3 = 7$, $M_7 = 127$

…

$M_{74,207,281} \sim 10^{22,338,618}$

Marin Mersenne 1588-1648

I Just Wanted a Random Integer!

Marin Mersenne 1588-1648

$M_n = 2^n - 1$, prime n

$M_3 = 7$, $M_7 = 127$

…

$M_{74,207,281} \sim 10^{22,338,618}$

Why std::mt19937?

Period = $M_{19937} \sim 10^{6002}$

I Just Wanted a Random Integer!

Marin Mersenne 1588-1648

$M_n = 2^n - 1$, prime n

$M_3 = 7$, $M_7 = 127$

…

$M_{74,207,281} \sim 10^{22,338,618}$

Why std::mt19937?

Period = $M_{19937} \sim 10^{6002}$

```
typedef mersenne_twister_engine<uint_fast32_t, 32, 624, 397, 31,
0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15,
0xefc60000UL, 18, 1812433253UL> mt19937;
```

I Just Wanted a Random Integer!

$M_n = 2^n - 1$, prime n

$M_3 = 7$, $M_7 = 127$

…

$M_{74,207,281} \sim 10^{22,338,618}$

Why std::mt19937?

Period = $M_{19937} \sim 10^{6002}$

Marin Mersenne 1588-1648

M. Matsumoto and T. Nishimura, Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator, ACM Transactions on Modeling and Computer Simulation, Vol. 8, No. 1, January 1998, pp 3-30.

I Just Wanted a Random Integer!

Guidelines Make Sense

- Use std::random_device to seed
- Keep std::mt19937 off stack
- It can be static, thread local
- On stack, but beware construction
- ~~std::minstd_rand is much faster~~
- std::minstd_rand cycle is smaller

random_device	mt19937	minstd_rand
44.3 seconds	3.6 seconds	4.7 seconds

I Just Wanted a Random Integer!

Guidelines Make Sense

- Not so fast...

I Just Wanted a Random Integer!

STL Inspiration

I Just Wanted a Random Integer!

STL Inspiration

Melissa O’Neill, Harvey Mudd College

PCG http://www.pcg-random.org

https://www.reddit.com/r/programming/comments/2momvr/pcg_a_family_of_better_random_number_generators/

Search reddit.com for “PCG random”

http://preview.tinyurl.com/hyfb73l

I Just Wanted a Random Integer!

Melissa O’Neill, PCG

- Smaller (16 bytes) than std::mt19937
- C++ <random> compatible library
- https://github.com/imneme/pcg-cpp
- Faster than std::mt19937

I Just Wanted a Random Integer!

Melissa O’Neill, PCG

- Smaller (16 bytes) than std::mt19937
- C++ <random> compatible library
- https://github.com/imneme/pcg-cpp
- Faster than std::mt19937

random_device	mt19937	pcg_32
44.3 seconds	3.6 seconds	1.5 seconds

- Not in the standard
- Much less real-life experience

I Just Wanted a Random Integer!

std::uniform_int_distribution

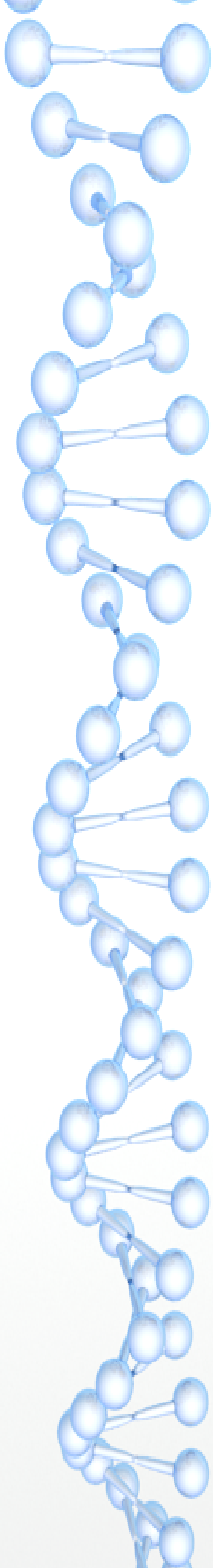
I Just Wanted a Random Integer!

std::uniform_int_distribution

Instantiate in inner loop?

I Just Wanted a Random Integer!

- std::uniform_int_distribution
- Instantiate in inner loop?
- Look into implementation
- Stores two template arguments
- Constructor is trivial



I Just Wanted a Random Integer!

std::uniform_int_distribution

Instantiate in inner loop?

