How do Rust closures work and how does it execute a closure?

Asked 4 years, 4 months ago

Active 4 years, 4 months ago

Viewed 889 times

4

Does it create a new thread and then execute that anonymous function inside the new thread?

I noticed many ownership / borrowing restrictions when I'm working with a closure. For example, if I have `Fn()`, I cannot pass a mutable variable inside the closure or it needs to be wrapped with a `Mutex`:

```
fn helloworld(f: &Fn(f64)) {
    f(42f64);
}

pub fn main() {
    let mut killer = 2;
    helloworld(&|n| {
        println!("{}", n);
        killer += 1;
    });
}
```

If a closure can be unsafe like that then something asynchronous or parallel is going on behind the scene and that's why Rust compiler doesn't let me to compile such code.

I might just be confused because I'm coming from a JavaScript / Python world and things are completely different there.

multithreading rust closures

Share
Improve this question
Follow

edited Aug 29 '17 at 12:54

Shepmaster
305k ● 59 ● 824 ● 1083

asked Aug 29 '17 at 9:14

Afshin Mehrabani
29.6k ● 26 ● 121 ● 191

1

A closure is an anonymous function with variable capture, it is not multithreaded (unless you ask for it). Moreover, could you post the code you refer to, when you say "if I have `Fn()`, I cannot pass a mutable variable inside the closure or it needs to be wrapped with a `Mutex`"?
– Boiethios
Aug 29 '17 at 9:17

@Boiethios sure, just updated the question with a link.
– Afshin Mehrabani
Aug 29 '17 at 9:21

You need to use `FnMut`: play.rust-lang.org/… But I let someone with more knowledge to answer with detailed explanations.
– Boiethios
Aug 29 '17 at 9:25

@Boiethios that is correct, changing to FnMut or using a Mutex can solve the problem but my question is, why? If it is not a separate thread, why it should be even important to pass a Mutex?
– Afshin Mehrabani
Aug 29 '17 at 9:29

Add a comment

1 Answer

Active Oldest Votes

21

There are two layers to this question.

First, a closure in Rust is just an anonymously-defined type that implements one or more "callable" traits. For example, this:

```
fn main() {
    let a = 6;
    let closure = |b| {
        println!("product is: {}", a * b);
    };
    closure(7);
}
```

is de-sugared into something similar to:

```
fn main() {
    let a = 6;
    let closure = {
        struct Closure<'a> {
            a: &'a i32,
        }
        impl<'a> Fn<(i32,)> for Closure<'a> {
            extern "rust-call" fn call(&self, (b,): (i32,)) {
                println!("product is: {}", (*self.a) * b);
            }
        }
        impl<'a> FnMut<(i32,)> for Closure<'a> {
            extern "rust-call" fn call_mut(&mut self, args: (i32,)) {
                self.call(args)
            }
        }
        impl<'a> FnOnce<(i32,)> for Closure<'a> {
            type Output = ();
            extern "rust-call" fn call_once(self, args: (i32,)) {
                self.call(args)
            }
        }
        Closure {
            a: &a,
        }
    };
    FnOnce::call_once(closure, (7,));
}
```

> **Note**: the above code relies on unstable, internal details and will not work on a stable compiler. It is provided for explanation only; you should **not** use this pattern yourself.

There's no threading involved, and nothing magical is happening. They boil down to a regular function call with an extra initial "context" argument.

This brings us to the second layer, which is why your *specific* code doesn't work: *because you told the compiler to forbid it*. One critical concern for callables is how the context is passed to the callable's code. This is represented by the `Fn`, `FnMut` and `FnOnce` traits (which are explained in the answer to the question When does a closure implement Fn, FnMut and FnOnce?). By taking `&Fn(f64)`, you've restricted yourself to only accepting closures which require immutable access to their context.

If you want a closure to be able to mutate its context, you need to use `FnMut` instead. Or, if you only need to call a closure once, you can use `FnOnce` (although not as a trait object like you're doing in your example).

Share
Improve this answer
Follow

edited Aug 29 '17 at 12:55

Shepmaster
305k ● 59 ● 824 ● 1083

answered Aug 29 '17 at 9:38

DK.
47.4k ● 3 ● 150 ● 142

2

Oh wow, this is a nice and clear explanation. Thank you so much DK.
– Afshin Mehrabani
Aug 29 '17 at 9:59

Add a comment

Your Answer

Post Your Answer

By clicking "Post Your Answer", you agree to our terms of service, privacy policy and cookie policy

Not the answer you're looking for? Browse other questions tagged multithreading rust closures or ask your own question.

The Overflow Blog

- ✎
  Sequencing your DNA with a USB dongle and open source code
- ✎
  Don’t push that button: Exploring the software that flies SpaceX rockets and...

Featured on Meta

- Providing a JavaScript API for userscripts
- Congratulations to the 59 sites that just left Beta

Linked

When does a closure implement Fn, FnMut and FnOnce?

Are Rust closures stack-allocated or heap-allocated by default?

"the `and_then` method cannot be invoked on a trait object" error when type annotations are added to the `and_then`

Related

How do JavaScript closures work?

What is the difference between a 'closure' and a 'lambda'?

When and how should I use a ThreadLocal variable?

In PHP, what is a closure and why does it use the "use" identifier?

How do servlets work? Instantiation, sessions, shared variables and multithreading

C++11 introduced a standardized memory model. What does it mean? And how is it going to affect C++ programming?

Why doesn't println! work in Rust unit tests?

When does a closure implement Fn, FnMut and FnOnce?

Method not compatible with trait with confusing error message

Rust cloned closures expected closure, found different closure

Hot Network Questions

How to install a package via `apt-get` without flagging it as manually installed

Help identify a short story about professor using voodoo doll to prevent the marriage of a much younger woman he loves - by Henry Slesar

Is the science in "Don't Look Up" realistic?

Why couldn't Smith absorb Neo in The Matrix Reloaded?

Using a friend to move cash into my checking account

more hot questions

Question feed