



Products

Using a struct to store a reference to a non-Copy value

log in sign up

Ask Question

Asked 3 years, 2 months ago

Active 3 years, 2 months ago

Viewed 152 times



1



I need an object that contains a reference to a process child and enables me to execute functions on it.

```
pub struct Shell {
    child: std::process::Child,
}

impl Shell {
    pub fn init() -> Shell {
        let mut cmd = std::process::Command::new("Command");
        let process = cmd.spawn();
        let new = Shell {
            child: process.unwrap(),
        };
        new
    }

    pub fn f1(mut self) {
        //do something with self
    }

    pub fn f2(mut self) {
        {
            let stdin = self.child.stdin.as_mut().unwrap();
        }
        let output = self.child.wait_with_output();
    }
}

fn main() {
    let mut shell = Shell::init();
    shell.f1();
    shell.f2();
}
```

```
error[E0382]: use of moved value: `shell`
--> src/main.rs:28:5
|
27 |   shell.f1();
|   ----- value moved here
28 |   shell.f2();
|   ^^^^^ value used here after move
= note: move occurs because `shell` has type `Shell`, which does not implement the `Copy` trait
```

[>Try it](#)

The problem is that when I initialize my object, I can call functions on the object only once, because the value is moved on the first call due to standard Rust behaviour.

A simple `#[derive(Copy, Clone)]` does not work here, because `std::process::Child` does not seem to implement the `Copy` trait. Is there a way to circumvent that or wrap it into something copy-able?

Test Implementations

When using a mutable reference as the function argument, the initial problem appears to be solved, however, it is then not possible to access the `self.child` more than once.

Your privacy

By clicking "Accept all cookies", you agree Stack Exchange can store cookies on your device and disclose information in accordance with our [Cookie Policy](#).

Accept all cookies

Customize settings

```
pub struct Shell {
    child: std::process::Child,
}

impl Shell {
    pub fn init() -> Shell {
        let mut cmd = std::process::Command::new("Command");
        let process = cmd.spawn();
        let new = Shell {
            child: process.unwrap(),
        };
        new
    }

    pub fn f1(&mut self) {
        //do something with self
    }

    pub fn f2(&mut self) {
        {
            let stdin = self.child.stdin.as_mut().unwrap();
        }
        let output = self.child.wait_with_output();
    }
}

fn main() {
    let mut shell = Shell::init();
    shell.f1();
    shell.f2();
}
```

```
error[E0507]: cannot move out of borrowed content
--> src/main.rs:21:22
|
21 |     let output = self.child.wait_with_output();
|           ^^^^^ cannot move out of borrowed content
```

[Try it](#)

Is there a way to solve that?

[struct](#) [reference](#) [rust](#)

Share

[Improve this question](#)

Follow

edited Oct 9 '18 at 19:32

asked Oct 9 '18 at 10:12



[rubib](#)

217 ● 1 ● 9

[Add a comment](#)

1 Answer

[Active](#) [Oldest](#) [Votes](#)



0



The problem is that `self.child` has to be consumed by `wait_with_output()`. This is why `self` must not be passed to `f2` by reference, but by value:

```
pub struct Shell {
    child: std::process::Child,
}

impl Shell {
    pub fn init() -> Shell {
        let mut cmd = std::process::Command::new("Command");
        let process = cmd.spawn();
        let new = Shell {
            child: process.unwrap(),
        };
        new
    }

    pub fn f1(&mut self) {
        //do something with self
    }

    pub fn f2(mut self) {
        {
            let stdin = self.child.stdin.as_mut().unwrap();
        }
        let output = self.child.wait_with_output();
    }
}

fn main() {
    let mut shell = Shell::init();
    shell.f1();
    shell.f2();
}
```

>[Try it](#)

However this implies that `f2` must be the last function that accesses `self.child` .

[Share](#)

[Improve this answer](#)

Follow

answered Oct 9 '18 at 19:20



[rubb](#)

217 ● 1 ● 9

[Add a comment](#)

Your Answer

Post Your Answer

By clicking "Post Your Answer", you agree to our [terms of service](#), [privacy policy](#) and [cookie policy](#)

Not the answer you're looking for? Browse other questions tagged [struct](#) [reference](#) [rust](#) or [ask your own question](#).

The Overflow Blog

- Sequencing your DNA with a USB dongle and open source code
- Don't push that button: Exploring the software that flies SpaceX rockets and...

Featured on Meta

- Providing a JavaScript API for userscripts
- Congratulations to the 59 sites that just left Beta

Related

[Cannot move out of borrowed content when borrowing a generic type](#)

[Factory method: instance does not live long enough](#)

[Why can't I store a value and a reference to that value in the same struct?](#)

[Reference to unwrapped property fails: use of partially moved value: `self`](#)







1
the trait `'_embedded_hal_digital_InputPin'` is not implemented for `'PE2<Output<OpenDrain>>'`

Awaiting a Number of Futures Unknown at Compile Time

0
Cannot call read on `std::net::TcpStream` due to unsatisfied trait bounds

Error: use of moved value: `'path'`. How to correct this code?

Hot Network Questions

-  'Cloth shop' and 'Clothes shop'
 -  What edges are not in a Gabriel graph, yet in a Delauney graph?
 -  Regular expressions within QGIS expressions: logical operator AND
 -  Calculating mean follow up from ONLY sample size and range
 -  Why does making a quantum circuit more noise resilient make it easier to simulate classically?
- [more hot questions](#)
-  [Question feed](#)

STACK OVERFLOW

[Questions](#)
[Jobs](#)
[Developer Jobs Directory](#)
[Salary Calculator](#)
[Help](#)
[Mobile](#)

PRODUCTS

[Teams](#)
[Talent](#)
[Advertising](#)
[Enterprise](#)

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](#)

site design / logo © 2021 Stack Exchange Inc; user contributions licensed under [cc by-sa](#). rev 2021.12.22.41046