O'REILLY®

# Programming Rust

Rust编程（影印版）

Jim Blandy, Jason Orendorff 著

東南大學出版社

6.0

# Rust编程（影印版）

## Programming Rust

Jim Blandy, Jason Orendorff 著

Beijing · Boston · Farnham · Sebastopol · Tokyo O'REILLY®

O'Reilly Media, Inc.授权东南大学出版社出版

南京 东南大学出版社

**图书在版编目(CIP)数据**

Rust 编程：英文/(美)吉姆·布兰迪(Jim Blandy),(美)詹森·奥兰多夫(Jason Orendorff)著. —影印本. —南京：东南大学出版社，2018.7

书名原文：Programming Rust

ISBN 978-7-5641-7733-1

Ⅰ. ①R… Ⅱ. ①吉… ②詹… Ⅲ. ①程序语言-程序设计-英文 Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字(2018)第 100035 号
图字：10-2018-111 号

Rust 编程(影印版)

出版发行：东南大学出版社
地　　址：南京四牌楼 2 号　　邮编：210096
出 版 人：江建中
网　　址：http://www.seupress.com
电子邮件：press@seupress.com
印　　刷：常州市武进第三印刷有限公司
开　　本：787 毫米×980 毫米　16 开本
印　　张：39
字　　数：764 千字
版　　次：2018 年 7 月第 1 版
印　　次：2018 年 7 月第 1 次印刷
书　　号：ISBN 978-7-5641-7733-1
定　　价：118.00 元

本社图书若有印装质量问题，请直接与营销部联系。电话(传真)：025-83791830

# Preface

Rust is a language for systems programming.

This bears some explanation these days, as systems programming is unfamiliar to most working programmers. Yet it underlies everything we do.

You close your laptop. The operating system detects this, suspends all the running programs, turns off the screen, and puts the computer to sleep. Later, you open the laptop: the screen and other components are powered up again, and each program is able to pick up where it left off. We take this for granted. But systems programmers wrote a lot of code to make that happen.

Systems programming is for:

- Operating systems
- Device drivers of all kinds
- Filesystems
- Databases
- Code that runs in very cheap devices, or devices that must be extremely reliable
- Cryptography
- Media codecs (software for reading and writing audio, video, and image files)
- Media processing (for example, speech recognition or photo editing software)
- Memory management (for example, implementing a garbage collector)
- Text rendering (the conversion of text and fonts into pixels)
- Implementing higher-level programming languages (like JavaScript and Python)
- Networking
- Virtualization and software containers

- Scientific simulations
- Games

In short, systems programming is *resource-constrained* programming. It is programming when every byte and every CPU cycle counts.

The amount of systems code involved in supporting a basic app is staggering.

This book will not teach you systems programming. In fact, this book covers many details of memory management that might seem unnecessarily abstruse at first, if you haven't already done some systems programming on your own. But if you are a seasoned systems programmer, you'll find that Rust is something exceptional: a new tool that eliminates major, well-understood problems that have plagued a whole industry for decades.

## Who Should Read This Book

If you're already a systems programmer, and you're ready for an alternative to C++, this book is for you. If you're an experienced developer in any programming language, whether that's C#, Java, Python, JavaScript, or something else, this book is for you too.

However, you don't just need to learn Rust. To get the most out of the language, you also need to gain some experience with systems programming. We recommend reading this book while also implementing some systems programming side projects in Rust. Build something you've never built before, something that takes advantage of Rust's speed, concurrency, and safety. The list of topics at the beginning of this preface should give you some ideas.

## Why We Wrote This Book

We set out to write the book we wished we had when we started learning Rust. Our goal was to tackle the big, new concepts in Rust up front and head-on, presenting them clearly and in depth so as to minimize learning by trial and error.

## Navigating This Book

The first two chapters of this book introduce Rust and provide a brief tour before we move on to the fundamental data types in Chapter 3. Chapters 4 and 5 address the core concepts of ownership and references. We recommend reading these first five chapters through in order.

