

Eliminando Gargalos de Processamento Utilizando Rust

Johnathan Fercher

1. Introdução
2. Quem usa? E para que?
3. Sugestões de regras do Velocity
4. Mostre-me o código
5. Curiosidades
6. Material de estudos

Introdução

- IO-Bound;
- CPU-Bound;

- IO-Bound → Async Programming;
- CPU-Bound → Algorithms, Parallel Programming, Programming Languages;

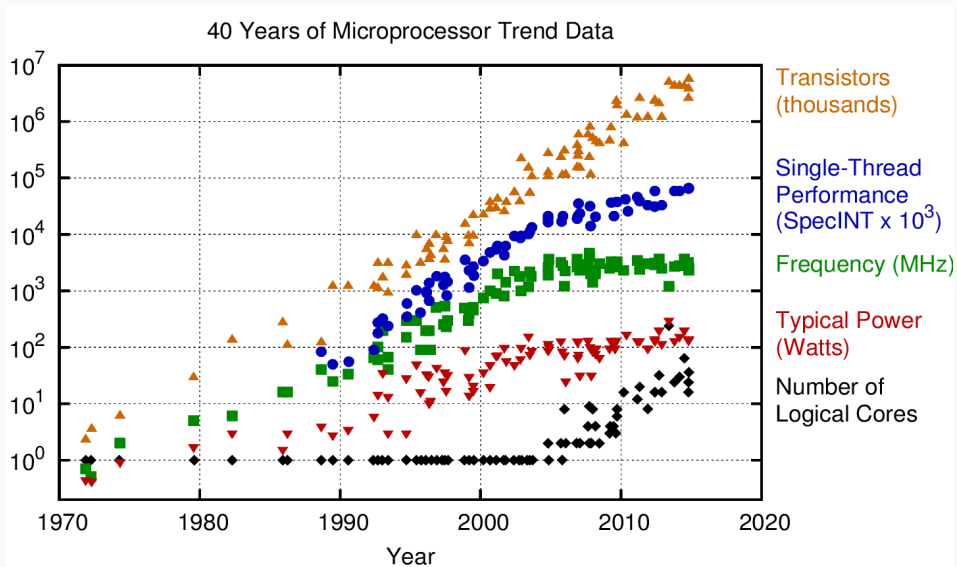


- Programming Language;
- Memory and thread-safe;
- Rust \rightarrow LLVM \rightarrow EXE;
- C-Bindings;
- Object-Oriented and Functional;
- Unit-tests and Package Manager;
- Interfaces and Generics;
- Without Garbage Collector;

"O clock dos processadores dobra a cada 18 meses."

Lei de Moore, 1965.

Motivação



"The way the processor industry is going, is to add more and more cores, but nobody knows how to program those things. I mean, two, yeah; four, not really; eight, forget it."

Steve Jobs, Apple.

Motivação

Programação Paralela



Problemas:

- Data races;
- Deadlock;
- Use After Free;
- Double Free;

Bug 650064

Running Aurora and Firefox in parallel

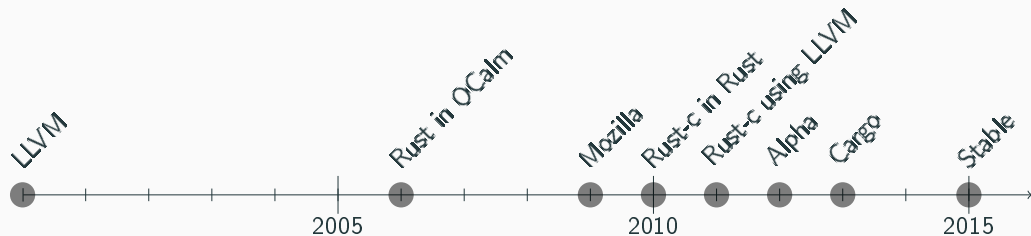
UNCONFIRMED Unassigned

▼ Status

Product: Firefox ▼
Component: General ▼
Status: UNCONFIRMED

Reported: 8 years ago
Modified: 6 years ago

História



- Licença MIT no Github;
- Duas versões: Stable e Nightly;
- Atualizações a cada 6 semanas;
- Processo de RFC;
- Quando uma RFC é aprovada ela é adicionada na versão Nightly;
- Após algum tempo em Nightly, ela pode ser adicionada na versão Stable, deixada de lado ou alterada;

Quem usa? E para que?

