



Products

# Why is there a borrow error when no borrowing overlap is occurring?

Login Sign Up

Ask Question

Asked 2 years, 8 months ago

Active 2 years, 8 months ago

Viewed 82 times



4



The following [code](#) fails with a borrow error:

```
extern crate chrono; // 0.4.6

fn main() {
    let mut now = chrono::Local::today();
    now = std::mem::replace(&mut now, now.succ());
}
```

The error is:

```
error[E0502]: cannot borrow `now` as immutable because it is also borrowed as mutable
  -> src/lib.rs:5:39
   |
5 |   now = std::mem::replace(&mut now, now.succ());
   |   ~~~~~^~~~~~ ^^^ immutable borrow occurs here
   |       |
   |       mutable borrow occurs here
   |       mutable borrow later used by call
```

Why is there a borrow error here? [now.succ\(\)](#) returns a new object, and it would look like the `succ()` call should return the new object, end the immutable borrow before the mutable borrow occurs with `replace`.

[rust](#) [borrow](#)

Share

Improve this

question

Follow

edited Apr 30 '19 at 14:30

asked Apr 30 '19 at 14:18



Listerone

1,139 ● 5 ● 15

I think order matter. But anyway, you already have the solution

– [Stargateur](#)

Apr 30 '19 at 14:28

Yes, the question is not how to fix it. It's why it's an error at all.

– [Listerone](#)

Apr 30 '19 at 14:31

[Add a comment](#)

1 Answer

[Active](#) [Oldest](#) [Votes](#)



3



The order of the arguments matters. For example this works:

```
/// Same as 'std::mem::replace', but with the reversed parameter order.
pub fn replace<T>(src: T, dest: &mut T) -> T {
    std::mem::replace(dest, src)
}
```

**Your privacy**

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

now = replace(now.succ(), &mut now);

Accept all cookies Customize settings

[\(link to playground\)](#)

But in your example, `&mut now` appears first, and when evaluating the second parameter, it is already borrowed.

[Share](#)

[Improve this answer](#)

[Follow](#)

edited Apr 30 '19 at 14:40



[Stargateur](#)

19.1k ● 7 ● 49 ● 74

answered Apr 30 '19 at 14:38



[mcarton](#)

22.7k ● 5 ● 63 ● 75

Why `now = replace(now.succ(), &mut now);` ? look strange xd it's a do nothing line

– [Stargateur](#)

Apr 30 '19 at 14:42

That would be a compiler limitation then.

– [Listerone](#)

Apr 30 '19 at 14:46

1

@Stargateur it never even crossed my mind to make that example function work. I was even going to type `unimplemented()` but got too lazy to do even this.

– [mcarton](#)

Apr 30 '19 at 14:55

@Listerone while the order wouldn't actually matter for borrowing in this example, for more complex examples with different lifetimes and references, it could matter, so the compiler must at some point set a line.

– [mcarton](#)

Apr 30 '19 at 14:56

@mcarton Is that really true? Since the `immutable borrow` is a call that must return before `replace` can actually receive the mutable reference to `now`, is there *any* scenario where these two could interact other than as expected?

– [jspencer](#)

Aug 8 '19 at 6:33

[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](#) 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 indexing a mutable vector based on its `len\(\)` considered simultaneous borrowing?](#)

Related

["the immutable borrow prevents mutable borrows" when pumping events with `rust-sdl2`](#)

[Borrow errors for multiple borrows](#)

Why does a call to 'fn pop(&mut self) -> Result<T, &str>' continue to borrow my data structure?

Rust 'Vec' - cannot borrow 'Vec' as immutable inside 'impl' method (error[E0502])






Why I cannot borrow something I borrowed and why I can move a method's self?

1  
Prevent cannot borrow '\*self' as immutable because it is also borrowed as mutable when accessing disjoint fields in struct?

Eronous mutable borrow (E0502) when trying to remove and insert into a HashMap

Borrow checker error when overwriting variable with mutable reference

## Hot Network Questions

-  How can I let my opponent know that I'm touching a piece to adjust it?
  -  Is it possible to convert a taproot address into a native segwit address?
  -  What was the plutonium for, that was stolen at the start of The Amazing Spider-Man 2?
  -  Alternatives to replace tandem single pole MWBC breakers?
  -  Question on connections in general relativity and particle physics
- [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