`std::rand::random::<Talk>()`

Huon Wilson

December 18, 2014

http://huonw.github.io/rand-dec14/

Digital Randomness

A sequence of bits, e.g.

$$11011110\,11111000\,01001010\,00111100\ldots,$$

A sequence of bits, e.g.

$$\underbrace{11011110}_{222}\underbrace{11111000}_{248}\underbrace{01001010}_{74}\underbrace{00111100}_{60}\ldots,$$

Usually generated/consumed in chunks.

Why?

Lots of uses for randomness:

- simulations: scientific
- games: shuffling cards
- security: secret keys

All want "high quality" random numbers.

What is quality?

It depends!

Usually:

- uniformity: every bit has 50% chance of being 0 or 1
- unpredictability: the value of a bit can't be guessed base on the value of others

Determinism

Conventional computer RNGs follow patterns.

Obvious 0, 1, 2, 3, 4, . . . , 7, 0, 1, . . .

more examples

Rust

Rust

Thread-safety (by default)

Rust

SIMD: dSFMT

```
__m128i v, w, x, y, z;

// ...
x = a->si;
z = _mm_slli_epi64(x, DSFMT_SL1);
z = _mm_xor_si128(z, b->si);
y = _mm_xor_si128(y, z);

v = _mm_srli_epi64(y, DSFMT_SR);
w = _mm_and_si128(y, sse2_param_mask.i128);
v = _mm_xor_si128(v, x);
v = _mm_xor_si128(v, w);
r->si = v;
u->si = y;
```

http://www.math.sci.hiroshima-u.ac.jp/~%20m-mat/MT/SFMT/

```
__m128i v, w, x, y, z;

// ...
x = a->si;
z = _mm_slli_epi64(x, DSFMT_SL1);
z = _mm_xor_si128(z, b->si);
y = _mm_xor_si128(y, z);

v = _mm_srli_epi64(y, DSFMT_SR);
w = _mm_and_si128(y, sse2_param_mask.i128);
v = _mm_xor_si128(v, x);
v = _mm_xor_si128(v, w);
r->si = v;
u->si = y;
```

http://www.math.sci.hiroshima-u.ac.jp/~%20m-mat/MT/SFMT/

```
// ...
let y = (a << SSE2_SL) ^ b ^ y;
let v = (y >> SSE2_SR) ^ (y & SSE2_PARAMS_MASK) ^ a;

*r = v;
*u = y;
```

https://github.com/Grieverheart/dsfmt-rs

Rust

Traits & Type Inference

```
impl Rand for XorShiftRng
impl Rand for ChaChaRng
// ...
```

```
impl Rand for XorShiftRng
impl Rand for ChaChaRng
// ...
```

Get an RNG with a random seed:

```
use std::rand::{mod, XorShiftRng};

let mut rng: XorShiftRng = rand::random();
```