Quem usa? E para que?



"Optimizing cloud file-storage."

The Canonical logo, featuring the word "CANONICAL" in a dark red, sans-serif font. The letter "O" is replaced by a dark red circle with a white dot in the center, resembling a target or a stylized eye.

"Everything from server monitoring to middleware!"



"Developing memory-safe embedded applications on our SmartThings Hub and supporting services in the cloud."

moz://a

"Building the Servo browser engine, integrating into Firefox, other projects."



"Programming Assignments in secured Docker containers."

Quem usa? E para que?



"Letting you develop, deploy and manage infrastructure, run-time environments and applications."



"We use Rust in a service for analyzing petabytes of source code."



"Replacing C and rewriting performance-critical bottlenecks in the registry service architecture."

Quem usa? E para que?



"Using Rust instead of C/C++ to let you build, deploy and manage packages."

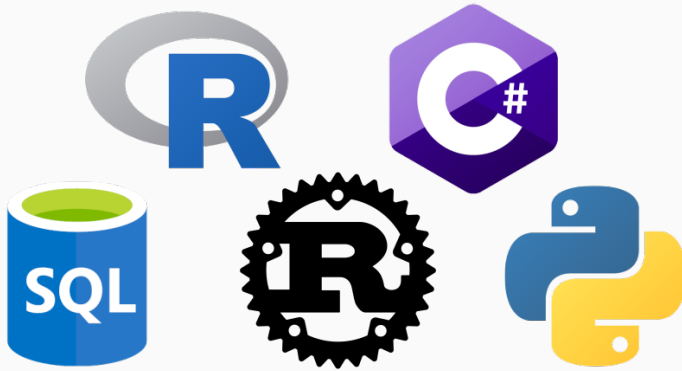


"Extracting anti-fraud insights by analyzing millions of transactions from hundreds of e-commerces."

Quem usa? E para que?

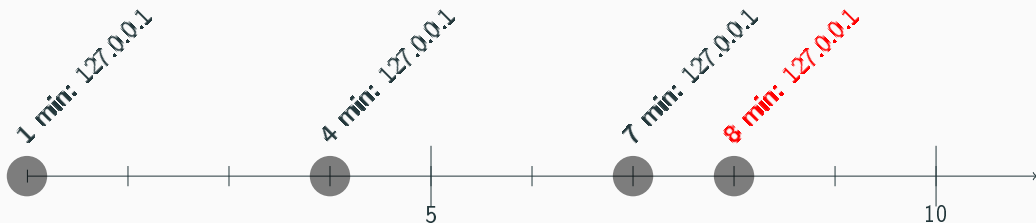
- No site oficial da linguagem há mais 123 empresas que deixaram claro que utilizam Rust em produção;

Sugestões de regras do Velocity



Regras de repetição:

- 5 repetições de um **Cpf** em 10 minutos;
- 10 repetições de um **Cartão** em 2 dias;
- 2 repetições de um **Ip** em 5 minutos;



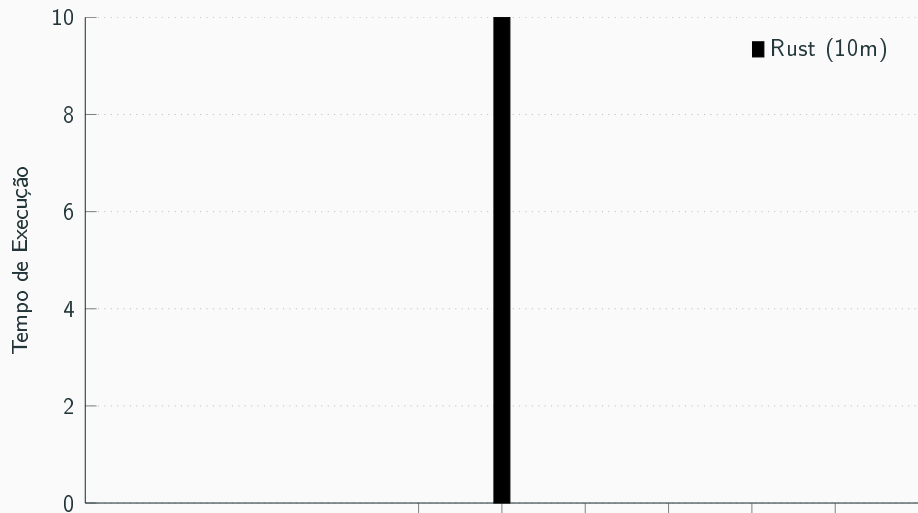
Algoritmo

```
var transactions = new List<Transaction>();  
var quarantine = new DateTime();  
  
for transaction in transactions do  
    if quarantine_is_active(quarantine, transaction) then  
        block(transaction);  
        update(quarantine);  
    else  
        if extrapolate_rule(transaction) then  
            block(transaction);  
            update(quarantine);  
        end  
    end  
end  
end
```

