Why does Rust not recognize that I re-assign to the moved variable inside a closure?

Asked 4 years ago
Active 4 years ago
Viewed 349 times

2

I am having trouble understanding why a particular pattern is not compiling.

Rust recognizes when I move a variable and then reassign to it outside of a closure and I think properly allows the code to compile, but when I try to do the same in a closure that will be run more than once it will not.

```
fn main() {
    let mut v = vec![1, 2, 3, 4];
    v.into_iter().fold(0, |a, b| a + b);
    v = vec![1, 2, 3, 4];
    vec![1, 2, 3].into_iter().for_each(|x| {
        v.into_iter().fold(x, |a, b| a + b);
        v = vec![1, 2, 3, 4];
    });
}
```

```
error[E0507]: cannot move out of captured outer variable in an `FnMut` closure
 --> src/main.rs:6:9
  |
2 |     let mut v = vec![1, 2, 3, 4];
  |         ----- captured outer variable
...
6 |         v.into_iter().fold(x, |a, b| a + b);
  |         ^ cannot move out of captured outer variable in an `FnMut` closure
```

It seems to me that the reassignment to `v` should satisfy the borrow checker that no variable will be accessed after being moved. Am I missing something?

rust borrow-checker

Share
Improve this question
Follow

edited Dec 7 '17 at 2:03
Shepmaster
305k 59 824 1083

asked Dec 7 '17 at 1:47
blyncsy-david-lewis
21 1 2

My guess: `FnMut` closures can possibly be called multiple times. In this case a `for_each` will execute in sequence, but what if that closure were being passed to multiple threads? You could consume `v` and then other thread try to access it before you can reassign to it. But this is just a wild guess - I'm a Rust newbie myself so I'm not sure.
– julioolvr
Dec 7 '17 at 2:15

@julioolvr I had that thought too, but I believe rust has traits that must be implemented for that to be so. The Send and Sync traits are what tell rust that types can be shared between threads, and for_each does not require either of those types so that should never be possible.
– blyncsy-david-lewis
Dec 7 '17 at 2:20

Add a comment

2 Answers

Active Oldest Votes

4

As @Shepmaster mentioned, the fix is to use `std::mem::replace`.

So, what is the difference between:

```
v.into_iter().fold(x, |a, b| a + b);
v = vec![1, 2, 3, 4];
```

and

```
let v_old = std::mem::replace(&mut v, vec![1, 2, 3, 4]);
v_old.into_iter().fold(x, |a, b| a + b);
```

?

In two words: **exception safety**.

If, for some reason, the expression `v.into_iter().fold(...)` would panic, it would leave `v` moved out and the next statement would never be executed.

This is perfectly acceptable in a `FnOnce`, as you will never call the closure another time, but not acceptable in a `FnMut` or `Fn` as on the next call... *what would you do with* `v`?

On the other hand, using `std::mem::replace`, you swap *first* and *then* execute the potentially panicking operation. If the operation does panic, then all that is left "moved out" is a temporary variable which disappears at the end of the stack frame anyway. No issue.

Share
Improve this answer
Follow
answered Dec 7 '17 at 7:50

Matthieu M.
261k • 40 • 396 • 665

Add a comment

2

the reassignment to `v` should satisfy the borrow checker that no variable will be accessed after being moved

Pay attention to the error message details — there isn't a move to start with:

```
cannot move out of captured outer variable in an `FnMut` closure
```

Since there was no move out, it doesn't make sense to move something back in.

Instead, you can replace the value through the mutable reference and consume the old value:

```
fn main() {
  let mut v = vec![1, 2, 3, 4];

  vec![1, 2, 3].into_iter().for_each(move |x| {
    let v_old = std::mem::replace(&mut v, vec![1, 2, 3, 4]);
    v_old.into_iter().fold(x, |a, b| a + b);
  });
}
```

Share
Improve this answer
Follow
answered Dec 7 '17 at 2:53

Shepmaster
305k • 59 • 824 • 1083

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 rust borrow-checker 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

Related

336
How do I print the type of a variable in Rust?

Passing a closure to a recursive function

6
cannot move out of borrowed content when unwrapping a member variable in a &mut self method

How to convert C variable-length array code to Rust?

Why can't I store a value and a reference to that value in the same struct?

Returning a reference from a HashMap or Vec causes a borrow to last beyond the scope it's in?

Why does the closure take ownership of the vector here?

Why does the Rust compiler not optimize code assuming that two mutable references cannot alias?

How to deal with rust borrow checker

Hot Network Questions

Could mass timber construction techniques have been used in the past?
FEM for vector valued problems: reference request
What did John the Baptist say about Jesus that resulted in many Jews in Jesus' time to put faith in him? John 10:41-42
Is there a deterministic guide to landing?
What happened to this character in MCU Spider-Man's life?
more hot questions

Question feed