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

Strange behaviour using println with raw pointer

Ask Question

Asked 3 years, 11 months ago

Modified 3 years, 11 months ago

Viewed 165 times

0

I need a binary tree data structure in C, and there is already exist a binary tree implementation in Rust. So I decided to wrap that.

I made a C-compatible struct contains raw pointer to BTreeMap and add some methods that take a pointer to my wrapper struct.

The problem is in module test. The test is passed if I using my methods only, but whenever I put the println! macro between method calls, The test failed.

```
use std::collections::BTreeMap;

#[repr(C)]
pub struct my_btree {
    btree: *mut BTreeMap<u64, u64>,
}

#[no_mangle]
pub extern "C" fn my_btree_new() -> *mut my_btree {
    let boxed = Box::new(BTreeMap::<u64, u64>::new());
    let mut ret = my_btree {
        btree: Box::into_raw(boxed),
    };

    &mut ret as *mut my_btree
}

#[no_mangle]
pub extern "C" fn my_btree_insert(
    btree: *mut my_btree,
    key: u64,
    val: u64,
) -> bool {
    unsafe {
        let mut boxed = Box::from_raw((*btree).btree);
        let contains = (*boxed).contains_key(&key);
        if contains {
            return false;
        }

        (*boxed).insert(key, val);
        println!("{:?}", boxed);
        Box::into_raw(boxed);

        return true;
    }
}

#[no_mangle]
pub extern "C" fn my_btree_contains(btree: *mut my_btree, key: u64) -> bool {
    unsafe {
        let boxed = Box::from_raw((*btree).btree);
        println!("{:?}", boxed);
        let ret = (*boxed).contains_key(&key);

        Box::into_raw(boxed);

        ret
    }
}

#[no_mangle]
pub extern "C" fn my_btree_free(btree: *mut my_btree) {
    unsafe {
        let _boxed = Box::from_raw((*btree).btree);
    }
}
```

```
#[cfg(test)]
mod tests {
   use super::*;

   #[test]
   fn insert() {
       let map = my_btree_new();
       my_btree_insert(map, 1, 1);
       my_btree_insert(map, 2, 2);
       my_btree_insert(map, 3, 30);
       let err = my_btree_contains(map, 1);
       assert_eq!(err, true);
       println!("???"); // If this line commented out, then test success without error.
       my_btree_free(map);
   }
}
```

when I run the test with below command,

```
$ cargo test -- --nocapture
```

In my terminal,

```
running 1 test
{1: 1}
{1: 1, 2: 2}
{1: 1, 2: 2, 3: 30}
{1: 1, 2: 2, 3: 30}
???
error: test failed, to rerun pass '--lib'
```

But if I comment out the `println!("???");` then test pass without any error.

And if I put println between my_btree_insert calls, it failed in next call. It's weird. Why this happened?

- rust

Share

Improve this question

Follow

asked Dec 10, 2021 at 2:39

hardboiled65

35144 silver badges1212 bronze badges

1

> 1
>
> welcome to undefined behavior, take a seat.
>
> Stargateur – Stargateur
>
> 2021-12-10 03:10:41 +00:00
>
> Commented Dec 10, 2021 at 3:10

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)

1

You've got several issues.

> In `my_btree_new`, you construct an instance of `my_btree` on the stack (ie. local to the function) and then return a pointer to it.
>
> In `my_btree_insert`, you take your pointer, then construct a `Box` around it. Before returning, you deconstruct the `Box` so that the item won't be freed - but you also have an early return path that does not deconstruct the `Box`. Your test case does not exercise that code path, but I expect it would crash if it did.

Why does it only crash when you insert the `println!`? Simple - the extra function call(s) that are generated trash the area of the stack containing the `my_btree`.

Here are a few suggestions for fixing it up:

You do not really need the wrapper struct (at least for this basic example). But if you are going to have one, the whole struct should be on the heap, not just the BTree structure.

You don't need to box/unbox the pointer in every method; it is not adding value and is just increasing the chances you will forget to rebox it in some code path, resulting in a double-free crash. Only re-box it in the `my_btree_free` function.

You have made all these functions safe with the code inside wrapped in an unsafe block. That's not really correct - the compiler cannot verify that the pointer being supplied is correct, therefore the function is not safe (or putting it another way, a safe function should not crash regardless of the arguments you supply).