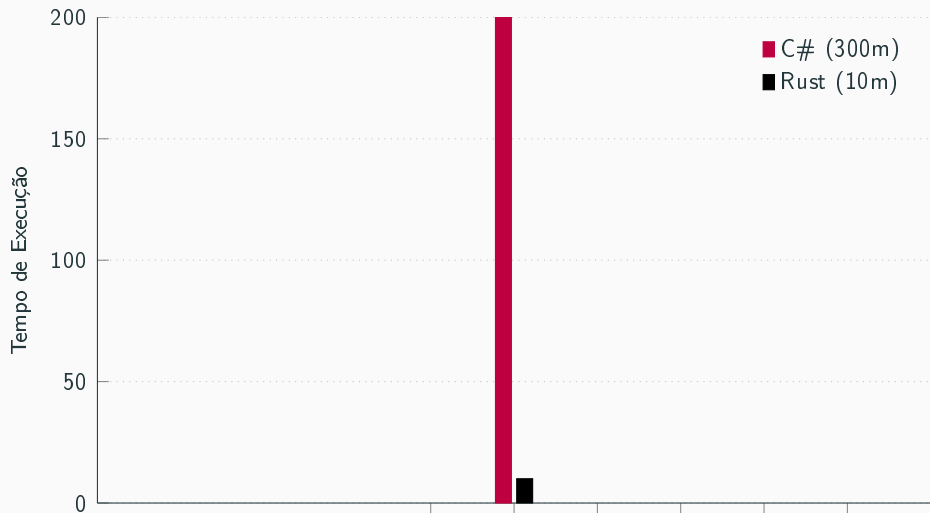
Hardware: I7 8770 (6 núcleos + 6 threads), 8GB RAM DDR4, SSD;

- Todas as versões representam o Velocity de forma exata;
- Não é utilizado agrupamento e nem programação funcional;
- C# e Rust paralelizam o processamento;
- C# e Rust realmente processaram 640 regras, o tempo de R é uma estimativa;
- R executou no banco SQL de monitoria;

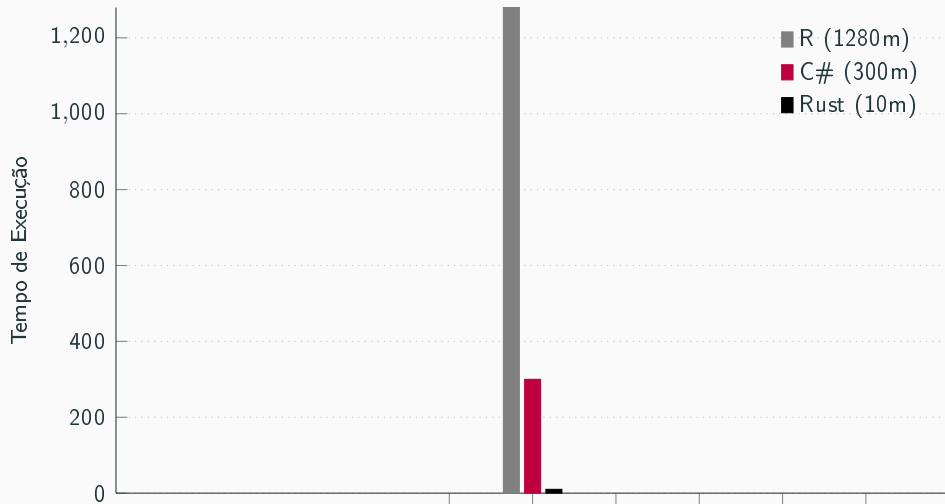
Benchmark (v1.0)



Benchmark (v1.0)



