درس ۱۱

Rust

همزمانی بی‌ترس

main.rs

```rust
let c = Arc::new(Mutex::new(0));
thread::spawn(move || {
  *c.lock().unwrap() += 1;
});
```

یک کورس رایگان از MyWebUniversity

برنامه‌نویسی Rust

درس ۱۱

smart pointer ها و همزمانی بی‌ترس

Box، Rc، RefCell، thread ها، closure های move، channel ها، Arc و Mutex



Box، Rc و RefCell

Box یک مقدار را با یک مالک روی heap می‌گذارد، برای نوع‌های بازگشتی مثل درخت‌ها لازم است. Rc ...

RefCell اجازه می‌دهد داده پشت یک reference شریک تغییر کند، با قاعده‌های قرض که هنگام اجرا ...

نوع	به شما می‌دهد	thread؟
`Box<T>`	heap، یک مالک	✓
`Rc<T>`	مالکان شریک	✗
`RefCell<T>`	تغییر از طریق &	✗
`Arc<T>`	شریک، atomic	✓
`Mutex<T>`	تغییر با قفل	✓

01

ساختن thread ها

thread::spawn یک closure را روی یک thread نو سیستم عامل اجرا می‌کند. join روی handle منتظر ...

closure به move ضرورت دارد تا مالک داده‌ای باشد که استفاده می‌کند، چون thread ممکن است ...

main.rs

```rust
use std::thread;

let files = vec!["a.log", "b.log", "c.log"];
let handles: Vec<_> = files.into_iter().map(|f| {
    thread::spawn(move || count_lines(f))
}).collect();

let total: usize = handles.into_iter().map(|h| h.join().unwrap()).sum();
```

02

channel ها

channel های mpsc مقدارها را بین thread ها می‌فرستند: تولیدکننده‌های زیاد، یک مصرف‌کننده. ...

فرستادن یک مقدار آن را move می‌کند، پس دو thread هرگز نمی‌توانند همزمان از آن استفاده کنند.

RUST PROGRAMMING · 11

main.rs

```rust
use std::sync::mpsc;

let (tx, rx) = mpsc::channel();
for id in 0..3 {
    let tx = tx.clone();
    thread::spawn(move || tx.send(format!("worker {id} done")).unwrap());
}
drop(tx);
for msg in rx { println!("{msg}"); }
```

03

Arc و Mutex

برای شریک کردن وضعیت قابل تغییر بین thread ها، آن را در Arc<Mutex> بپیچانید. Arc مالکیت را ...

lock یک guard برمی‌گرداند، و قفل وقتی guard از scope بیرون می‌رود خودکار آزاد می‌شود.

main.rs

```rust
use std::sync::{Arc, Mutex};

let counter = Arc::new(Mutex::new(0));
let mut hs = vec![];
for _ in 0..8 {
    let c = Arc::clone(&counter);
    hs.push(thread::spawn(move || { *c.lock().unwrap() += 1; }));
}
for h in hs { h.join().unwrap(); }
println!("{}", *counter.lock().unwrap());    // 8
```

04

چرا بی‌ترس است

trait های Send و Sync نشان می‌دهند کدام نوع‌ها برای move یا شریک شدن بین thread ها امن ...

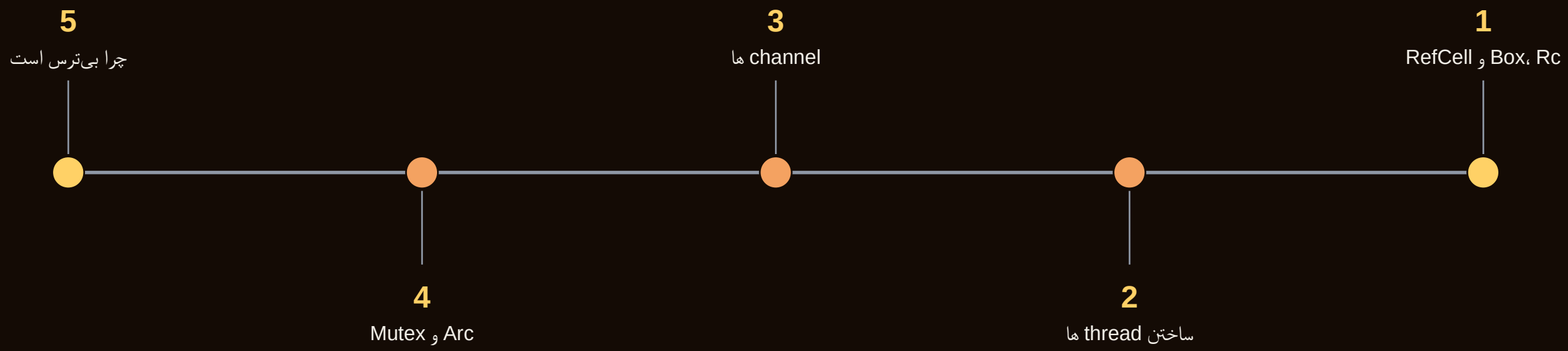
data race ها خطای کمپایل می‌شوند، نه باگ‌هایی که ساعت سه صبح دنبال آن‌ها می‌گردید. برای کود ...

terminal

```
error[E0277]: `Rc<i32>` cannot be sent between threads safely
   |
8  |     thread::spawn(move || println!("{}", rc));
   |     ^^^^^^^^^^^^^ `Rc<i32>` cannot be sent between threads safely
   = help: use `Arc<i32>` instead
```

05

نقشهٔ درس


1
Box، Rc و RefCell
2
ساختن thread ها
3
channel ها
4
Arc و Mutex
5
چرا بی‌ترس است

منابع و مطالعهٔ بیشتر

نوع	به شما می‌دهد	thread؟
`Box<T>`	heap، یک مالک	✓
`Rc<T>`	مالکان شریک	✗
`RefCell<T>`	تغییر از طریق &	✗
`Arc<T>`	شریک، atomic	✓
`Mutex<T>`	تغییر با قفل	✓

RUST PROGRAMMING · 11

main.rs

```rust
use std::sync::mpsc;

let (tx, rx) = mpsc::channel();
for id in 0..3 {
    let tx = tx.clone();
    thread::spawn(move || tx.send(format!("worker {id} done")).unwrap());
}
drop(tx);
for msg in rx { println!("{msg}"); }
```

تشکر از تماشای شما

MyWebUniversity.com

آموزش رایگان تکنالوژی از سال ۲۰۰۰ — انگلیسی و دری

@MyWebUniversity-Dari · @MyWebUniversity