How can I expose a safe wrapper around an owned pointer?

Ask Question

Asked 10 years, 3 months ago

Modified 10 years, 3 months ago

Viewed 148 times

4

I'm wrapping a C library that has two structs: one has a pointer to the other.

```
struct StructA {
   void * some_mem;
};

struct StructB {
   void * some_mem;
   struct StructA * some_struct;
};
```

Both of these structs own memory, so my wrapper has constructors and destructors for both of them.

```
struct StructA(*mut c_void);

impl StructA {
   fn new() -> Self {
       StructA(c_constructor())
   }
}

impl Drop for StructA {
   fn drop(&mut self) {
       let StructA(ptr) = self;
       c_destructor(ptr);
   }
}
```

There's also a function that takes a pointer to `StructB` and returns its pointer to `StructA`:

```
const struct StructA * get_struct(const struct StructB * obj);
```

The user of this function should not free the returned pointer, since it will be freed when the user frees `obj`.

How can I wrap this function? The problem is that the destructor for `StructB` frees all its memory, including the one for `StructA`. So if my wrapping of `get_struct` returns an object, then the wrapped `StructA` will be freed twice (right?). It could instead return a reference to an object, but where would that object live?

I could have separate structs for `StructA` based on whether it's standalone and needs to be freed or if it's a reference, but I'm hoping that's unnecessary.

- rust

Share

Improve this question

Follow

asked Aug 28, 2015 at 21:40

awelkie

2,69222 gold badges2727 silver badges3333 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)

4

> I could have separate structs for StructA based on whether it's standalone and needs to be freed or if it's a reference, but I'm hoping that's unnecessary.

It's necessary. The difference between an owned `StructA *` and a borrowed `StructA *` is precisely the same as the difference between a `Box<T>` and a `&T`. They're both "just a pointer", but the semantics are completely different.

Something along these lines is probably what you want:

```
use std::marker::PhantomData;
struct OwnedA(*mut c_void);
impl Drop for OwnedA {
   fn drop(&mut self) { }
}
impl OwnedA {
   fn deref(&self) -> RefA { RefA(self.0, PhantomData) }
}
struct RefA<'a>(*mut c_void, PhantomData<&'a u8>);

struct OwnedB(*mut c_void);

impl Drop for OwnedB {
   fn drop(&mut self) { }
}

impl OwnedB {
   fn get_a(&self) -> RefA { RefA(get_struct(self.0), PhantomData) }
}
```

In particular, it's worth noting that lifetime parameter on RefA lets the compiler make sure you don't use a `RefA` after the backing structure has been freed.

Share

Improve this answer

Follow

answered Aug 28, 2015 at 23:39

Eli Friedman

2,39311 gold badge1515 silver badges1111 bronze badges

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

Comments

Add a comment

2

> I could have separate structs for StructA based on whether it's standalone and needs to be freed or if it's a reference, but I'm hoping that's unnecessary.

I believe this would be the accepted pattern. For backup, I'd point to the fact that this is a normal pattern in the Rust library. `&str` and `String`, `&[T]` and `Vec<T>`, `Path` and `PathBuf`, and probably lots of others I can't think of.

The good news is that you can use similar patterns as these pairs, leveraging `Deref` or `DerefMut` to call down to shared implementation:

```
use std::ops::{Deref, DerefMut};

enum RawFoo {}

fn c_foo_new() -> *const RawFoo { std::ptr::null() }
fn c_foo_free(_f: *const RawFoo) {}
fn c_foo_count(_f: *const RawFoo) -> u8 { 42 }
fn c_foo_make_awesome(_f: *const RawFoo, _v: bool) { }

struct OwnedFoo(Foo);

impl OwnedFoo {
   fn new() -> OwnedFoo {
       OwnedFoo(Foo(c_foo_new()))
   }
}

impl Drop for OwnedFoo {
   fn drop(&mut self) { c_foo_free((self.0).0) }
}

impl Deref for OwnedFoo {
   type Target = Foo;
   fn deref(&self) -> &Self::Target { &self.0 }
}
```

```
impl DerefMut for OwnedFoo {
   fn deref_mut(&mut self) -> &mut Self::Target { &mut self.0 }
}

struct Foo(*const RawFoo);

impl Foo {
   fn count(&self) -> u8 { c_foo_count(self.0) }
   fn make_awesome(&mut self, v: bool) { c_foo_make_awesome(self.0, v) }
}

fn main() {
   let mut f = OwnedFoo::new();
   println!("{}", f.count());
   f.make_awesome(true);
}
```

