



Products

Sharing String between threads in Rust

Log in Sign up

Ask Question

Asked 4 years, 10 months ago

Active 4 years, 10 months ago

Viewed 3k times



3



1



I'm trying to request multiple URLs with multiple `std::thread`. This is how my code looks so far:

```
fn fetch(urls: Vec<&str>) {
    let (tx, rx) = mpsc::channel();

    for url in urls {
        let tx = tx.clone();

        thread::spawn(|| {
            let ssl = NativeTlsClient::new().unwrap();
            let connector = HttpsConnector::new(ssl);
            let client = Client::with_connector(connector);
            let mut res = client.get(url).send().unwrap();
            let mut result = String::new();
            res.read_to_string(&mut result);

            tx.send(result).unwrap();
        });
    }

    //let mut result: Vec<String> = vec![];
    for _ in urls {
        println!("{}", rx.recv().unwrap());
    }
}
```

But I got an error that said:

```
error[E0277]: the trait bound `std::sync::mpsc::Sender<std::string::String>: std::marker::Sync` is not satisfied
--> src/lib.rs:18:9
|
18 |     thread::spawn(|| {
|         ~~~~~ the trait `std::marker::Sync` is not implemented for `std::sync::mpsc::Sender<std::string::String>`
|
= note: `std::sync::mpsc::Sender<std::string::String>` cannot be shared between threads safely
= note: required because of the requirements on the impl of `std::marker::Send` for `&std::sync::mpsc::Sender<std::string::String>`
= note: required because it appears within the type `[closure@src/lib.rs:18:23: 29:10 url:&str, tx:&std::sync::mpsc::Sender<std::string::String>]`
= note: required by `std::thread::spawn`
```

When I tried to put the `move` in the `thread::spawn`:

```
thread::spawn(move || {
    ...
})
```

I got another error related to lifetime:

```
error[E0495]: cannot infer an appropriate lifetime due to conflicting requirements
--> src/lib.rs:15:16
|
15 |     for url in urls {
|         ~~~~
|
note: first, the lifetime cannot outlive the anonymous lifetime #1 defined on the block at 12:26...
--> src/lib.rs:12:27
|
12 | fn fetch(urls: Vec<&str>) {
|         ^
|
note: ...so that expression is assignable (expected std::vec::Vec<&str>, found std::vec::Vec<&str>)
--> src/lib.rs:15:16
|
15 |     for url in urls {
|         ~~~~
|
= note: but, the lifetime must be valid for the static lifetime...
note: ...so that the type `[closure@src/lib.rs:18:23: 27:10 url:&str, tx:&std::sync::mpsc::Sender<std::string::String>]` will meet its required lifetime bounds
--> src/lib.rs:18:9
|
18 |     thread::spawn(move || {
|         ~~~~~
```

So, what is the proper way to send strings from threads through channel here? And how can I solve the lifetime problem in the later error?

Thank you so much!

multithreading

rust

Your privacy

By clicking "Accept all cookies", you agree Stack Exchange can store cookies on your device and disclose information in accordance with our [Cookie Policy](#).

Improve this question

Accept all cookies

Follow

Customize settings

edited Feb 26 '17 at 11:51



Lukas Kalbertodt

61.1k • 18 • 189 • 248

asked Feb 26 '17 at 11:22



Huy Tran

765 • 1 • 10 • 22

Add a comment

1 Answer

Active Oldest Votes



12



Adding the `move` is the correct solution to your first problem. The second error indicates a problem in your code that was there before already, but is detected only in a later compiler stage. So what does this second error mean?

Well, a spawned thread *can* run forever (more precisely: as long as the main thread/the whole program runs). In your case they don't, because you block the calling thread waiting for the results from the channel. But the compiler doesn't know that. Therefore, `thread::spawn()` requires the passed closure to be `: 'static` which means that it doesn't reference anything that lives shorter than the whole program.

But in your case the closure has a reference to the url, a `&str`. But how long does the string behind that reference actually live? Not necessarily forever! That's the problem here.

The typical solution to problems like these is to use an `Arc` and wrap the owned value in it. But this is not possible here, because your function does not have access to the owned value. There are a few possible solutions for you:

- Use a *scoped thread API*, like [crossbeam offers](#). This API makes sure that the spawned thread doesn't outlive its parent, so you can just reference the `&str` inside of your closure. I think this is actually the best solution with the only downside of pulling in a new dependency.
- Change your function signature to `fn fetch(urls: Vec<&'static str>)`. It works, but it limits the callers of your function as they have to provide static strings. I guess that the list of URLs is not just a list of string literals, but dynamically generated; so this is not really a solution for you.
- Clone the `&str` to move the resulting `String` into the closure. This is not really a nice solution, though, as useless clones should be avoided. But it could be tolerable in your situation as the HTTP request will takes a whole lot longer than cloning a rather small (url) string.

Share

Improve this answer

Follow

answered Feb 26 '17 at 11:50



Lukas Kalbertodt

61.1k • 18 • 189 • 248

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](#) 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

[Why is 'data' still borrowed at the end of my function?](#)

[Pass a String parameter to several threads in Rust](#)

Related

[What is the difference between a process and a thread?](#)

[Difference between wait\(\) and sleep\(\)](#)

[What are the differences between Rust's 'String' and 'str'?](#)

[Why doesn't println! work in Rust unit tests?](#)

[Trying to implement core::fmt::Show](#)

["the type does not fulfill the required lifetime" when using a method in a thread](#)

[Problems with rust lifetime specifier](#)

[1
the trait '_embedded_hal_digital_InputPin' is not implemented for 'PE2<Output<OpenDrain>>'](#)

[Awaiting a Number of Futures Unknown at Compile Time](#)


Hot Network Questions

 [Sample without replacement from 1 to N and stop when the value is less than the previous one](#)

 [What does "Gracōs Argōs" in this sentence mean? \(LLpsI\)](#)

 [Seeing oneself in an abstract painting](#)

 [What would prevent Big Pharma from marketing witch potions to consumers in pill form?](#)

 [Is it acceptable to omit "about" in this sentence? "I love everything \(about\) math."](#)

[more hot questions](#)

 [Question feed](#)

STACK OVERFLOW

[Questions](#)
[Jobs](#)
[Developer Jobs Directory](#)
[Salary Calculator](#)
[Help](#)
[Mobile](#)

PRODUCTS

[Teams](#)
[Talent](#)
[Advertising](#)
[Enterprise](#)

COMPANY

[About](#)
[Press](#)
[Work Here](#)
[Legal](#)
[Privacy Policy](#)
[Terms of Service](#)
[Contact Us](#)
[Cookie Settings](#)
[Cookie Policy](#)

STACK EXCHANGE NETWORK

[Technology](#)
[Culture & recreation](#)
[Life & arts](#)
[Science](#)
[Professional](#)
[Business](#)
[API](#)
[Data](#)

[Blog](#)
[Facebook](#)
[Twitter](#)
[LinkedIn](#)
[Instagram](#)

site design / logo © 2021 Stack Exchange Inc; user contributions licensed under [cc by-sa](#). rev 2021.12.22.41046