开源社
kaiyuanshe

Rust 忽略的二三事

Ignore Things in Rust Test

Antonio from COSCUP

目录

CONTENTS

Test in Rust

- Test cases
- Examples
- Documents
- Features

Test in Rust - Basic

开源社 kaiyuanshe

Cargo test

```rust
1 ///```
2 /// assert_eq!(test_lib::add(2, 2), 4);
3 ///```
4 pub fn add(a: usize, b: usize) -> usize {
5     a + b
6 }
7
8 #[test]
9 fn it_works() {
10     assert_eq!(add(2, 2), 4);
11 }
```

```
running 1 test
test it_works ... ok

test result: ok. 1 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s

   Doc-tests test-lib

running 1 test
test src/lib.rs - add (line 1) ... ok

test result: ok. 1 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.12s
```

Test in Rust – With Filter

Cargo test **add_**

```rust
#[test]
fn add_1() {
    assert_eq!(add(1, 1), 2);
}

#[test]
fn add_2() {
    assert_eq!(add(2, 2), 4);
}

#[test]
fn reduce_1() {
    assert_eq!(reduce(1, 1), 0);
}
```

```
running 2 tests
test add_1 ... ok
test add_2 ... ok

test result: ok. 2 passed; 0 failed; 0 ignored; 0 measured; 1 filtered out; finished in 0.00s
```

Test in Rust – By Example

Cargo test **add_**

```rust
#[test]
fn add_1() {
    assert_eq!(add(1, 1), 2);
}

#[test]
fn add_2() {
    assert_eq!(add(2, 2), 4);
}

#[test]
fn reduce_1() {
    assert_eq!(reduce(1, 1), 0);
}
```

```
running 2 tests
test add_1 ... ok
test add_2 ... ok

test result: ok. 2 passed; 0 failed; 0 ignored; 0 measured; 1 filtered out; finished in 0.00s
```

Test in Rust – By Features

开源社 kaiyuanshe

Cargo test **--features=redudant**

```rust
#[test]
fn add_1() {
    assert_eq!(add(1, 1), 2);
}

#[cfg(feature = "redundant")]
#[test]
fn add_2() {
    assert_eq!(add(2, 2), 4);
}

#[test]
fn reduce_1() {
    assert_eq!(reduce(1, 1), 0);
}
```

```
running 3 tests
test add_2 ... ok
test add_1 ... ok
test reduce_1 ... ok

test result: ok. 3 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s
```

Test in Rust

开源社 kaiyuanshe

It is awesome, but…

manually not smart

Test in Rust

开源社 kaiyuanshe

Could the test case be smart?

Why just cargo test

Test in Rust

Just cargo test

```
running 4 tests
test tests::cpu_core_test_ignored ... ignored, because the cpu core less than 32
test tests::mem_test_ignored ... ignored, because the memory less than 999GB
test tests::physical_cpu_core_test_ignored ... ignored, because the physical cpu core less than 32
test tests::swap_test_ignored ... ignored, because the swap less than 999GB

test result: ok. 0 passed; 0 failed; 4 ignored; 0 measured; 0 filtered out; finished in 0.00s
```

it is smart

and it knows it should run or not,

and report the status

Things Rust Ignored

- Ignored message
- Check condition

Things Rust Ignored

```
#[ignore=”because the cpu core less than 32”]
fn test() {...}
```

```
running 4 tests
test tests::cpu_core_test_ignored ... ignored, because the cpu core less than 32
test tests::mem_test_ignored ... ignored, because the memory less than 999GB
test tests::physical_cpu_core_test_ignored ... ignored, because the physical cpu core less than 32
test tests::swap_test_ignored ... ignored, because the swap less than 999GB

test result: ok. 0 passed; 0 failed; 4 ignored; 0 measured; 0 filtered out; finished in 0.00s
```

Things Rust Ignored

Ignore message works after 2022

Provide ignore message in the result of test #92714

Merged bors merged 1 commit into `rust-lang:master` from `yanganto:ignore-message` on Feb 25, 2022

Conversation 79 | Commits 1 | Checks 0 | Files changed 7

yanganto commented on Jan 10, 2022 Contributor

Provide ignore the message in the result of the test.

This PR does not need RFC, because it is about the presentation of the report of `cargo test`.

However, the following document listed here helps you to know about

- RFC
- Rendered
- Previous discussion on IRLO

If there is something improper, please let me know.
Thanks.