Here is a version that works:

```
use std::collections::BTreeMap;

#[repr(C)]
pub struct my_btree {
   btree: BTreeMap<u64, u64>,
}

#[no_mangle]
pub extern "C" fn my_btree_new() -> *mut my_btree {
   let boxed = Box::new(my_btree { btree: BTreeMap::<u64, u64>::new() } );
   Box::into_raw(boxed)
}

#[no_mangle]
pub unsafe extern "C" fn my_btree_insert(
   btree: *mut my_btree,
   key: u64,
   val: u64,
) -> bool {
   let contains = (*btree).btree.contains_key(&key);
   if contains {
       return false;
   }
   (*btree).btree.insert(key, val);

   return true;
}

#[no_mangle]
pub unsafe extern "C" fn my_btree_contains(btree: *mut my_btree, key: u64) -> bool {
   (*btree).btree.contains_key(&key)
}

#[no_mangle]
pub unsafe extern "C" fn my_btree_free(btree: *mut my_btree) {
   let _boxed = Box::from_raw(btree);
}

#[cfg(test)]
mod tests {
   use super::*;

   #[test]
   fn insert() {
       let map = my_btree_new();
       unsafe {
           my_btree_insert(map, 1, 1);
           my_btree_insert(map, 2, 2);
           my_btree_insert(map, 3, 30);
           let err = my_btree_contains(map, 1);
           assert_eq!(err, true);
           println!("???"); // If this line commented out, then test success without error.
           my_btree_free(map);
       }
   }
}
```

[Share](#)

[Improve this answer](#)

Follow

answered Dec 10, 2021 at 3:15

[harmic](#)

31k66 gold badges7575 silver badges102102 bronze badges

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

2 Comments

Add a comment

hardboiled65

[hardboiled65](#) [Over a year ago](#)

Thank you! But this library is intended using in C. In C header file, the struct is defined as `typedef struct my_btree { void *btree; } my_btree;`. is member btree(type is BTreeMap<u64, u64>) safely compatible with C? I'm not sure about that.

2021-12-10T03:26:23.33Z+00:00

0

Reply

- Copy link

trent

[trent](#) [Over a year ago](#)

@hardboiled You should define the type in the header file as just `struct my_btree;` with no body. This is called an opaque type and it is provided for exactly this kind of situation: when the actual implementation of the type is irrelevant because the interface only uses pointers. In C, all object pointers are layout-compatible (which is not the same as having the same valid bit patterns, but is good enough for code like this).

2021-12-10T03:45:04.84Z+00:00

1

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

75
Why does printing a pointer print the same thing as printing the dereferenced pointer?
0
Raw pointer's data disappearing
4
Raw pointer turns null passing from Rust to C
0
Why is dereferencing a null raw pointer undefined behaviour?
0
Why does a node in a linked list using raw pointers become corrupted?
1
Why doesn't converting a u8 pointer into a [bool; 8] pointer yield the number bit-by-bit?
0
rust raw pointer not working without a println!() statement
6
Why can unsafe code not move-out non-copy types from behind a raw pointer?
3
Why can't I use println with a str?
0
Unexpected segfault when working with raw pointers

Hot Network Questions

- Can I Automatically List Supported Kernel Parameters in GRUB?
- "Us" in Hegel's "Greater Logic"
- How do Buddhists interpret the Buddha’s explanation of earthquakes in AN 8.70?

- "Imagine if..." vs. "Imagine..."
- Does short range teleporting end floating disk?
- Unity HDRP custom pass stencil buffer not working in build mode
- How to prevent a batch file call from escaping a caret character?
- Installing root certificates by government
- Where in the world is the chicken?
- Explicit examples separating Wasserstein distances from Jensen–Shannon divergence?
- I2C bus problems when using two distance sensors in Arduino
- When should I use tied notes vs. the whole note duration?
- Recent time traveling TV show. The lead might have been Asian(?), set in San Francisco(?)
- Vertical alignment of equations inside tabularx
- Statistical testing for bimodal/multimodal sample
- King defense setups
- Why does black allow white to have a protected passed central pawn here?
- Can a 10*10 square be paved with 1*4 rectangular stone plates?
- Why are left and right cosets different in the non-normal case?
- Why does Mtt 13:34 say that Jesus always spoke in parables?
- What do the terms postponement, slip and scrub mean in the context of the space shuttle program?
- Finding real variable where a 2×2 complex matrix is singular in Mathematica
- Do something and "win"
- Explain this seeming contradiction in Euclid Book 1 Proposition 16

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.

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