How to swap two fields of a struct

Asked 5 years, 10 months ago
Active 5 years, 10 months ago
Viewed 693 times

5

I've started studying Rust and I'm trying to implement a simple 1D cellular automata. I want to represent the automata state (`Board`) as struct holding the size and two different vectors (of same size). I tried:

```
struct Board {
    n: usize,
    cur:  Vec<u32>,
    next: Vec<u32>,
}

impl Board {
    fn new(size: usize) -> Board {
        Board {
            n: size,
            cur: vec![0;size],
            next: vec![0;size],
        }
    }
}
```

So far so good. I'm also able of mutating both vectors. But then I want to be able to swap both vectors (or rather their references), such as:

```
fn swap(&mut self) -> &Board {
    let tmp = self.cur;
    self.cur = self.next;
    self.next = tmp;
    self
}
```

It fails, with an `cannot move out of borrowed content [E0507]` which I think I can understand. I also tried `mem::swap` which I found in a similarly title question without success.

How can I make this example work ? (Since I'm a total beginner with Rust, don't hesitate to suggest a different data representation).

struct reference rust swap

Share
Improve this question
Follow

edited Feb 26 '16 at 14:39
Shepmaster
305k ● 59 ● 824 ● 1083

asked Feb 26 '16 at 11:00
paradigmatic
39.3k ● 18 ● 86 ● 145

Add a comment

2 Answers

Active Oldest Votes

5

> What's the problem?

You are **punching holes** in your data:

```
fn swap(&mut self) -> &Board {
    let tmp = self.cur;         // 1
    self.cur = self.next;      // 2
    self.next = tmp;          // 3
    self
}
```

If we analyze line by line:

1. `self.cur` is now uninitialized
2. `self.next` is now uninitialized
3. everything is kosher again

If for some reason computation is interrupted before line (3) has a change to tighten up the situation, `self` is now poisoned and may cause all kinds of nasty things to happen. Notably, its destructor could attempt to free memory twice for example.

In theory, you could have the compiler check for temporary holes and prove, without doubt, that:

- no function accesses `self` while a hole is punched
- by the end of the scope, whether it is reached normally or by unwinding, the holes are filled up again

and indeed at some point it was considered... but the truth is that it's complicated and there are readily available work-arounds.

So?

The answer lies in `std::mem` which exposes functions to perform such low-level operations in a safe manner. While the functions themselves are implemented using `unsafe` under the hood, they lean on their understanding of the language and runtime to expose safe interfaces.

The two particular functions that will interest you are:

- `replace` : replaces the content of `dest: &mut T` by `src: T` and returns the what was previously contained behind `dest`
- `swap` : exchanges the content of its arguments

With those two simple and safe primitives, you can avoid punching holes in your data.

Share
Improve this answer
Follow
edited Feb 26 '16 at 14:39

Shepmaster
305k ● 59 ● 824 ● 1083

answered Feb 26 '16 at 12:52

Matthieu M.
261k ● 40 ● 396 ● 665

Add a comment

4

✓

As you noticed, `mem::swap` is the way to go:

```
fn swap(&mut self) -> &Board {
    std::mem::swap(&mut self.cur, &mut self.next);
    self
}
```

This works. Note that when using `.` you are dereferencing `self`. So while `self` has the type `&mut Board`, `self.cur` has the type `Vec<u32>`. Therefore the compiler complained about "move out of borrowed content" and we need the additional `&mut` s.

Share
Improve this answer
Follow
answered Feb 26 '16 at 11:08

Lukas Kalbertodt
61.1k ● 18 ● 189 ● 248

Add a comment

Your Answer

Post Your Answer

Not the answer you're looking for? Browse other questions tagged struct reference rust swap 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

What's the difference between struct and class in .NET?

When should you use a class vs a struct in C++?

1523
When should I use a struct rather than a class in C#?

Difference between 'struct' and 'typedef struct' in C++?

How do I pass a variable by reference?

1174
How to copy a dictionary and only edit the copy

List changes unexpectedly after assignment. Why is this and how can I prevent it?

Is there a standardized method to swap two variables in Python?

How to print struct variables in console?

Hot Network Questions

Who (or what) created the atropal?

What does the numbers mean in this guitar tab if they already gave the chords to play?

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?

Have there been no deaths due to omicron in Africa?

What's the largest REG_SZ value that Regedit can edit?

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