Skip to main content

Stack Overflow

1. About
2. Products
3. For Teams

1. Stack Internal Implement a knowledge platform layer to power your enterprise and AI tools.
2. Stack Data Licensing Get access to top-class technical expertise with trusted & attributed content.
3. Stack Ads Connect your brand to the world’s most trusted technologist communities.
4. Releases Keep up-to-date on features we add to Stack Overflow and Stack Internal.
5. About the company Visit the blog

Loading…

current community

Stack Overflow

help chat

- Meta Stack Overflow

your communities

Sign up or log in to customize your list.

more stack exchange communities

company blog

3. Log in
4. Sign up

AI Assist is now on Stack Overflow. Start a chat to get instant answers from across the network. Sign up to save and share your chats.

Home

Questions

AI Assist

Tags

Challenges

7. Chat

Articles

Users

Companies

Collectives

Communities for your favorite technologies. Explore all Collectives

Stack Internal

Stack Overflow for Teams is now called **Stack Internal**. Bring the best of human thought and AI automation together at your work.

Try for free Learn more

Stack Internal

Bring the best of human thought and AI automation together at your work. Learn more

How does automatic de-referencing work in Rust? [duplicate]

Ask Question

Asked 2 years, 6 months ago

Modified 2 years, 6 months ago

Viewed 706 times

1

This question already has answers here:

What are Rust's exact auto-dereferencing rules? (4 answers)

Closed 2 years ago.

I am trying to learn Rust and am having some trouble understanding why we need to dereference pointers in some cases and not in others. Fishing for answers online, I found out that Rust has automatic dereferencing, but I couldn't find much on the topic. Can someone explain to me how this exactly works?

To make it more specific, here is a sample code where I encountered this problem.

```
fn mode(v: &mut Vec<i32>) -> Option<i32> {
   let mut map: HashMap<i32, i32> = HashMap::new();
   let mut max = (0, 0);
   if v.is_empty() {
       return None;

   } else {
       ();

   }
   for i in v {
       let count = map.entry(*i).or_insert(0);
       *count += 1;

   }
   for (k, v) in map.iter() {
       if v > &max.1 {
           max = (*k, *v);
          } else {
           ();

       }

   }
   Some(max.0)
```

Here, there is apparently no need to dereference vector v anywhere, but we need to dereference several other variables such as i in line 75, and I do not understand why.

- rust

Share

Improve this question

Follow

edited May 28, 2023 at 4:58

asked May 28, 2023 at 4:48

Keshav Gulati

2144 bronze badges

3

`is_empty(&self)` needs a reference on self. v is already a reference (`&mut Vec`), so there is no need to dereference. `i in v` calls `into_iter(self) where self is &Vec` , so there is also no need to derefence. Assigning to `(i32,i32)` needs dereferencing.

CoronA – CoronA

2023-05-28 05:11:03 +00:00

Commented May 28, 2023 at 5:11

2

@CoronA It does need to dereference, though, because the `is_empty` method comes from the slice type, not `Vec` itself. It's usable because `Vec<T>` implements `Deref<Target = [T]>`. If the compiler didn't auto-deref, you'd have to manually deref the `Vec` to get a slice.

cdhowie – cdhowie

2023-05-28 05:27:44 +00:00

Commented May 28, 2023 at 5:27

@cdhowie: I voted up your comment, because I did not check it. But after checking it I cannot find anything wrong with my statement: `is_empty(&self)` is implemented for `Vec<T>` and pasting the code into my IDE also claims that this method is called.

CoronA – CoronA

2023-05-28 19:18:16 +00:00

Commented May 28, 2023 at 19:18

Add a comment |

1 Answer

Sorted by: Reset to default

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

5

The easiest way to illustrate this is within the context of the sample code `x.foo(y)`.

`x` here is the *receiver.* Given the type of `x`, the compiler has to figure out what function `foo` is. It attempts to locate `foo` on `x`'s type, and additionally by *auto-dereferencing* `x`, which is what you are asking about. This process considers the type of `x` as well as every type that is encountered by dereferencing `x` as many times as possible. (It also considers `&` and possibly `&mut` variants of these types, depending on the mutability of `x` or its reference type.)

If exactly one `foo` is found, then the call is transformed to `T::foo(z, y)` where `T` is the type the `foo` method was located on, and `z` is whatever sequence of dereferences (plus one final, optional `&` or `&mut`) results in a `T`, `&T`, or `&mut T`, as required by `T::foo`.

