VecDeque: length 0 underflow and bogus values from pop_front(), triggered by a certain sequence of reserve(), push_back(), make_contiguous(), pop_front() #79808

Closed ayourtch opened this issue on Dec 7, 2020 · 9 comments · Fixed by #79814

Assignees

Labels A-collections C-bug I-unsound P-critical T-libs T-libs-api

ayourtch commented on Dec 7, 2020 • edited

This is my first bug report, so please correct me s if I miss anything :-)

I use VecDeque in a toy application (https://github.com/ayourtch/flex-sftp-server) which is handling external reads in a loop, and does possible partial handling of the input.

My pattern is I call reserve(N) before push_back() N times, then make_contiguous(), then doing some pop_front(). I noticed the app was behaving badly, and was able to generate a stand-alone test case which shows the bug.

The code is pretty boring and repetitive, so I put it at https://github.com/ayourtch/deq-bug/ to avoid spamming here.

Just issue "cargo run" after cloning out that code, and look for the word "BUG" within the terminal output:

According to the docs, pop_front() on an empty VecDeque should return None.

Instead, this happens:

```
pop: Some(75)
pop: Some(6e)
pop: Some(74)
pop: Some(75)
deq len: 0
pop: Some(2f)
BUG ^^^
deq len: 31
pop: Some(75)
pop: Some(62)
pop: Some(75)
pop: Some(6e)
```

If I set the boolean "do_reserve" to be false, thus getting rid of all the reserve() calls, I get the expected behavior:

```
pop: Some(75)
pop: Some(6e)
pop: Some(74)
pop: Some(75)
deq len: 0
pop: None
BUG ^^^
deq len: 0
pop: None
pop: None
pop: None
pop: None
pop: None
```

The same happens if I set "do_make_contiguous" to false as well - so calling both functions is a prerequisite to trigger the bug.
The same thing happens in debug and release builds.

Meta

The below is the version I found it in, but thanks a lot to Steve Klabnik for also testing it on nightly and getting the same buggy behavior.

`rustc --version --verbose`:

```
rustc 1.48.0 (7eac88abb 2020-11-16)
binary: rustc
commit-hash: 7eac88abb2e57e752f3302f02be5f3ce3d7adfb4
commit-date: 2020-11-16
host: x86_64-unknown-linux-gnu
release: 1.48.0
LLVM version: 11.0
```

ayourtch added the C-bug label on Dec 7, 2020

jonas-schievink added A-collections E-needs-mcve T-libs labels on Dec 7, 2020

naim94a commented on Dec 7, 2020

I created a minimal sample:

```rust
use std::collections::VecDeque;

fn ab(dq: &mut VecDeque<i32>, sz: usize) {
    for i in 0..sz {
        dq.push_back(i as _);
    }
    dq.make_contiguous();
    for _ in 0..sz {
        dq.pop_front();
    }
}

fn main() {
    let mut dq = VecDeque::with_capacity(2);
    ab(&mut dq, 2);
    ab(&mut dq, 3);

    dbg!(dq.len()); // this is zero

    dbg!(dq.pop_front()); // uaf+double frees
}
```

6

jonas-schievink added I-unsound and removed E-needs-mcve labels on Dec 7, 2020

rustbot added the I-prioritize label on Dec 7, 2020

camelid commented on Dec 7, 2020 Member

Hmm, if it double-frees then why does Miri not catch it?

camelid added P-critical and removed I-prioritize labels on Dec 7, 2020

camelid commented on Dec 7, 2020 Member

Assigning `P-critical` and removing `I-prioritize` as discussed in the prioritization working group.

camelid added the T-libs-api label on Dec 7, 2020

leonardo-m commented on Dec 7, 2020

Some time ago I've read somewhere that Rust stdlib VecDeque is cursed. I didn't believe them. I'm changing opinion... -.-

6

naim94a commented on Dec 7, 2020

> Hmm, if it double-frees then why does Miri not catch it?

This specific case doesn't double free (i32::drop is a noop), using a `Box` might have been a better example...

camelid commented on Dec 7, 2020 Member

Still, this is memory-unsafe, right?

naim94a commented on Dec 7, 2020

Yes, It's still a use-after-free regardless

1

This comment has been minimized. Sign in to view

lcnr self-assigned this on Dec 7, 2020

camelid commented on Dec 7, 2020 • edited Member

> Yes, It's still a use-after-free regardless

@naim94a Based on my conversation with lcnr, jonas-schievink, and others on Zulip, it seems that using `make_contiguous` with copy types is *not* UB (it still exhibits a bug though!) because copy types aren't freed. It *is* UB with non-copy types though (e.g. `Box`).

lcnr mentioned this issue on Dec 7, 2020

fix soundness issue in `make_contiguous` #79814

Merged

bors closed this as completed in `d32c320` on Dec 10, 2020

Qwaz mentioned this issue on Dec 21, 2020

Add advisories for standard library soundness bugs rustsec/advisory-db#539

Closed

matled mentioned this issue on Dec 22, 2020

VecDeque iter+collect causes an infinite loop (1.48.0) #80293

Closed

JohnTitor added a commit to JohnTitor/rust that referenced this issue on Jan 10, 2021

`Add another test case for rust-lang#79808` ··· `39e1331`

bors added a commit to rust-lang-ci/rust that referenced this issue on Jan 11, 2021

`Auto merge of rust-lang#80898 - JohnTitor:vecdeque-another-case, r=lcnr` ··· `cc4a086`

JohnTitor added a commit to JohnTitor/rust that referenced this issue on Jan 12, 2021

`Rollup merge of rust-lang#80898 - JohnTitor:vecdeque-another-case, r=…` ··· `a9e3125`

JohnTitor mentioned this issue on Jan 12, 2021

Rollup of 8 pull requests #80939

Merged

bors added a commit to rust-lang-ci/rust that referenced this issue on Jan 12, 2021

`Auto merge of rust-lang#80939 - JohnTitor:rollup-pymns4q, r=JohnTitor` ··· `b6b4616`

Assignees

lcnr

Labels

A-collections **C-bug** I-unsound **P-critical** **T-libs** **T-libs-api**

Projects

None yet

Milestone

No milestone

Development

Successfully merging a pull request may close this issue.

fix soundness issue in `make_contiguous`
lcnr/rust

7 participants