Gabriel Gian

# A linguagem de programação Rust

Guaxupé, Minas Gerais

2021

Gabriel Gian

# A linguagem de programação Rust

Centro Universitário da Fundação Educacional Guaxupé
– UNIFEG

Guaxupé, Minas Gerais
2021

Gabriel Gian
A linguagem de programação Rust/ Gabriel Gian. – Guaxupé, Minas Gerais, 2021-
15p. : il. (algumas color.) ; 30 cm.

Trabalho – Centro Universitário da Fundação Educacional Guaxupé
– UNIFEG, 2021.

1. Linguagens de programação 2. Rust I. UNIFEG

# 1 Introdução

A linguagem de programação Rust tem obtido uma adoção crescente nos setores da programação de sistemas. Tendo sido criada e impulsionada pela Mozilla Foundation, a linguagem, atualmente na versão 1.50.0, tem recebido elogios e recomendações de diversos profissionais, além figurar no topo da lista de linguagens mais amadas pelo site Stack Overflow há mais de 5 anos.

# 2 Rust

## 2.1 História

A linguagem de programação Rust é uma linguagem multiparadigma, compilada, de tipagem estática e forte, criada como um projeto pessoal de Graydon Hoare, empregado da Mozilla. A fundação começou a apoiar o projeto em 2009, tendo também como colaboradores de desenvolvimento Dave Herman e Brendan Eich, e anuciando-o em 2010. O compilador original era escrito em OCaml, até que em 2011, o compilador auto-hospedado rustc conseguiu compilar-se. O rustc utiliza o LLVM como back-end. A linguagem foi lançada oficialmente numa versão pré-alfa em 2012. A primeira versão estável foi lançada em 15 de maio de 2015.

## 2.2 A linguagem

A linguagem tem como objetivo proporcionar uma melhor experiência para a programação de sistemas, sendo uma linguagem voltada à programação em níveis mais baixos, como C e C++, mas tendo a vantagem de proporcionar uma maior segurança para uma modalidade de programação tão melindrosa. A Rust inclui um gerenciador de dependências denominado Cargo, capaz não só de gerenciar dependências e bibliotecas, como compilar código Rust de maneira eficiente. O compilador do Cargo é notório por oferecer dicas de otimização e de correção de código em tempo de compilação, dado que determinado bloco de código esteja no modo *safe*. O Cargo também é capaz de gerar uma documentação do código que vai muito além de reles comentários. Há também uma extensa compatibilidade com IDEs diversas, graças ao Rust Language Server, fornecendo completação de código e mensagens de erro embutidas.

# Parte I

# Funcionalidades da linguagem

Sendo uma linguagem inspirada em C e C++, a Rust utiliza um método main como função principal. O método main é sempre o primeiro a ser chamado, como aconteceria em qualquer linguagem advinda da família da linguagem C. A linguagem Rust também faz uso de chaves para cercear escopos e blocos de código, assim como dos clássicos ponto e vírgulas ao final de cada instrução. As variáveis em Rust são, por padrão, imutáveis. Essa decisão tem origem nas inspirações em linguagens funcionais como Haskell, onde o uso da mutabilidade é desencorajado por possibilitar a ocorrência de efeitos colaterais em um determinado código, sendo necessário, para contornar essa função da linguagem, a declaração de uma variável mutável com a palavra-chave *mut*. Assim como as variáveis podem ser declaradas como mutáveis, outro desvio dos padrões da linguagem deve ser executado para que possa-se ter um controle maior do seu código como ocorreria em linguagens de programação de sistemas tradicionais. Por padrão, a Rust declara blocos de código *seguros*. Isto significa que certas funcionalidades para o acesso a regiões de memória e ponteiros, por exemplo, são administradas pela própria linguagem, com o uso de ferramentas como o *borrow checker*, responsável pelo gerenciamento da memória "emprestada"entre threads e objetos, uma funcionalidade ovacionada da linguagem. Um bloco de código inseguro é um bloco que abre mão de funcionalidades como esta e permitem ao programador o desenvolvimento mais específico e controlado, embora com menos seguranças, do seu software. Tal bloco pode ser delcarado com o uso da palavra-chave *unsafe*. Como isso pode acarretar problemas que não podem ser contornados pelos subterfúgios da linguagem, o compilador rustc do Cargo deixa de gerar mensagens de otimização e de correção para o desenvolvedor, estando ele ciente de que seu código vai além do que é assegurado pela linguagem. Essa flexibilidade é uma das maiores qualidades da linguagem Rust.

