Dereferencing boxed struct and moving its field causes it to be moved

Ask Question

Asked 9 years, 1 month ago

Modified 6 years, 4 months ago

Viewed 892 times

2

Dereferencing a boxed struct and moving its field causes it to be moved, but doing it in another way works just fine. I don't understand the difference between these two pop functions. How does one fail when the other one doesn't?

```
pub struct Stack<T> {
   head: Option<Box<Node<T>>>,
   len: usize,
}
struct Node<T> {
   element: T,
   next: Option<Box<Node<T>>>,
}

impl<T> Stack<T> {
   pub fn pop(&mut self) -> Option<T> {
       self.head.take().map(|boxed_node| {
           let node = *boxed_node;
           self.head = node.next;
           node.element
       })
   }

   pub fn pop_causes_error(&mut self) -> Option<T> {
       self.head.take().map(|boxed_node| {
           self.head = (*boxed_node).next;
           (*boxed_node).element
       })
   }
}
```

```
error[E0382]: use of moved value: `boxed_node`
 --> src/main.rs:22:13
  |
21 |             self.head = (*boxed_node).next;
  |                         ------------------ value moved here
22 |             (*boxed_node).element
  |             ^^^^^^^^^^^^^^^^^^^^^ value used here after move
  |
  = note: move occurs because `boxed_node.next` has type `std::option::Option<std::boxed::Box<Node<T>>>`, which does not imple
```

- rust

Share

Improve this question

Follow

edited Jul 31, 2018 at 15:55

Shepmaster

439k116116 gold badges1.3k1.3k silver badges1.5k1.5k bronze badges

asked Oct 29, 2016 at 12:57

Dulguun Otgon

1,96811 gold badge2121 silver badges4242 bronze badges

Add a comment |

2 Answers

Sorted by: Reset to default

Highest score (default) Trending (recent votes count more) Date modified (newest first) Date created (oldest first)

5

You can only move out of a box once:

```
struct S;
```

```
fn main() {
   let x = Box::new(S);
   let val: S = *x;
   let val2: S = *x; // <-- use of moved value: `*x`
}
```

In the first function, you moved the value out of the box and assigned it to the `node` variable. This allows you to move different fields out of it. Even if one field is moved, other fields are still available. Equivalent to this:

```
struct S1 {
   a: S2,
   b: S2,
}
struct S2;

fn main() {
   let x = Box::new(S1 { a: S2, b: S2 });

   let tmp: S1 = *x;
   let a = tmp.a;
   let b = tmp.b;
}
```

In the second function, you move the value to the temporary (`*boxed_node`) and then move a field out of it. The temporary value is destroyed immediately after the end of the expression, along with its other fields. The box doesn't have the data anymore, and you don't have a variable to take the other field from. Equivalent to this:

```
struct S1 {
   a: S2,
   b: S2,
}
struct S2;

fn main() {
   let x = Box::new(S1 { a: S2, b: S2 });

   let tmp: S1 = *x;
   let a = tmp.a;

   let tmp: S1 = *x; // <-- use of moved value: `*x`
   let b = tmp.b;
}
```

Share

Improve this answer

Follow

edited Jul 31, 2018 at 17:05

Shepmaster

439k116116 gold badges1.3k1.3k silver badges1.5k1.5k bronze badges

answered Oct 29, 2016 at 13:06

Pavel Strakhov

40.8k66 gold badges9393 silver badges131131 bronze badges

Sign up to request clarification or add additional context in comments.

Comments

Add a comment

1

Some good news is that non-lexical lifetimes will allow your original code to work:

```
pub struct Stack<T> {
   head: Option<Box<Node<T>>>,
   len: usize,
}

struct Node<T> {
   element: T,
```

```
    next: Option<Box<Node<T>>>,
}

impl<T> Stack<T> {
    pub fn pop_no_longer_causes_error(&mut self) -> Option<T> {
        self.head.take().map(|boxed_node| {
            self.head = (*boxed_node).next;
            (*boxed_node).element
        })
    }
}
```

NLL enhances the borrow checker to better track the moves of variables.

