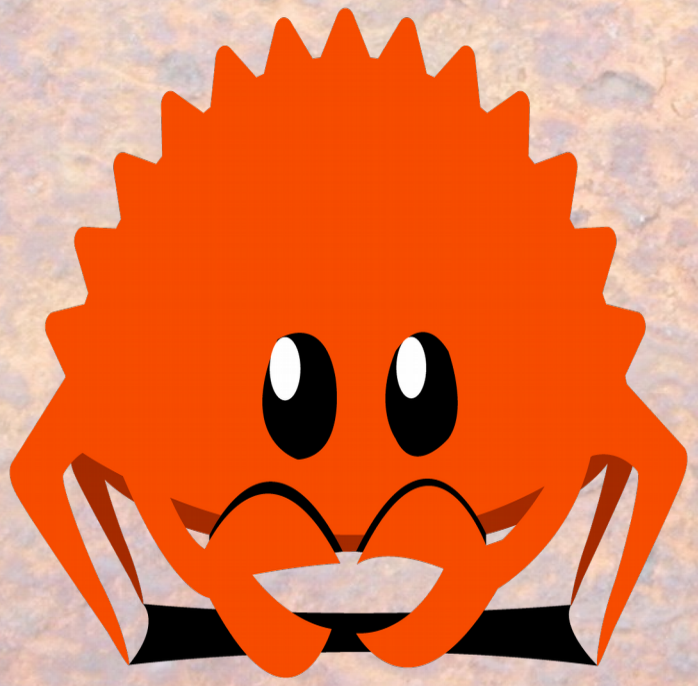
Programming with Rust

Marcel Schlitzer

UbuCon Europe 2016
19. November



Marcel Schlitzer

- From Dortmund
- Working for otris software AG as a Software Developer
- Hobby-Rustacean
- Twitter: @m-Schlitzer

Contents

- An Introduction to Rust
- Rust “in-depth”
- The future of Rust

Basic information about Rust

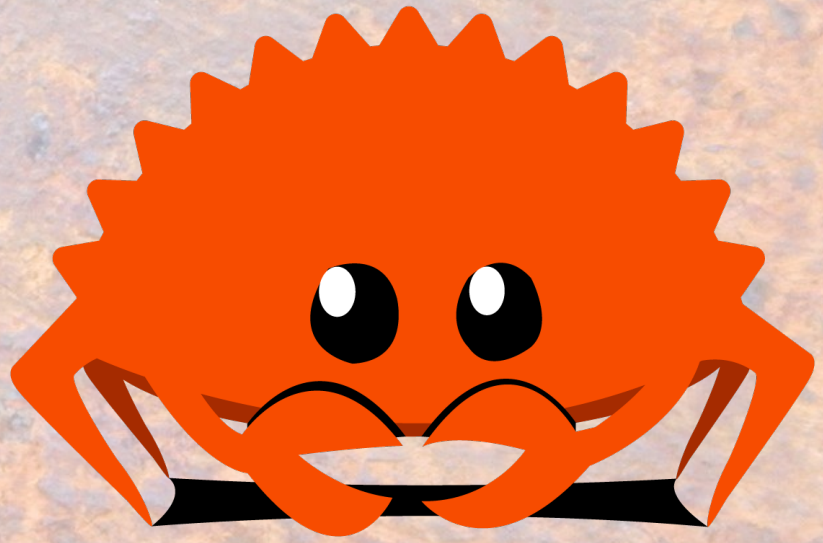
- First stable release in May 2015
- Current stable release 1.13.0
- Beta 1.14.0
- Nightly 1.15.0

Basic information about Rust

- Licensed under MIT/Apache 2.0 Licenses
- C/C++ like syntax
- Designed by Graydon Hoare

Fun facts about Rust

- Won “most loved programming language” Stack Overflow developer survey 2016
- Packages are called “crates”
- Rust users = “Rustaceans”



- Is believed to take its name from the rust family of fungi

The code

```
1 fn main() {
2     println!("Hello, UbuCon!");
3 }
```

The execution

```
m-schlitzer@schlitzer:~/Documents/Talk/hello master* 40s ± cargo run
   Compiling hello v0.1.0 (file:///home/m-schlitzer/Documents/Talk/hello)
     Running `target/debug/hello`
Hello, UbuCon!
```

Why Rust?