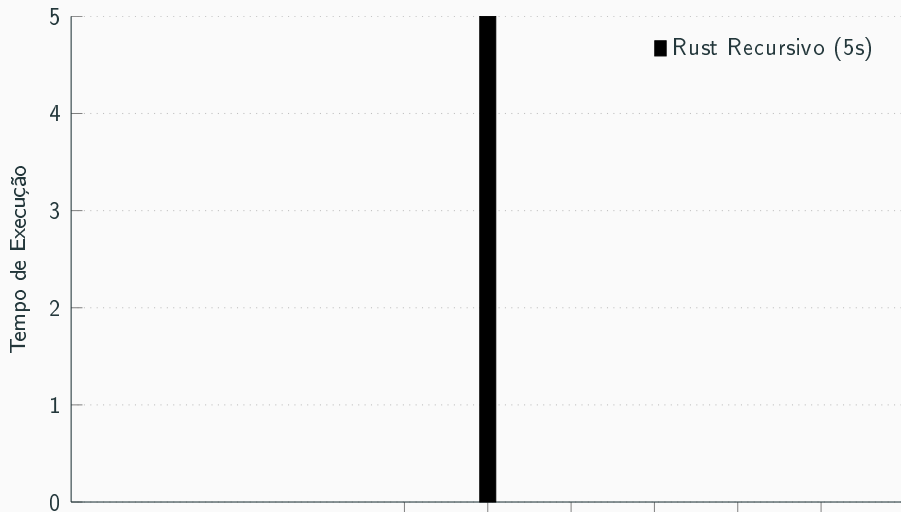
Benchmark (v1.0)



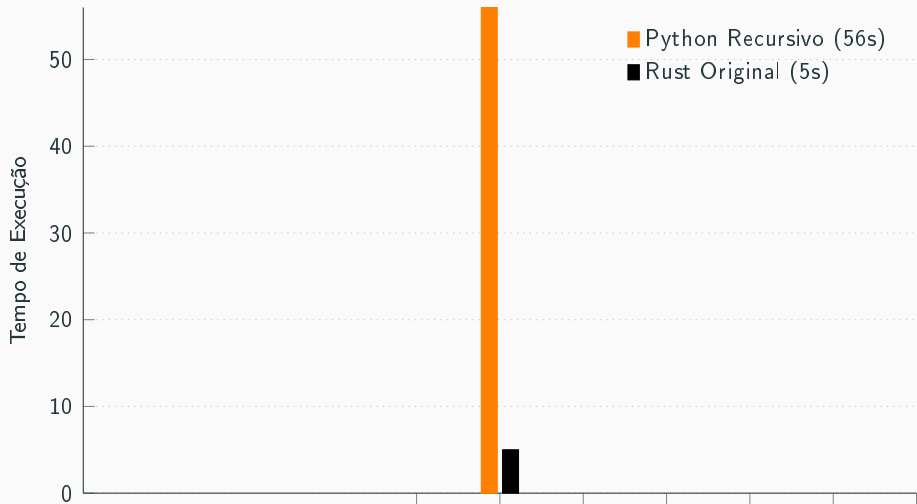
Hardware: I7 8770 (6 núcleos + 6 threads), 8GB RAM DDR4, SSD;

- Todos os algoritmos utilizam agrupamento;
- Rust e Python utilizam programação funcional;
- Duas versões do algoritmo: Original e Recursivo;
- Python utiliza a lib Numpy, que é feita em C, C++ e Fortran;

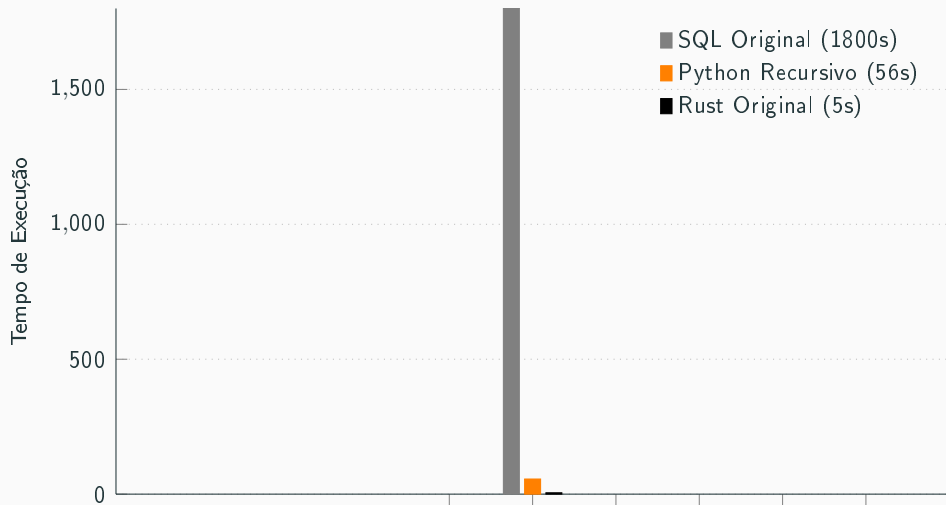
Benchmark (v2.0)