Look into implementation

Stores two template arguments

Constructor is trivial

Operator() branches and calculates

Not much code

I Just Wanted a Random Integer!

```
static std::random_device entropySource;
static std::mt19937 randGenerator(entropySource());
std::uniform_int_distribution<int> theIntDist(0, 99);

for (auto i = 0; i < 1’000’000’000; i++) {
    volatile auto r = theIntDist(randGenerator);
}
// 23.4 seconds
```

I Just Wanted a Random Integer!

```
static std::random_device entropySource;
static std::mt19937 randGenerator(entropySource());

for (auto i = 0; i < 1’000’000’000; i++) {
    std::uniform_int_distribution<int> theIntDist(0, 99);
    volatile auto r = theIntDist(randGenerator);
}
```

I Just Wanted a Random Integer!

```
static std::random_device entropySource;
static std::mt19937 randGenerator(entropySource());

for (auto i = 0; i < 1’000’000’000; i++) {
    std::uniform_int_distribution<int> theIntDist(0, 99);
    volatile auto r = theIntDist(randGenerator);
}
// 5.1 seconds
```

I Just Wanted a Random Integer!

Constructor Outside Loop	Constructor Inside Loop
23.4 seconds	5.1 seconds

I Just Wanted a Random Integer!

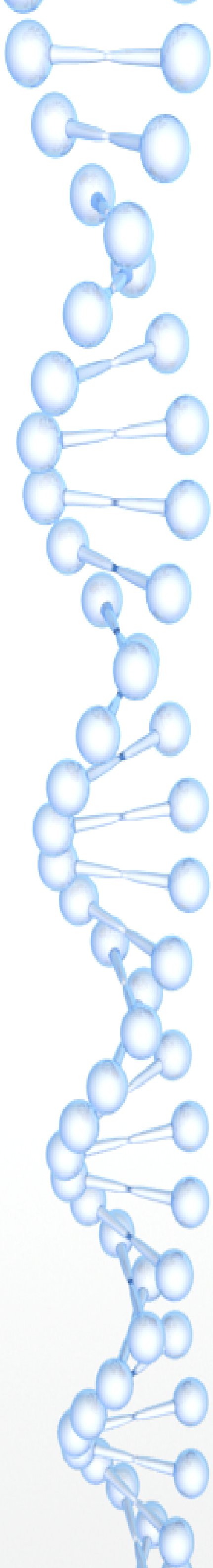
Constructor Outside Loop	Constructor Inside Loop
23.4 seconds	5.1 seconds
No Optimization	No Optimization
49.4 seconds	57.5 seconds

I Just Wanted a Random Integer!

```
const __uctype __urange
  = __uctype(__param.b()) - __uctype(__param.a());

__uctype __ret;

if (__urngrange > __urange)
  {
    // downscaling
    const __uctype __uerange = __urange + 1; // __urange can be zero
    const __uctype __scaling = __urngrange / __uerange;
    const __uctype __past = __uerange * __scaling;
    do
      __ret = __uctype(__urng()) - __urngmin;
    while (__ret >= __past);
    __ret /= __scaling;
  }
```

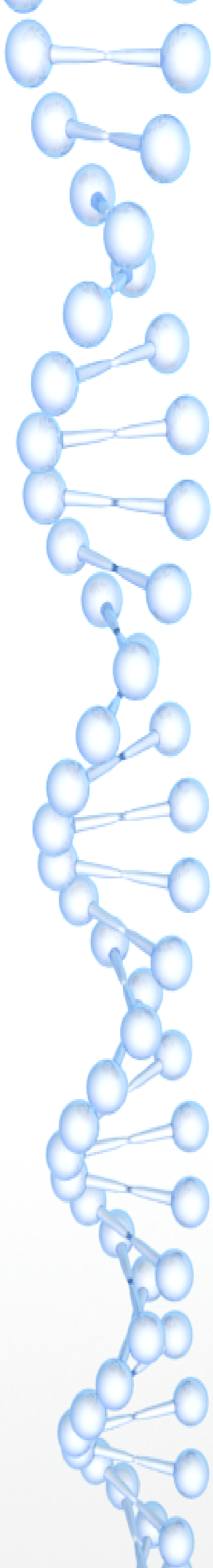


I Just Wanted a Random Integer!

```
const __uctype __urange
  = __uctype(__param.b()) - __uctype(__param.a());

__uctype __ret;

if (__urngrange > __urange)
  {
    // downscaling
    const __uctype __uerange = __urange + 1; // __urange can be zero
    const __uctype __scaling = __urngrange / __uerange;
    const __uctype __past = __uerange * __scaling;
    do
      __ret = __uctype(__urng()) - __urngmin;
    while (__ret >= __past);
    __ret /= __scaling;
  }
```



I Just Wanted a Random Integer!

Guidelines

I Just Wanted a Random Integer!

Guidelines

- Use your engineering judgment
- <random> is safe
- PCG is fast, small and simple
- Combine PCG with <random>
- Always measure. Always!

I Just Wanted a Random Integer!

Guidelines / Conclusions

- Use your engineering judgment
- <random> is safe
- PCG is fast, small and simple
- Combine PCG with <random>
- Always **measure**. Always!

I Just Wanted a Random Integer!

Conclusions

std::random_device::operator() to generate random numbers

std::mt19937 or PCG generators

Distributions are cheap to construct

Distributions are cheap to use

C++17 has std::sample

Benchmark your code

I Just Wanted a Random Integer!

?