

# 猜数游戏—生成神秘数字

Rust 本身并没有生成随机数的功能 但官方提供了一个库 用于生成随机数—Rand

**rand 0.9.0**

Random number generators and other randomness functionality.

Documentation

Crate Source Builds Feature flags

**Coverage**

100%

120 out of 120 items documented

44 out of 67 items with examples

**Size**

Source code size: 400.90 kB

Documentation size: 23.54 MB

**Links**

Homepage

rust-random/rand

1764 445 19

crates.io

**Dependencies**

log ^0.4.4 optional

rand\_chacha ^0.9.0 optional

rand\_core ^0.9.0

**Rand**

Rand is a set of crates supporting (pseudo-)random generators:

- Built over a standard RNG trait: `rand_core::RngCore`
- With fast implementations of both strong and small generators: `rand::rngs`, and more RNGs: `rand_chacha`, `rand_xoshiro`, `rand_pcg`, `rngs` repo
- `rand::rng` is an asymptotically-fast, automatically-seeded and reasonably strong generator available on all std targets
- Direct support for seeding generators from the `getrandom` crate

With broad support for random value generation and random processes:

- StandardUniform random value sampling, Uniform-ranged value sampling and more
- Samplers for a large number of non-uniform random number distributions via our own `rand_distr` and via the `statsrs`
- Random processes (mostly choose and shuffle) via `rand::seq` traits

All with:

- Portably reproducible output
- #[no\_std] compatibility (partial)
- Many performance optimisations thanks to contributions from the wide user-base

Rand is not:

- Small (LoC). Most low-level crates are small, but the higher-level `rand` and `rand_distr` each contain a lot of functionality.
- Simple (implementation). We have a strong focus on correctness, speed and flexibility, but not simplicity. If you prefer a

Cargo.lock 是第一次运行 cargo build 时出现的 表示了这些依赖项所有的版本

```
guessing_game > Cargo.lock
1  # This file is automatically @generated by Cargo.
2  # It is not intended for manual editing.
3  version = 4
4
5  [[package]]
6  name = "bitflags"
7  version = "2.8.0"
8  source = "registry+https://github.com/rust-lang/crates.io-index"
9  checksum = "8f68f53c83ab957f72c32642f3868e03eb974d1fb82e453128456482613d36"
10
11 [[package]]
12 name = "byteorder"
13 version = "1.5.0"
14 source = "registry+https://github.com/rust-lang/crates.io-index"
15 checksum = "1fd0f2584146f6f2ef48085050886acf353beff7305ebd1ae69500e27c67f64b"
16
17 [[package]]
18 name = "cfg-if"
19 version = "1.0.0"
20 source = "registry+https://github.com/rust-lang/crates.io-index"
21 checksum = "baf1de4339761588bc0619e3cbc0120ee582ebb74b53b4efbf79117bd2da40fd"
22
```

再次 build 的时候 如果 lock 文件已经存在 就不用再去找一遍这些依赖项

```
Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.27s
PS D:\rust\guessing_game> cargo build
Compiling guessing_game v0.1.0 (D:\rust\guessing_game)
Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.25s
PS D:\rust\guessing_game>
```

如图可见 cargo 编译速度快了零点几秒  
更新 cargo 包

```
PS D:\rust\guessing_game> cargo update
Updating crates.io index
Locking 0 packages to latest compatible versions
```

这种升级的规则是升级小版本号 比如把 rand 从 0.3.13 升级到 0.3.14 这个命令不会使 rand 升级到 0.4 如果要升级到 0.4 要到 toml 文件中对于包的版本进行修改

```
Cargo.toml
1  [package]
2  name = "guessing_game"
3  version = "0.1.0"
4  authors = ["dave <admin@rust.com>"]
5  edition = "2018"
6
7  # See more keys and their definitions at https://doc.rust-lang.org
8
9  [dependencies]
10 rand = "0.4.14"
```

use rand::Rng;

trait 可以看作是其他语言的接口 会定义很多方法 Rng 是随机数生成器的方法

```
fn main() {
    println!("猜数游戏! !");

    let secret_number = rand::thread_rng()

    println!("猜测一个数");

    let mut guess = String::new();

    io::stdin().read_line(&mut guess)
        // io::Result Ok, Err

    println!("你猜测的数是: {}", guess);
}
```

**pub fn thread\_rng() → ThreadRng**  
Retrieve the lazily-initialized thread-local random number generator, seeded by the system. Intended to be used in the method chaining style, e.g. `thread_rng().gen_range(0, 100)`, or cached locally, e.g. `let mut rng = thread_rng();`. Invoked by the `Default` trait, `ThreadRng::default()` equivalent.  
For more information see [ ThreadRng ].

运行过程中发现了奇怪的报错 从未出现过

```
note: method defined here
--> C:\Users\wujia\.cargo\registry\src\index.crates.io-6f17d22bba15001f\rand-0.9.0\src\rng.rs:334:8
334 |     fn gen_range<T, R>(&mut self...
    |         ^^^^^^^^^
help: remove the extra argument
6   - let secret_number = rand::thread_rng().gen_range(0, 100);
```

### 1.3 警告：使用了废弃的函数

```
warning: use of deprecated function `rand::thread_rng`: renamed to `rng`
--> src/main.rs:6:31
|
6 | ...r = rand::thread_rng().gen_range...
|           ^^^^^^^^^^^
|
= note: `[warn(deprecated)]` on by default

warning: use of deprecated method `rand::Rng::gen_range`: Renamed to
`random_range`
--> src/main.rs:6:44
|
6 | ...read_rng().gen_range(1,101);
|           ^^^^^^^^^^^
```

这些警告表明你使用的 `rand::thread_rng` 和 `gen_range` 方法已经被废弃，建议使用新的方法。

## Tips

因 `rand` 版本过新 旧的函数已经被废弃引起 这个月已经是第二次出现这种问题了  
python 的版本问题搞了两天 卸了重装卸了重装 可见选用最新发布的版本未必是好事 应该找一个风评最好最稳定的版本进行使用

如果没有提前清楚关于 `cargo` 这几个文件的知识 我想到这我应该已经开始晕头转向了  
`rust` 不像 `c` 语言和 `python` 他对于底层的构建十分清晰 所以有上手难度 但通过这段时间的学习 我猜想其他语言的编译和运行可能和 `rust` 经历的过程相似 只是不那么透明 更简洁一些

## 运行结果

```
● Finished `dev` profile [unoptimized + debuginfo]
target(s) in 0.07s
Running `target\debug\guessing_game.exe`
猜数！
神秘数字是37
猜测一个数
325
你猜测的数是:325
```