Benchmark (v2.0)



Benchmark (v2.0)



Mostre-me o código

Curiosidades

- Em apenas 3 anos o gerenciador de pacotes de Rust ultrapassou os gerenciadores de R e Haskell em número de pacotes;
- Em 2017 a Red Hat começou a dar suporte para Rust;
- Rust possui uma imagem oficial no Docker Hub;
- Em 2017 todas as IDEs da IntelliJ passaram a dar suporte para Rust;

- Amazon e Google estão contratando desenvolvedores Rust;
- Facebook e Github fazem parte dos Gold Sponsors da RustConf;
- Intel encoraja a utilização de Rust;
- Rust possui interoperabilidade com C, C++, C#, Swift, Python e Node.JS;
- Rust possui um gerador automático de APIs para bibliotecas C e C++;

Rust Playground

The screenshot shows the Rust Playground web interface. At the top is a browser address bar with the URL `https://play.rust-lang.org`. Below the browser is a toolbar with buttons for **RUN**, **DEBUG**, **STABLE**, **SHARE**, **TOOLS**, **CONFIG**, and a help icon. The main area contains a code editor with the following Rust code:

```
1 fn diego_otario() -> String {  
2     String::from("Diego Otário")  
3 }  
4  
5 fn main() {  
6     println!("Verdade: {}", diego_otario());  
7 }
```

Below the code editor is a panel with two tabs: **Execution** (selected) and **Close**. The **Execution** tab is divided into two sections: **Standard Error** and **Standard Output**. The **Standard Error** section contains the following text:

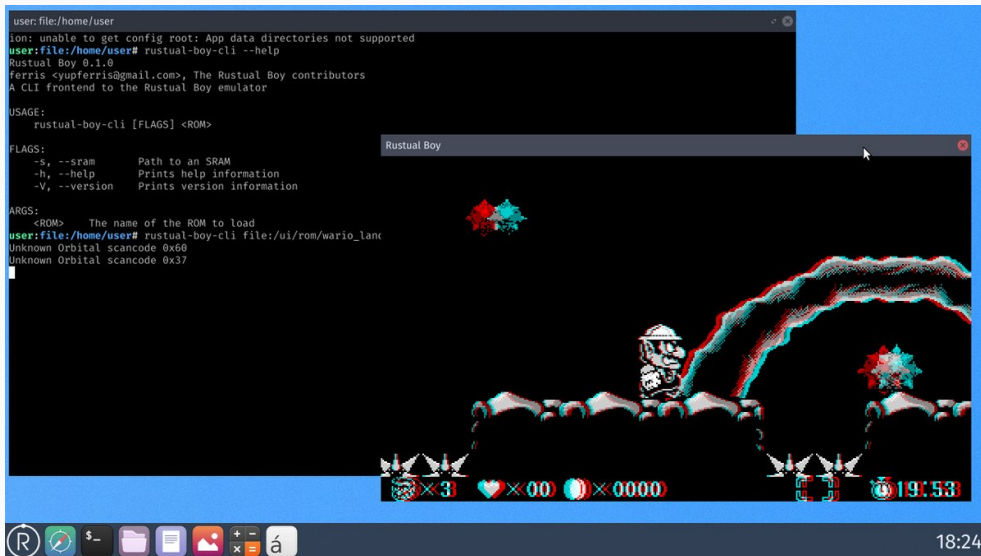
```
Compiling playground v0.0.1 (file:///playground)  
Finished dev [unoptimized + debuginfo] target(s) in 0.60s  
Running `target/debug/playground`
```