1

Show ignore message in console and json output #94566

Merged bors merged 4 commits into `rust-lang:master` from `yanganto:show-ignore-message` on Mar 29, 2022

Conversation 25 | Commits 4 | Checks 0 | Files changed 9

yanganto commented on Mar 3, 2022 Contributor

- Provide ignore the message in console and JSON output
- Modify the ignore message style in the log file

related: #92714

Land on Rust 1.61
#92714 #94566
Rfc #94566

Things Rust Ignored

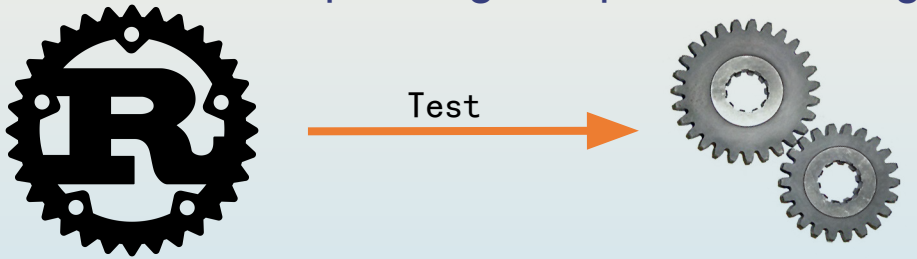
开源社 kaiyuanshe

Great, then we need the other part

check condition in test

Things Rust Ignored

- Run test when corresponding component existing



- Ignore when corresponding component is non-existing

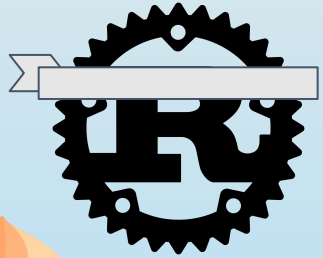
Things Rust Ignored

- Run test when **envar** existing

Test

```rust
#[test]
fn foo() {
    if env::var("FOO").is_none() { return; }
    let foo = env::var("FOO").unwrap();
}
```

- Ignore when **envar** absent



Issue #68007 opened in 2020

Things Rust Ignored

开源社 kaiyuanshe

It is 2023, but…

we can not check condition in test

and why?

Things Rust Ignored

开源社 kaiyuanshe

Each Rust test will be function,

and return a Result<...>

Things Rust Ignored

开源社 kaiyuanshe

Each Rust test will be function,

and return a Result<...>

Ok() map to **Pass**, **Err()** map to **Fail**

Things Rust Ignored

开源社 kaiyuanshe

Ignore should be decide before function build

Each Rust test will be function,

and return a Result<...>

Things Rust Ignored

开源社 kaiyuanshe

Wait!

There is still something similar call

panic!

It happened when function runs

Things Rust Ignored

```
fn some_function () {
  panic!("some message")
}
```

```
#[test]
fn test_case () {
  ignore!("some message")
}
```

Things Rust Ignored

Parse syntax & Build

Run testcase

Parse return & summary

```
#[test]
#[ignore]
fn test_case () {

}
```

```
#[test]
fn test_case () {

  ignore!("some message")

}
```

Things Rust Ignored

It seems wonderful, but still break the assumption

Things Rust Ignored

It seems wonderful, but still break the assumption

Each Rust test will be function, and return a Result<...>

~~Ok() map to Pass, Err() map to Fail~~

What ignore should map to?
Cargo does not support this.

Things Rust Ignored

Also, break the other assumption

~~Ignore should be decide before function build~~

Each Rust test will be function,

and return a Result<...>

Things Rust Ignored

Because breaking assumption (ignored test is static),
it will be breaking change

Run ignored and not ignored tests

```
cargo test -- --include-ignored
```

Run only ignored tests

```
cargo test -- --ignored
```

Things Rust Ignored

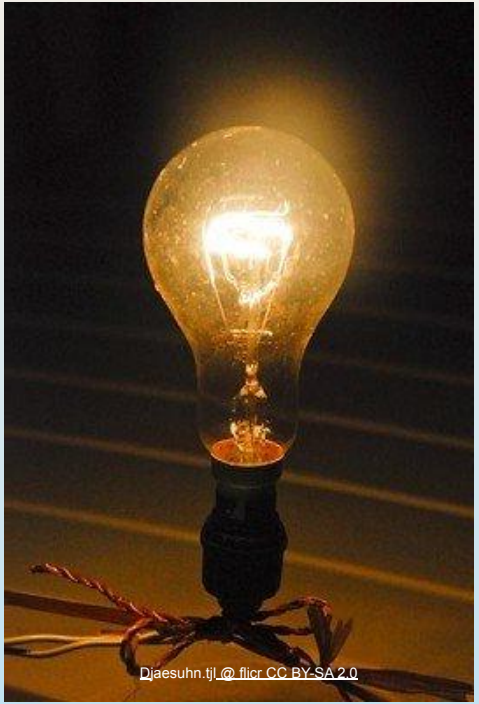
开源社 kaiyuanshe

Hard to let Rust have a **breaking change**,

so do it in a crate

Quick Solution

- Easy to use in CI
- No extra dependency on release product
- No extra test runner
- Still `cargo test`



