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

Collectives™ on Stack Overflow

Find centralized, trusted content and collaborate around the technologies you use most.

Learn more about Collectives

Stack Internal

Knowledge at work

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

Explore Stack Internal

How to define a variable at the declaration level and be able to modify it? [duplicate]

Ask Question

Asked 4 years, 3 months ago

Modified 4 years, 3 months ago

Viewed 535 times

1

This question already has answers here:

How do I create a global, mutable singleton? (9 answers)

Closed 4 years ago.

To summarize, I post a simple C++ example that illustrates my question. I want to write the equivalent in Rust. I have read the documentation, but I cannot find the precise answer to this.

```
#include <iostream>
using namespace std;

int _i;

void firstFunction() {
   _i++;
}

void secondFunction() {
   _i++;
}

int main()
{
   firstFunction();
   secondFunction();
   cout << "Value of _i is: " << _i; // 2 (yes!)
}
```

My attempt at Rust

```
let mut _i:i32 = 0; // ?

fn firstFunction() {
   _i = _i + 1;
}

fn secondFunction() {
   _i = _i + 1;
}

fn main() {
   firstFunction();
   secondFunction();
   println!("Value of _i is {}", _i); // must be 2
}
```

Answering my question, after the knowledge acquired by your excellent answers, I have written this approach, which seems closer to what I wanted to find,

```
struct Something {
   counter: i32,
}

impl Something {
   fn first_function(&mut self) {
       self.counter = self.counter + 1;
   }
   fn second_function(&mut self) {
       self.counter = self.counter + 1;
   }
}

fn main() {
   println!("Learning");

   let mut _t = Something { counter: 0 };
```

```
    _t.first_function();
    _t.second_function();

    println!("t: {}", _t.counter);
}
```

- rust

Share

Improve this question

Follow

edited Aug 23, 2021 at 16:43

asked Aug 22, 2021 at 18:46

Alexandra Danith Ansley

33822 silver badges1515 bronze badges

5

Your intent is not clear. You can create a struct that contains that variable and then f1 and f2 would be methods of that structure. Or you can make a local variabl ein main and then pass references to f1 and f2. Or you can make a global static variable, wrapped in a Mutex for access control. There are a lot of options, but it depends on what you want to achieve.

Svetlin Zarev – Svetlin Zarev

2021-08-22 19:15:21 +00:00

Commented Aug 22, 2021 at 19:15

In rust your code would be considered unsafe (in general across all platforms that rust could compile to), so you won't be able to do it as simply as you want. There are some various idiomatic ways with the simplest for integer being use atomics, but this question gives multiple different ways to approach: stackoverflow.com/questions/27791532/…

Jere – Jere

2021-08-22 21:03:05 +00:00

Commented Aug 22, 2021 at 21:03

This approach "How do I create a global, mutable singleton?", it is very interesting. My C++ example is more for understanding than for a solution. Certainly in all languages the matter is simple, and I have learned a lot from your comments. TXS

Alexandra Danith Ansley – Alexandra Danith Ansley

2021-08-22 22:11:24 +00:00

Commented Aug 22, 2021 at 22:11

It is a possible solution, but it takes it to such a different level that it moves away from the intention of the original question, looking for an intuitive approach between the two languages.

Alexandra Danith Ansley – Alexandra Danith Ansley

2021-08-23 16:16:43 +00:00

Commented Aug 23, 2021 at 16:16

Why did you mark your question as a duplicate? I do not think

Sith2021 – Sith2021

2021-08-23 16:46:15 +00:00

Commented Aug 23, 2021 at 16:46

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)

3

In general, global variables should be avoided (not only in Rust), and Rust makes their usage very difficult (on purpose, think about mutable borrows issues!). However, this page illustrates some ways to play around.

The intent of your C++ example is not clearly exposed, but you can achieve something similar with the example below:

- the global variable is made local to `main()`,
- the functions are converted to closures in order to capture this variable,
- closure definitions and invocations are reorganised in order to prevent from multiple simultaneous mutable borrows of the variable.

```
fn main() {
    let mut _i: i32 = 0;
    let mut first_function = || {
        _i = _i + 1;
    };
    first_function();
```

```
   let mut second_function = || {
       _i = _i + 1;
   };
   second_function();
   println!("Value of _i is {}", _i); // must be 2
}
```

---

edit: this ugly solution insists on using a global variable in order to be closer to the original C++ code; **this is absolutely not an example to be followed**.

```
static mut _I: i32 = 0;

fn get_i() -> i32 {
   unsafe { _I }
}

fn set_i(i: i32) {
   unsafe { _I = i };
}

fn first_function() {
   set_i(get_i() + 1);
}

fn second_function() {
   set_i(get_i() + 1);
}

fn main() {
   first_function();
   second_function();
   println!("Value of _i is {}", get_i()); // must be 2
}
```