Chapters 6 through 10 cover the basics of the language: expressions (Chapter 6), error handling (Chapter 7), crates and modules (Chapter 8), structs (Chapter 9), and enums and patterns (Chapter 10). It's all right to skim a little here, but don't skip the chapter on error handling. Trust us.

Chapter 11 covers traits and generics, the last two big concepts you need to know. Traits are like interfaces in Java or C#. They're also the main way Rust supports integrating your types into the language itself. Chapter 12 shows how traits support operator overloading, and Chapter 13 covers many more utility traits.

Understanding traits and generics unlocks the rest of the book. Closures and iterators, two key power tools that you won't want to miss, are covered in Chapters 14 and 15, respectively. You can read the remaining chapters in any order, or just dip into them as needed. They cover the rest of the language: collections (Chapter 16), strings and text (Chapter 17), input and output (Chapter 18), concurrency (Chapter 19), macros (Chapter 20), and unsafe code (Chapter 21).

## Conventions Used in This Book

The following typographical conventions are used in this book:

*Italic*

Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`

Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

**`Constant width bold`**

Shows commands or other text that should be typed literally by the user.

*`Constant width italic`*

Shows text that should be replaced with user-supplied values or by values determined by context.

This icon signifies a tip or suggestion.

This icon signifies a general note.

This icon indicates a warning or caution.

## Using Code Examples

Supplemental material (code examples, exercises, etc.) is available for download at *https://github.com/oreillymedia/programming_rust*.

This book is here to help you get your job done. In general, if example code is offered with this book, you may use it in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Programming Rust* by Jim Blandy and Jason Orendorff (O'Reilly). Copyright 2018 Jim Blandy and Jason Orendorff, 978-1-491-92728-1."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

## O'Reilly Safari

Safari (formerly Safari Books Online) is a membership-based training and reference platform for enterprise, government, educators, and individuals.

Members have access to thousands of books, training videos, Learning Paths, interactive tutorials, and curated playlists from over 250 publishers, including O'Reilly Media, Harvard Business Review, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Adobe, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, and Course Technology, among others.

For more information, please visit *http://oreilly.com/safari*.

## How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at *http://bit.ly/programming-rust*.

To comment or ask technical questions about this book, send email to *bookquestions@oreilly.com*.

For more information about our books, courses, conferences, and news, see our website at *http://www.oreilly.com*.

Find us on Facebook: *http://facebook.com/oreilly*

Follow us on Twitter: *http://twitter.com/oreillymedia*

Watch us on YouTube: *http://www.youtube.com/oreillymedia*

## Acknowledgments

The book you are holding has benefited greatly from the attention of our official technical reviewers: Brian Anderson, Matt Brubeck, J. David Eisenberg, and Jack Moffitt.

Many other unofficial reviewers read early drafts and provided invaluable feedback. We would like to thank Eddy Bruel, Nick Fitzgerald, Michael Kelly, Jeffrey Lim, Jakob Olesen, Gian-Carlo Pascutto, Larry Rabinowitz, Jaroslav Šnajdr, and Joe Walker for their thoughtful comments. Jeff Walden and Nicolas Pierron were especially generous with their time, reviewing almost the entire book. Like any programming venture, a programming book thrives on quality bug reports. Thank you.

Mozilla was extremely accommodating of our work on this project, even though it fell outside our official responsibilities and competed with them for our attention. We are grateful to our managers, Dave Camp, Naveed Ihsanullah, Tom Tromey, and Joe Walker, for their support. They take a long view of what Mozilla is about; we hope these results justify the faith they placed in us.

We would also like to express our appreciation for everyone at O'Reilly who helped bring this project to fruition, especially our editors Brian MacDonald and Jeff Bleiel.

Most of all, our heartfelt thanks to our wives and children for their unwavering love, enthusiasm, and patience.

# Table of Contents