Share

Improve this answer

Follow

edited Jul 22, 2019 at 22:57

answered Aug 23, 2018 at 23:21

Shepmaster

439k116116 gold badges1.3k1.3k silver badges1.5k1.5k bronze badges

Comments

Add a comment

Your Answer

Thanks for contributing an answer to Stack Overflow!

- Please be sure to *answer the question*. Provide details and share your research!

But *avoid* …

- Asking for help, clarification, or responding to other answers.
- Making statements based on opinion; back them up with references or personal experience.

To learn more, see our tips on writing great answers.

Sign up or log in

Sign up using Google

Sign up using Email and Password

Submit

Post as a guest

Name

Email

Required, but never shown

Post as a guest

Name

Email

Required, but never shown

Post Your Answer Discard

By clicking “Post Your Answer”, you agree to our terms of service and acknowledge you have read our privacy policy.

Start asking to get answers

Find the answer to your question by asking.

Ask question

Explore related questions

- rust

See similar questions with these tags.

- The Overflow Blog
  - Introducing Stack Overflow AI Assist—a tool for the modern developer
  - Treating your agents like microservices
- Featured on Meta
  - Chat room owners can now establish room guidelines
  - AI Assist is now available on Stack Overflow
  - Policy: Generative AI (e.g., ChatGPT) is banned

Linked

1
How does boxing affect borrow checking?
162
What are non-lexical lifetimes?

Related

0
Move a Box inside a Struct
7
How to bind multiple fields of a boxed struct without getting "use moved value" error?
2
Maybe move field inside struct method
5
Confused by move semantics of struct fields inside a Box
0
How can I pass a boxed member of a borrowed struct to a function without copying it?
2
How can I write a boxed closure which mutates a reference to a struct?
3
Putting thing (moving) inside struct gives lifetime error
2
Why does passing a struct to a function in Rust permanently move it to that function?
1
Rust why is the address of struct not the same address that is being dropped?
0
How can I change the value of a field of a boxed struct while making sure the pointer to the box stays valid?

Hot Network Questions

- Why does clang zero "eax" before calling a function with unspecified parameters, but gcc doesn't?
- Critiques of the real vs. nominal definition distinction

- Can enemies damage each other in Breath of the Wild?
- "Us" in Hegel's "Greater Logic"
- Preservation of universal sentences under substructures in higher-order logic
- What international legal mechanisms are in place to stop a country from outright expropriating foreign assets?
- Unity HDRP custom pass stencil buffer not working in build mode
- How to reduce GAM sensitivity to sparse data
- Statistical testing for bimodal/multimodal sample
- Swords and sandals movie where the ancient Greek hero fights in space
- Why does black allow white to have a protected passed central pawn here?
- "Save as webpage, HTML Only" Fail while offiline, Firefox
- is it possible to keep the space when using url in latex
- A very weird integral equation
- Is "We will review application starting from..." a deadline?
- Download is performed unsandboxed as root - Why am I getting this message, and how do I fix it?
- PNP BJT transistor for switching and sourcing to IC
- Obscure method of supression of transient voltage spike in CLC-circuit
- Diagonal Arrows Under an Equation
- How to check if two tables are identical?
- When did "doubt" meaning question or point of uncertainty fall out of use outside of Indian English?
- When the fuel governor fails, why is autorotation the only option for landing?
- I2C bus problems when using two distance sensors in Arduino
- Is the phrase "murder will out" from Chaucer?

more hot questions

Question feed

Subscribe to RSS

Question feed

To subscribe to this RSS feed, copy and paste this URL into your RSS reader.

- Professional
- Business
- API
- Data
- Blog
- Facebook
- Twitter
- LinkedIn
- Instagram