---

edit: as stated in the remark before the previous example and in the comments, the previous example is close to the C++ code but is totally incorrect regarding Rust expectations (no synchronisation if we were in a multithreaded context). A step towards a more Rust-correct solution, but away from the original C++ example, would be the following one. However, it is difficult to know what kind of answer is expected here: something that looks like C++ but that does not conform to Rust expectations, or something that would eventually be acceptable in Rust even if it does not look like the original C++ code...

```
// in Cargo.toml
//   [dependencies]
//   lazy_static = "1.4.0"
#[macro_use]
extern crate lazy_static;

use std::sync::Mutex;

lazy_static! {
   static ref _I: Mutex<i32> = Mutex::new(0);
}

fn first_function() {
   let mut i = _I.lock().unwrap();
   *i = *i + 1;
}

fn second_function() {
   let mut i = _I.lock().unwrap();
   *i = *i + 1;
}

fn main() {
   first_function();
   second_function();
   println!("Value of _i is {}", _I.lock().unwrap()); // must be 2
}
```

Share

Improve this answer

Follow

edited Aug 23, 2021 at 9:28

answered Aug 22, 2021 at 19:11

prog-fh

17.3k11 gold badge2121 silver badges3535 bronze badges

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

4 Comments

Add a comment

Alexandra Danith Ansley

Alexandra Danith Ansley Over a year ago

The purpose of the simple example is to illustrate the flow. The code that you suggest to me has a quite heterodox order and flow with what I thought. Hopefully there is a more similar way :_)

2021-08-22T19:34:29.137Z+00:00

0

Reply

- Copy link

Alexandra Danith Ansley

Alexandra Danith Ansley Over a year ago

The second approach is closer. In fact I learned several things.

2021-08-22T22:04:37.773Z+00:00

0

Reply

- Copy link

user4815162342

user4815162342 Over a year ago

In the second example all the functions (except `main`) should be marked `unsafe` because, as written, they don't provide a safe abstraction over the underlying unsafe functionality. Their unsafe contract is that one is not allowed to call them from multiple threads without synchronization.

2021-08-23T07:02:27.027Z+00:00

0

Reply

- Copy link

user4815162342

user4815162342 Over a year ago

Re edit: Making the *functions* in the second example as unsafe would make it totally conform to Rust's safety expectations, though it would still not be idiomatic Rust. As written, it's just unsound.

2021-08-23T16:05:48.37Z+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.

Linked

376
How do I create a global, mutable singleton?

Related

1087
What are the differences between Rust's `String` and `str`?
560
How do you disable dead code warnings at the crate level in Rust?
509
How do I print the type of a variable in Rust?
530
What is the difference between iter and into_iter?
298
What is the difference between Copy and Clone?
458
Why can't I store a value and a reference to that value in the same struct?
353
What's the de-facto way of reading and writing files in Rust 1.x?
216
What's the difference between use and extern crate?
126
How to define test-only dependencies?

Hot Network Questions

- Is it possible to use LaTeX to create RPG-related PDFs, such as character sheets or adventure modules?
- How do I add a text editor to my (debian-based) initramfs?
- When should I use tied notes vs. the whole note duration?
- Would it be plausible to have a flat eye with a moving pupil?
- EEPROM Decoupling Capacitor
- How many alien worlds could exist within 100 ly without us noticing?
- Why are left and right cosets different in the non-normal case?
- Generating Irregular, Separated Polygonal Cells along a Curve in Geometry Nodes
- How can I determine whether a complex rational equation has real solutions?
- Best Welding Method to Fix a Damaged Steel Surly Frame
- Ping sweep of IP subnets
- is it possible to keep the space when using url in latex
- The state of icy worlds during differentiation and accretion, and how they survive with their water
- How to check if two tables are identical?
- I2C bus problems when using two distance sensors in Arduino
- How do we define time-ordering operations in QFT?
- How to use Bézier curves in `TikZ` to draw a quotation mark?
- How to reduce GAM sensitivity to sparse data
- Storm Sphere and Obstruction
- Why doesn’t sleep mode (S3) turn on in the BIOS on my Lecoo Pro 14 laptop (N155A, Lenovo sub-brand) with Windows 11?
- "Imagine if..." vs. "Imagine..."
- I can't add a loop cut on one face
- Statistical testing for bimodal/multimodal sample
- Vertical alignment of equations inside tabularx

more hot questions

Stack Overflow

- Questions
- Help
- Chat

Business

- Stack Internal