Fearless concurrency

An elevator pitch for the Rust programming language

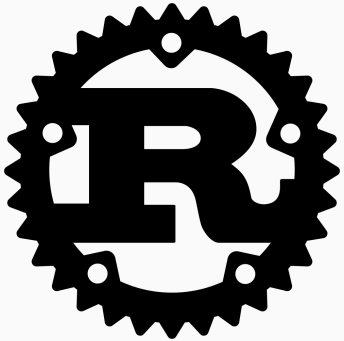
Christian Bourjau

September 8, 2017

Niels Bohr Institute
University of Copenhagen

What is Rust?

- System programming language
- Designed for maintainability of large code bases (Firefox)
- Minimizes run-time failures: If it compiles it works



"A pitfall with threads" (D. Pipardo)

```
void f(const std::string& s){
    std::cout << s;
}

void g(){
  std::string s("Hello");
  std::thread t(f,s);
  t.detach();
}
```

Very hard to collaborate and maintain! **Can't this be automatized at compile time?**

Rust's Ownership Rules

- Each value always has **one** owner
- Ownership can be moved
- Rust traces ownership at compile-time

Will this compile?

```
let s1 = String::from("hello");
let s2 = s1;

println!("{}, world!", s1);
```

No, it won't compile:

```
error[E0382]: use of moved value: ‘s1‘
 --> src/main.rs:4:27
  |
3 |     let s2 = s1;
  |         -- value moved here
4 |     println!("{}, world!", s1);
  |                            ^^ value used here after move
  |
```

Ownerships and threads

Does this compile?

```
let s = String::from("hello");

// Create new thread with lambda function
thread::spawn(move || {
    println!("{}, world!", s);
});
```

Ownerships and threads

Does this compile?

```
let s = String::from("hello");

thread::spawn(move || {
    println!("{}, world!", s);
});
println!("{}, world!", s);
```

Rust saved the day!

```
error[E0382]: use of moved value: ‘s‘
 --> src/main.rs:9:28
  |
6 |     let handle = thread::spawn(move || {
  |                                ----- value moved (into closure) here
...
9 |     println!("{}, world!", s);
  |                            ^ value used here after move
  |
```

Compiler guarantees to find these bugs even in huge code base!

There is much more...

- Borrowing
- Type system
- Data-oriented paradigm
- ...

Backup

Backup

References I