Quick Solution

Parse syntax & Build

Run testcase

Parse return & summary

Check condition when build

```
#[test]
#[ignore]
fn test_case () {

}
```

Quick Solution

```
// Cargo.toml
[dev-dependencies]
test-with = "0.7.5"
```

```rust
// PWD environment variable exists
#[test_with::env(PWD)]
#[test]
fn test_works() {
    assert!(true);
}
```

```
running 4 tests
test tests::cpu_core_test_ignored ... ignored, because the cpu core less than 32
test tests::mem_test_ignored ... ignored, because the memory less than 999GB
test tests::physical_cpu_core_test_ignored ... ignored, because the physical cpu core less than 32
test tests::swap_test_ignored ... ignored, because the swap less than 999GB

test result: ok. 0 passed; 0 failed; 4 ignored; 0 measured; 0 filtered out; finished in 0.00s
```

Conditions: envar, file, http service, tcp socket, user, cpu, .. etc.

Quick Solution

开源社
kaiyuanshe

It is good,

and what is the limitation.

Quick Solution

- `cargo test` when **web service** absent

```rust
// https service exists
#[test_with::https(www.rust-lang.org)]
#[test]
fn test_works() {
    assert!(true);
}
```

- Turn on the **web service**, and `cargo test` again

The code did not change,
so it will **not** build still
ignored

Quick Solution

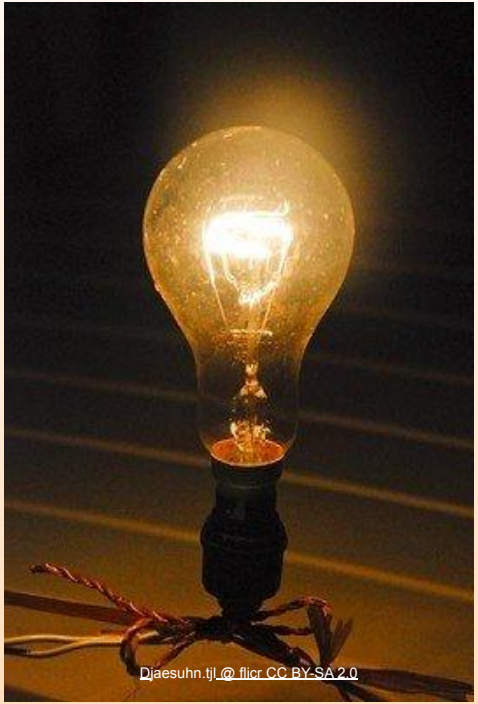
开源社
kaiyuanshe

We need a test runner,

where should we put in?

Example with Test Runner

- Put test in example
- Provide test runner in example
- Check condition in runtime



Example with Test Runner

```
cargo run --example
run_test
```

```
// Cargo.toml
[dependencies]
test-with = "0.11.0"
libtest-with = "0.6.1-2"
```

```rust
// example/run_test.rs
test_with::runner!(net, custom_mod);

#[test_with::module]
mod net {
    #[test_with::runtime_http(httpbin.org)]
    fn http_test_works() {
        assert!(true);
    }
}

fn something_happened() -> Option<String> {
    Some("because something happened".to_string())
}

#[test_with::module]
mod custom_mod {
    #[test_with::runtime_ignore_if(something_happened)]
    fn test_ignored() {
        assert!(false);
    }
}
```

Example with Test Runner

开源社 kaiyuanshe

The example output is the same as cargo test

```
└─[$] <git:(main*)> cargo run --example run_test
   Compiling runner v0.1.0 (/home/yanganto/data/side-project/test-with/examples/runner)
    Finished dev [unoptimized + debuginfo] target(s) in 1.15s
     Running `target/debug/examples/run_test`

running 2 tests
test test_ignored    ... ignored, because something happened
test http_test_works ... ok

test result: ok. 1 passed; 0 failed; 1 ignored; 0 measured; 0 filtered out; finished in 0.85s
```

Example with Test Runner

Still some trad-off

- extra dependency
  - can be avoided after rfc-3424
- put test case in example
- does not execute with cargo test

THANK YOU

QUESTIONS?

微信公众号：开源社
KAIYUANSHE
视频号：开源社KAIYUANSHE
新浪微博：开源社
B站：开源社KAIYUANSHE

简书：开源社
头条：开源社
Facebook：KaiyuansheChina
Twitter：开源社
KAIYUANSHE

扫码关注开源社公众号

扫码添加讲师联系方式