Why do generic lifetimes not conform to the smaller lifetime of a nested scope?

Asked 3 years, 9 months ago
Active 3 years, 9 months ago
Viewed 122 times

9

According to *The Rust Programming Language*:

> Since scopes always nest, another way to say this is that the generic lifetime `'a` will get the concrete lifetime equal to the smaller of the lifetimes of `x` and `y`.

```
fn main() {
    let x = "abcd";
    let result;
    {
        let y = "qwerty";
        result = longest(x, y);
    }
    println!("The longest string is {} ", result);

}

fn longest<'a>(x: &'a str, y: &'a str) -> &'a str {
    if x.len() > y.len() {
        x
    } else {
        y
    }
}
```

In the main function, "the smaller of the lifetimes of x and y" is the nested scope. This should be the lifetime of the value in `result` as well, but the result contains the correct value from outside of that nested scope.

Why does this code work correctly?

scope rust lifetime

Share
Improve this question
Follow

edited Mar 15 '18 at 15:41

Shepmaster
305k ● 59 ● 824 ● 1083

asked Mar 15 '18 at 5:53

saga
1,525 ● 1 ● 11 ● 33

Add a comment

1 Answer

Active Oldest Votes

6

That's only true when talking about lifetimes derived from borrowing local variables. In this case, the relevant lifetime is the lifetime of the string data, which for string literals is `'static`. In other words, the `&str`s are pointing to data stored elsewhere (in the static data segment, specifically), so they don't interact with stack lifetimes at all.

If we change the example slightly, we can induce the behaviour you're expecting:

```
fn main() {
    let x = "abcd";
    let result;
    {
        let y = "qwerty";
        result = longest(&x, &y);
    }
    println!("The longest string is {}  ", result);

}

fn longest<'a>(x: &'a &'static str, y: &'a &'static str) -> &'a &'static str {
    if x.len() > y.len() {
        x
    } else {
        y
    }
}
```

Which fails to compile with:

```
error[E0597]: `y` does not live long enough
  --> src/main.rs:6:35
   |
6  |         result = longest(&x, &y);
   |                                  ^ borrowed value does not live long enough
7  |     }
   |     - `y` dropped here while still borrowed
...
10 | }
   | - borrowed value needs to live until here
```

This fails because *now* we're talking about borrows into the stack.

Share
Improve this answer
Follow

edited Mar 15 '18 at 15:43

Shepmaster
305k ● 59 ● 824 ● 1083

answered Mar 15 '18 at 6:34

DK.
47.4k ● 3 ● 150 ● 142

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 scope rust lifetime or ask your own question.

Linked

Why rust ignore lifetime checks on &str?

Related

Do lifetime annotations in Rust change the lifetime of the variables?

How to understand the lifetime of function parameters and return values in Rust?

Lifetimes in Rust when using Strings

Understanding lifetimes: max lifetime and 'static

Why rust ignore lifetime checks on &str?

1
How is output lifetime of a function calculated?

Semantics of lifetime parameters

2
A simple test case to check my understanding of rust lifetimes

Hot Network Questions

- Is the science in "Don't Look Up" realistic?
- Calculating mean follow up from ONLY sample size and range
- What is this large long-legged orange and black insect?
- If we can get people to the moon and back, why are we so adamant that it's impossible to service James Webb at 4x that with a one way robotic vehicle?
- What do I do when my boss is sabotaging interviews?

more hot questions

Question feed