What about `y`, though? Arguments undergo *coercion*, which is a different thing altogether. The linked documentation lists all of the possible coercions, which I will omit for the sake of brevity, but notably absent are `&T` to `T` and `&mut T` to `T`, one of which would be required to make `map.entry(i)` work in your sample code. The explicit dereference is therefore required.

Note that one of the items present in the list of coercions is:

> `&T` or `&mut T` to `&U` if `T` implements `Deref<Target = U>`.

Observe that this *doesn't actually dereference a reference* but rather *converts the reference to another type of reference!* That is, the coercion is `&T` to `&U`, *not* `&T` to `U`. This is what allows you to give `&String` to an argument that expects a `&str`, for example (`String` implements `Deref<Target = str>`), but this rule doesn't provide coercion from `&mut i32` to `i32` as would be required to make `map.entry(i)` work.

`for i in v` is a completely different scenario; this works simply because `&mut Vec<T>` implements `IntoIterator`.

So the answer as to why you don't have to dereference `v` is because you are using it in receiver position, where auto-dereference happens. Function arguments instead undergo coercion, which has an entirely different set of rules.

Now having said that, in your code `v` is already a reference to a `Vec`, and so `v.is_empty()` could be explained through coercion as well, as this method comes from slices, to which `Vec` automatically dereferences. However, the overall point still stands -- you cannot conflate auto-dereferencing with deref coercion. This is one instance where they would happen to do the same thing, but in many other cases they would not.

Share

Improve this answer

Follow

edited May 28, 2023 at 5:34

answered May 28, 2023 at 5:18

cdhowie

172k2525 gold badges303303 silver badges324324 bronze badges

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

3 Comments

Add a comment

TaekYoung

TaekYoung Over a year ago

Isn't it dangerous to expand name space like that? Because what if Reference and Receiver have same method?

2024-04-30T19:04:44.563Z+00:00

0

Reply

- Copy link

cdhowie

cdhowie Over a year ago

@TaekYoung This is exactly why many built-in smart pointer "methods" require associated function syntax and can't be used with method-call syntax. For example, look at `Arc::downgrade`. Note that it takes `this: &Arc` instead of `&self` -- this means you have to call this function using the syntax

`Arc::downgrade(&foo)`. `foo.downgrade()` will not work. This is intentional so that if the `T` inside of the `Arc` has a `downgrade` method of its own, the method-call syntax will call T's method instead of `Arc`'s.

2024-04-30T19:42:44.37Z+00:00

1

Reply

- Copy link

TaekYoung

TaekYoung Over a year ago

Thank you! That explains a lot for me. I think rust book should contain your explanation also.

2024-05-02T09:43:31.22Z+00:00

0

Reply

- Copy link

Add a comment

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

357

What are Rust's exact auto-dereferencing rules?

Related

676

Why doesn't println! work in Rust unit tests?

509

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

408

How do I split a string in Rust?

560

How do you disable dead code warnings at the crate level in Rust?

390

Why are Rust executables so huge?

422

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

3

Rust "Borrowed pointers" syntax

3

Rust lifetime for database connection bundle

302

How can a Rust program access metadata from its Cargo package?

648

How do I concatenate strings?

Hot Network Questions

- Misunderstanding the lorentz transformation
- Explain this seeming contradiction in Euclid Book 1 Proposition 16
- How does supporting "One China" work without also supporting Taiwan-China reunification?
- "Creatures your opponents control" after the ability is activated
- How to prevent a batch file call from escaping a caret character?
- Curve has weird bumps when triangulated
- Would it be plausible to have a flat eye with a moving pupil?
- Why are left and right cosets different in the non-normal case?
- Generating Irregular, Separated Polygonal Cells along a Curve in Geometry Nodes
- EEPROM Decoupling Capacitor
- What may cause any version of Microsoft Outlook to run slowly?
- Is zero-point energy a mathematical artifact?
- A very weird integral equation
- What is the difference between an ultraviolet limit and ultraviolet stability?
- variable name change causes segfault
- Does short range teleporting end floating disk?
- "Us" in Hegel's "Greater Logic"
- Can a 10*10 square be paved with 1*4 rectangular stone plates?
- Critiques of the real vs. nominal definition distinction
- Is there any way to restore hearts without consuming anything, without hot springs, and without shrines?
- How to add blur inside subtracted element in Figma?
- Could a world tree have different biomes?
- What's the point of swapping polarity between mod slots?
- Why is "cmd | xargs" to join lines of cmd with spaces wrong

more hot questions

Stack Overflow

- Questions
- Help
- Chat

Business

- Stack Internal
- Stack Data Licensing
- Stack Ads

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