The Rust programming language

Denis Merigoux

Prosecco – Inria

Dec 18th 2018

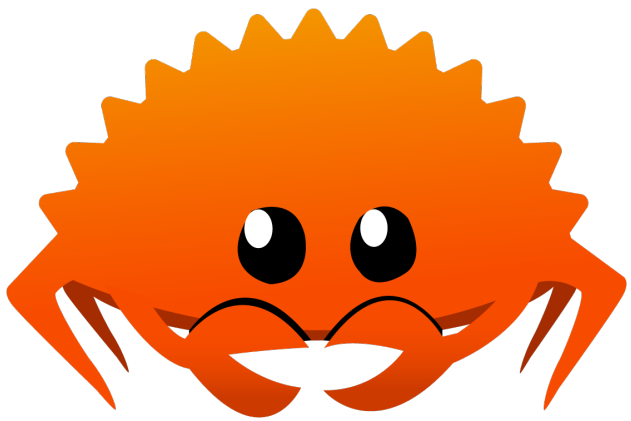
What is Rust ?

- You use C, C++ or even Fortran because you really care about yout programs running fast, but wish you could spend less of your time chasing mysterious segmentation faults?
- You're a functional programmer and want to get your hands dirty with systems programming?
- You want to create an optimized but safe library for the critical part of your Python scripts?

What is Rust ?

- You use C, C++ or even Fortran because you really care about yout programs running fast, but wish you could spend less of your time chasing mysterious segmentation faults?
- You're a functional programmer and want to get your hands dirty with systems programming?
- You want to create an optimized but safe library for the critical part of your Python scripts?

Then you will love:



```
enum TrafficLight { Green, Yellow, Red }

fn traffic_light_approach(
  car: &mut Car,
  in_a_hurry: bool,
  light: &TrafficLight
) {
  match light {
    TrafficLight::Green => car.maintain_course(),
    TrafficLight::Yellow => {
      if in_a_hurry { car.accelerate(); }
      else { car.slow_down_and_stop(); }
    },
    TrafficLight::Red => car.slow_down_and_stop()
  }
}
```

Key ideas of Rust

- Designed for systems programming
- Zero-cost abstractions
- “Pay as you use” for reference-counted pointers, etc.

Key ideas of Rust

- Designed for systems programming
- Zero-cost abstractions
- “Pay as you use” for reference-counted pointers, etc.

Borrowing and ownership

The type system of Rust checks that at any time, for every object there exists:

1. either a single pointer that can mutate the object points to it;
2. either one or more read-only pointers points to it.

Key ideas of Rust

- Designed for systems programming
- Zero-cost abstractions
- "Pay as you use" for reference-counted pointers, etc.

Borrowing and ownership

The type system of Rust checks that at any time, for every object there exists:

1. either a single pointer that can mutate the object points to it;
2. either one or more read-only pointers points to it.

Rust is safe

If a program typechecks, it is memory-safe and data-race-free !

The mighty borrow checker

```
let v = vec![1, 2, 3];
let v2 = v;
println!("v[0] is: {}", v[0]);

/*
let v2 = v;
         - value moved here
println!("v[0] is: {}", v[0]);
                        ^ value borrowed here after move
*/
```

The mighty borrow checker

```
let v = vec![1, 2, 3];
let v2 = &v;
println!("v[0] is: {}", v[0]);

/*
Compilation OK !
*/
```

Lots of features !

- Package and dependency manager, building tool, testing and benchmarking framework:
  cargo
- Automatic formatting, on-the-fly linting, IDE support:
  Rust Language Server
- Fast-growing community and code base:
  `https://crates.io`
- Designed for safe multi-threaded parallelization:
  Fearless Concurrency
- Interoperability with **C/C++** or **Python**

Try it yourself !

`https://www.rust-lang.org`