The **Standard Output** section contains the following text:

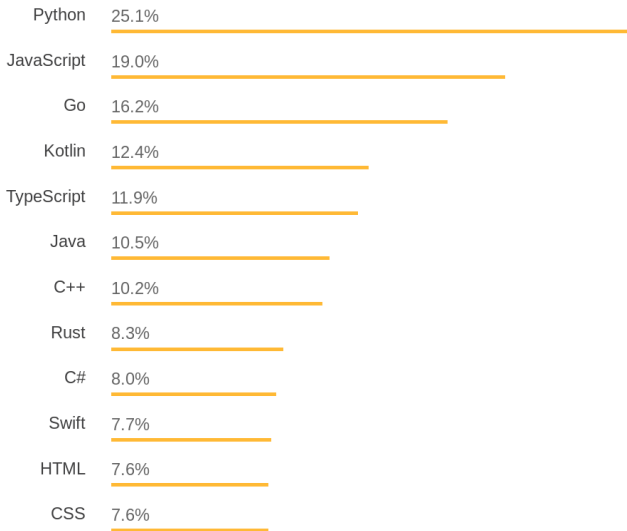
```
Verdade: Diego Otário
```



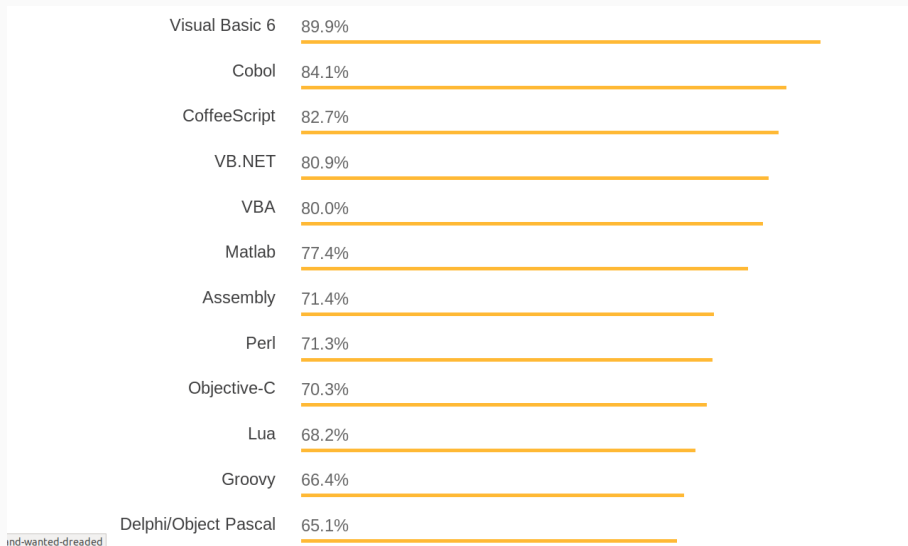
Sistema Operacional



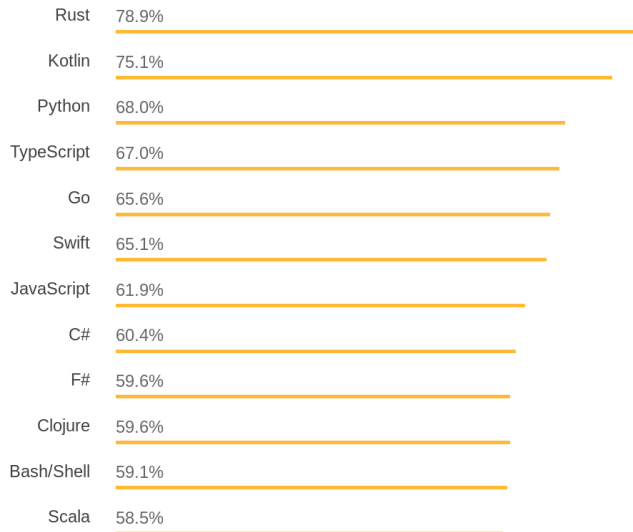
Stackoverflow Insights: Wanted



Stackoverflow Insights: Dreaded



Stackoverflow Insights: Loved



Material de estudos

Introduction

1. Hello World

1.1. Comments

1.2. Formatted print

1.2.1. Debug

1.2.2. Display

1.2.2.1. Testcase: List

1.2.3. Formatting

2. Primitives

2.1. Literals and operators

2.2. Tuples

2.3. Arrays and Slices

3. Custom Types

3.1. Structures

3.2. Enums

3.2.1. use

3.2.2. C-like

3.2.3. Testcase: linked-list

3.3. constants

4. Variable Bindings



Rust by Example

[Rust](#) is a modern systems programming language focusing on safety, speed, and concurrency. It accomplishes these goals by being memory safe without using garbage collection.