Then, when you get a borrowed pointer from your other object, just wrap it up in a `&Foo`:

```
use std::mem;

fn c_bar_foo_ref() -> *const RawFoo { std::ptr::null() }

// Ignoring boilerplate for wrapping the raw Bar pointer
struct Bar;

impl Bar {
   fn new() -> Bar { Bar }

   fn foo(&self) -> &Foo {
       unsafe { mem::transmute(c_bar_foo_ref()) }
   }

   fn foo_mut(&mut self) -> &mut Foo {
       unsafe { mem::transmute(c_bar_foo_ref()) }
   }
}

fn main() {
   let mut b = Bar::new();
   println!("{}", b.foo().count());
   b.foo_mut().make_awesome(true);

   // Doesn't work - lifetime constrained to Bar
   // let nope = Bar::new().foo();
}
```

Share

Improve this answer

Follow

edited Aug 29, 2015 at 13:44

answered Aug 28, 2015 at 23:48

Shepmaster

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

1 Comment

Add a comment

dcoles

dcoles Jan 20 at 6:15

How can you transmute a `*const RawFoo` to a `&Foo` where `Foo` is `struct Foo(*const RawFoo)`? Isn't that like trying to transmute a `*const RawFoo` to a `& *const RawFoo`? My slightly modified example causes Miri to complain about this: `error: Undefined Behavior: constructing invalid value: encountered a dangling reference (going beyond the bounds of its allocation)`.

2025-01-20T06:15:49.913Z+00:00

0

Reply

- Copy link

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

Related

3

How do I manage optional owned pointers in rust?

0

Using managed and owned pointers together

3

How do you borrow a mutable pointer in rust?

3

Can you convert an unsafe pointer to an owned pointer in rust somehow?

0

How can I create an owning pointer to an unsized type?

7

How do I deal with wrapper type invariance in Rust?

3

Convenient access to members of raw pointers?

0

Dereferencing raw pointer with explicit annotation in Rust

0

Raw pointers w/o unsafe

2

How to wrap a struct in Rust?

Hot Network Questions

- Where in the world is the chicken?
- Critiques of the real vs. nominal definition distinction
- Can a 10*10 square be paved with 1*4 rectangular stone plates?
- "Creatures your opponents control" after the ability is activated
- Zero Inflated Beta Regression (or not)?
- Is it possible to use LaTeX to create RPG-related PDFs, such as character sheets or adventure modules?
- How can I secure a loose kitchen cabinet?
- Is there a term for forming emotional or relational bonds with AI chatbots? If not, is “cyberpomorphic” a valid neologism?
- What's the point of swapping polarity between mod slots?
- Finding real variable where a 2×2 complex matrix is singular in Mathematica
- Why does Mtt 13:34 say that Jesus always spoke in parables?
- Is my homebrew Species "Attuned" balanced for D&D 2024?
- Why is "cmd | xargs" to join lines of cmd with spaces wrong
- Download is performed unsandboxed as root - Why am I getting this message, and how do I fix it?
- Does short range teleporting end floating disk?
- Ping sweep of IP subnets
- Proportional odds logistic regression for ordered category outcome - how to convert odds ratios to probabilities in this case?
- Can I Automatically List Supported Kernel Parameters in GRUB?
- Recent time traveling TV show. The lead might have been Asian(?), set in San Francisco(?)
- Storm Sphere and Obstruction
- 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?
- Installing root certificates by government
- Documentation for DUSHIN software

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.