- It’s safe by default
- It’s concurrent
- It’s fast
- Easily expandable

But, what’s not good?

- Sometimes too strict
- Larger binaries
- Longer compile times
  - About 2x slower than usual

```c
#include <stdio.h>

int main(void) {
    printf("What's your name?\n");
    char input[100] = {0};
    scanf("%s", input);
    printf("Hello %s!\n", input);
    return 0;
}
```

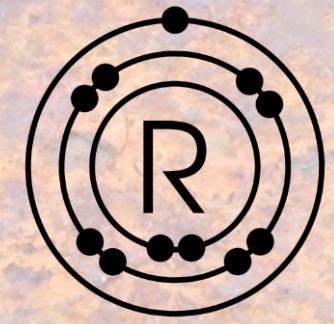
Rewriting this in Rust, you may get something like the following:

```rust
use std::io;

fn main() {
    println!("What's your name?");
    let mut input = String::new();
    io::stdin().read_line(&mut input).unwrap();
    println!("Hello {}!", input);
}
```

```rust
#![feature(lang_items)]
#![feature(libc)]
#![feature(no_std)]
#![feature(start)]
#![no_std]

extern crate libc;

extern "C" {
    fn printf(fmt: *const u8, ...) -> i32;
    fn scanf(fmt: *const u8, ...) -> i32;
}

#[start]
fn start(_argc: isize, _argv: *const *const u8) -> isize {
    unsafe {
        printf(b"What's your name?\n\0".as_ptr());
        let mut input = [0u8; 100];
        scanf(b"%s\0".as_ptr(), &mut input);
        printf(b"Hello %s!\n\0".as_ptr(), &input);
        0
    }
}

#[lang="eh_personality"] extern fn eh_personality() {}
#[lang="panic_fmt"] fn panic_fmt() -> ! { loop {} }
#[lang="stack_exhausted"] extern fn stack_exhausted() {}
```

Where is Rust used?

- Mozillas “Servo” browser engine
- Redox (OS)



- And many more..

Starting with Rust

- What do you need?
  - rustc (the compiler)
  - Cargo (the package manager)

- Where can you get it?
  - Just use apt to install it
  - e.g. **`sudo apt install rustc cargo`**

OS Support

- What other Operating Systems have Rust?
  - Most major GNU/Linux distros
  - Windows
  - Mac

Rust basics

- Package Manager: **Cargo**
- Packages hosted on crates.io
- Compiler: **rustc** (written in Rust)
- Debugging possible with GDB

Rust basics

- IDE plugins
  - Visual Studio
  - Eclipse
  - Intellij IDEA
- Texteditor plugins
  - Emacs
  - Vim
  - Sublime
  - And more

Setting it up

- Run **`cargo new foo --bin`**

```
m-schlitzer@schlitzer:~/Documents/Talk $ cargo new foo --bin
```

- Let's take a look

```
m-schlitzer@schlitzer:~/Documents/Talk $ tree foo
foo
├── Cargo.toml
└── src
    └── main.rs

1 directory, 2 files
```

The file contents

```
1 [package]
2 name = "foo"
3 version = "0.1.0"
4 authors = ["Marcel Schlitzer <marcel.schlitzer@gmail.com>"]
5
6 [dependencies]
```

```
1 fn main() {
2     println!("Hello, world!");
3 }
```

Rust in-depth

- Zero-cost abstractions

- Lifetimes: the answer to memory safety
  - No segfaults
  - Predictable cleanup of resources
  - Lower memory management overhead
  - Essentially no runtime system

Rust in-depth

- Macros
  - println!
  - panic!
  - Whatever you want!
- Boxes:
  - Content:
    - *Actual Magic*
    - *Fairy Dust*
  - In reality:
    - Smart Pointers

Rust in-depth

- Traits
  - “Abilities for variables”
  - Drop
    - Can be used like a destructor
  - mut
    - Makes a variable mutable

Rust in-depth

```
struct Firework {
    strength: i32,
}

impl Drop for Firework {
    fn drop(&mut self) {
        println!("BOOM times {}!!!", self.strength);
    }
}

fn main() {
    let firecracker = Firework { strength: 1 };
    let tnt = Firework { strength: 100 };
}
```