Rust by Example (RBE) is a collection of runnable examples that illustrate various Rust concepts and standard libraries. To get even more out of these examples, don't forget to [install Rust locally](#) and check out the [official docs](#). Additionally for the curious, you can also [check out the source code for this site](#).

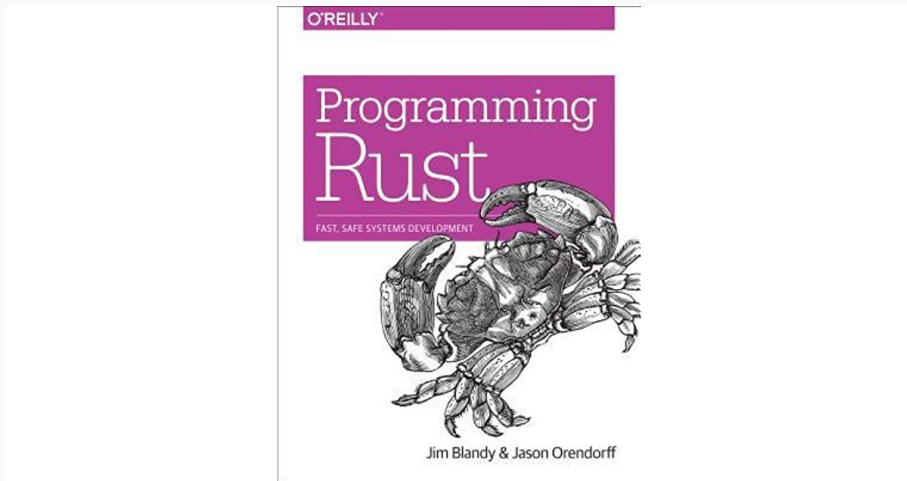
Now let's begin!

- [Hello World](#) - Start with a traditional Hello World program.
- [Primitives](#) - Learn about signed integers, unsigned integers and other primitives.
- [Custom Types](#) - `struct` and `enum`.
- [Variable Bindings](#) - mutable bindings, scope, shadowing.
- [Types](#) - Learn about changing and defining types.

Rust - Concorrência e alta performance com segurança

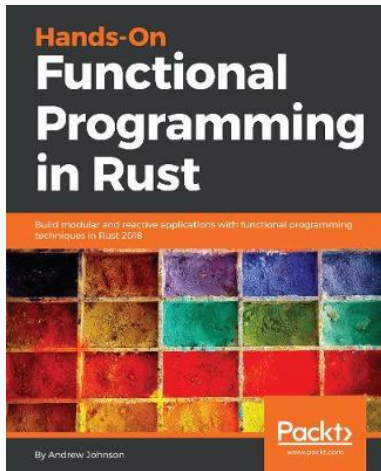


Programming Rust



Livro mais popular do Safari Ebooks no mês de seu lançamento em Outubro de 2017

Hands-On Functional Programming in Rust



Rust Fundamentals

PLURALSIGHT COURSES Business Personal **LIVE 2018**

Rust Fundamentals

★★★★★ By Dmitri Nesteruk

This course introduces Rust: a native code programming language with a focus on safety and correctness.

[Start a FREE 10-day trial](#)

Summary

```
// operators.rs
fn operators() {
    // arithmetic
    let mut a = 2+3*4; // 14
    println!("{}", a);
    a = a+1;
    a += 2; // a is now 17
    println!("{}", a);

    // remainder of /
    let a_cubed = 122;
    println!("{}", a_cubed);

    let b = 2.5;
    let b_cubed = f64::pow(b, 3);
    let b_to_pi = f64::pow(b, std::f64::consts::PI);
    println!("{}", b_cubed);

    // bitshift
    let c = 1; // 1
    let c_and = c & 1; // 1
    let c_or = c | 1; // 1
    let c_xor = c ^ 1; // 0
    let c_not = !c; // 0
}
```



Obrigado