# Parte II

# Aplicações da linguagem

A linguagem tem sido adotada como substituta do C e do C++ para projetos de programação de sistemas diversos, seja a escrita de drivers e motores de renderização (grande parte do motor do Firefox já foi reescrita em Rust), como também no desenvolvimento para embarcados como o Raspberry Pi. Exemplos de aplicações incluem o Stratis, um gerenciador de arquivos para o Fedora Linux e o RHEL 8; o Redox, um microkernel; o Google Fuchsia, um sistema operacional; o utilitário exa, que funciona como substituto moderno ao famoso comando unix *ls*; a rede Tor, que está experimentando o uso de Rust devido às suas garantias de segurança; o Deno, um runtime de JavaScript desenvolvido pelo criador do Node.js; o Discord, serviço de bate-papo e chamadas de áudio e vídeo direcionado a gamers também utiliza Rust em seu back-end, além de também a utilizar na codificação em seus clientes do lado do usuário.

# 3 Referências bibliográficas


01. KLABNIK, Steve, NICHOLS, Carol. *The Rust Programming Language*. Segunda edição. São Francisco, CA. No Starch Press, Agosto de 2019.

02. FAQ – The Rust Project". Rust-lang.org. Archived from the original on 2016-06-09. Acessado em 27-06-2019.

03. Hoare, Graydon (7-07-2010). Project Servo (PDF). Mozilla Annual Summit 2010. Whistler, Canada. Acessado em 22 February 2017.

04. Hoare, Graydon (2010-10-02). "Rust Progress". Archived from the original on 2014-08-15. Acessado em 2010-10-30.

05. Hoare, Graydon (2011-04-20). "[rust-dev] stage1/rustc builds". Acessado em 2011-04-20.

06. Yegulalp, Serdar. "Rust's Redox OS could show Linux a few new tricks". infoworld. Acessado em 21-03-2016.

07. Sei, Mark (10-10-2018). "Fedora 29 new features: Startis now officially in Fedora". Marksei, Weekly sysadmin pills. Acessado em 2019-05-13.

08. "RHEL 8: Chapter 8. Managing layered local storage with Stratis". 10-10-2018.

09. Nichols, Shaun (27-06-2018). "Microsoft's next trick? Kicking things out of the cloud to Azure IoT Edge". The Register. Acessado em 2019-09-27.

10. Balbaert, Ivo (27-05-2015). Rust Essentials. Packt Publishing. p. 6. ISBN 978-1785285769. Acessado em 21-03-2016.

11. Frank, Denis (5-12-2013). "Using HyperLogLog to Detect Malware Faster Than Ever". OpenDNS Security Labs. Acessado em 19-03-2016.

12. Denis, Frank (4-10-2013). "ZeroMQ: Helping us Block Malicious Domains in Real Time". OpenDNS Security Labs. Acessado em 19-03-2016.

13. Hahn, Sebastian (2017-03-31). "[tor-dev] Tor in a safer language: Network team update from Amsterdam". Acessado em 2017-04-01.

14. asn (2017-07-05). "The Wilmington Watch: A Tor Network Team Hackfest". Tor Blog. Acessado em 2018-01-03.

15. Garbutt, James (27-01-2019). "First thoughts on Deno, the JavaScript/TypeScript run-time". 43081j.com. Acessado em 2019-09-27.

16. Howarth, Jesse (2020-02-04). "Why Discord is switching from Go to Rust". Acessado em 2020-04-14.