This will output:

```
BOOM times 100!!!
BOOM times 1!!!
```

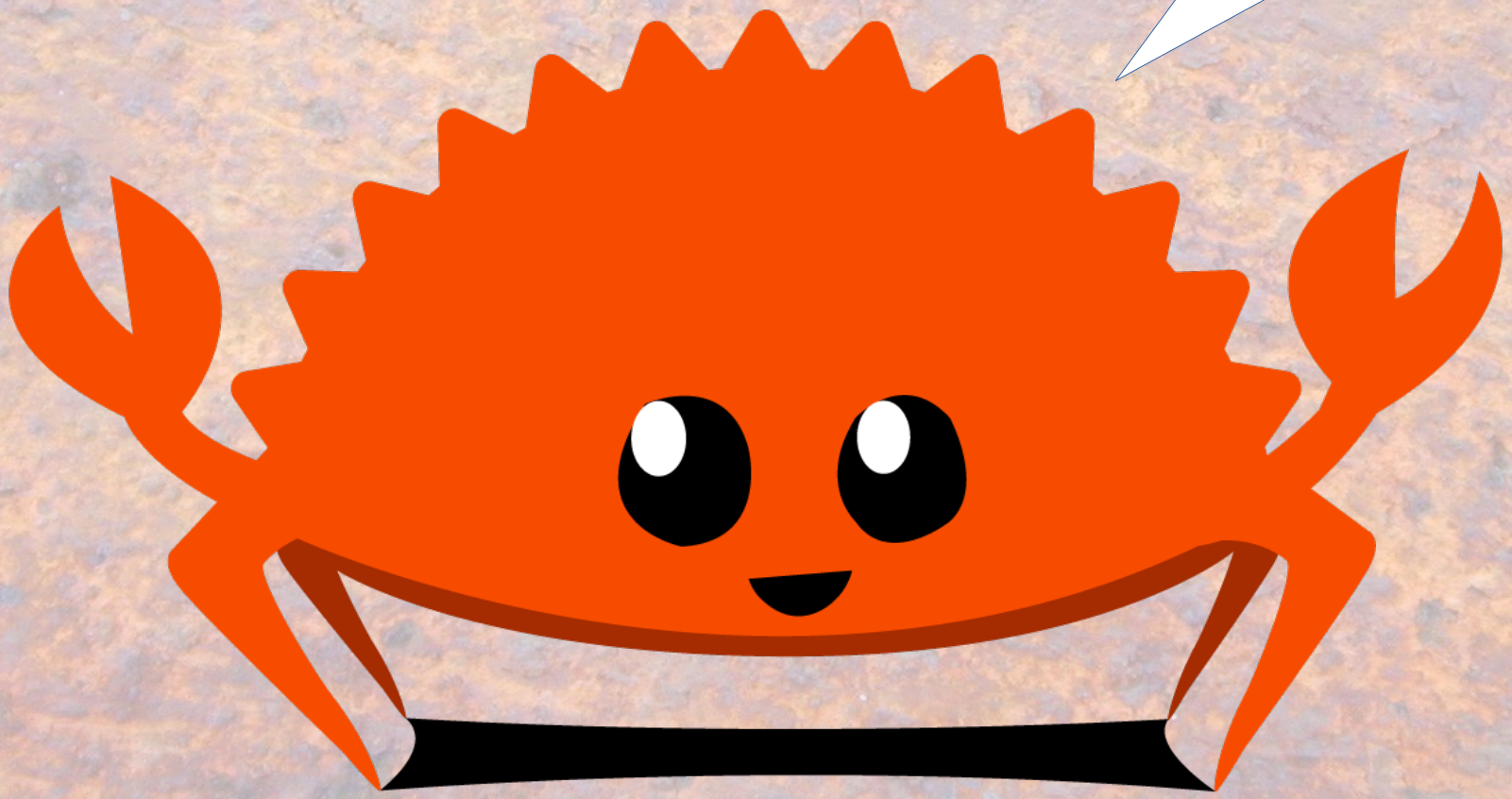
The future of Rust

- What we might get:
  - Optional garbage collection
  - Faster compile times
  - Integrated benchmarking infrastructure
  - C++ like templates

- What is wanted:
  - Beer & Bacon

Credits and useful links

- rust-lang.org
- rustbyexample.com
- The “book” - https://doc.rust-lang.org/book/
- Ferris the Crab – made by Karen Rustad Tölva


Thank you for your attention