Community Experience Distilled

# Rust Essentials

Discover how to use Rust to write fast, secure, and concurrent systems and applications

Ivo Balbaert

[PACKT] PUBLISHING open source* community experience distilled

# Rust Essentials

Discover how to use Rust to write fast, secure, and concurrent systems and applications

**Ivo Balbaert**


BIRMINGHAM - MUMBAI

# Rust Essentials



First published: May 2015

Production reference: 1220515

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-78528-576-9

www.packtpub.com

# Credits

**Author**
Ivo Balbaert

**Reviewers**
Alfie John
Anthony Miyaguchi
Bharadwaj Srigiriraju
Syed Omar Faruk Towaha
Tony Zou

**Commissioning Editor**
Akram Hussain

**Acquisition Editor**
Rebecca Youé

**Content Development Editor**
Manasi Pandire

**Technical Editors**
Tanmayee Patil
Shiny Poojary
Mohita Vyas

**Copy Editor**
Jasmine Nadar

**Project Coordinator**
Suzanne Coutinho

**Proofreaders**
Stephen Copestake
Safis Editing

**Indexer**
Tejal Soni

**Production Coordinator**
Aparna Bhagat

**Cover Work**
Aparna Bhagat

# About the Author

**Ivo Balbaert** is currently a lecturer of (web) programming and databases at CVO Antwerpen (`www.cvoantwerpen.be`), a community college in Belgium. He received a PhD in applied physics from the University of Antwerp in 1986. He worked in the software industry as a developer and consultant for several companies for 20 years and as a project manager at the University Hospital of Antwerp for 10 years. From 2000 onwards, he switched to partly teaching and partly developing software (KHM Mechelen, CVO Antwerp).

He wrote an introductory book in Dutch about developing in Ruby and Rails, *Programmeren met Ruby en Rails*, *Van Duuren Media*.

In 2012, he authored a book on the Go programming language, *The Way To Go*, *iUniverse*.

In 2013, in collaboration with Dzenan Ridzanovic, he wrote *Learning Dart* and *Dart Cookbook*, both by Packt Publishing.

In 2014, he wrote *Getting Started with Julia*, *Packt Publishing*.

---

I would like to thank the technical reviewers, especially Brian Anderson, Alfie John, and Anne-Marie Mission, for their many useful remarks that improved the text, and my wife, Christiane, for her support.

---

# About the Reviewers

**Anthony Miyaguchi** is a computer science and engineering student at UCLA. He is active in the open source community and has worked on a variety of different projects, from embedded programming to web technologies. If he finds free time, he would like to make a dent in his collection of books.

**Bharadwaj Srigiriraju** is a computer science graduate from IIITDM, Jabalpur, who now works as a software developer at Chumbak, Bangalore. He is a technology enthusiast who loves to develop web apps and hack on (shiny) new technologies. He specializes in Python and firmly believes that Rust will replace C very soon. You can reach him at `krishna.bharadwaj6@gmail.com` or visit his GitHub to know more `github.com/bharadwaj6`.

**Syed Omar Faruk Towaha** is a programmer and physicist from Shahjalal University of Science and Technology, Sylhet, Bangladesh. He is involved with the Rust development team and writes and reviews books on several programming languages. He is an Oracle Certified Professional (OCP) developer and loves open source technology. He has been working with several science projects and some research projects at his university as well as in international laboratories. He enjoys designing algorithms and circuit theory. He volunteers at Mozilla by arranging events as a Mozilla representative (`http://reps.mozilla.org/`).

He is the president of a famous astronomical organization, CAM-SUST (`http://camsust.org/`). He loves working in teams and being associated with interesting projects.

His recent books include *How You Should Design Algorithms*, *Easy Circuits for Kids*, *Wonder in Quantum Physics*, and *Fundamentals of Ruby*.

You can contact him at `soft@hotmail.co.uk`. To find out more details about him, go to `http://towaha.me/`.

---

I would like to thank the author of this wonderful book and also Suzanne Coutinho and Nikita Michael for their help. This is a pretty good book on Rust, and I will recommend it to anyone who wants to learn Rust. I hope that the author writes more books on Rust, especially by developing games and some exciting things to let the common people know how rich the rust language is.

---

**Tony Zou** is currently pursuing his undergraduate studies at the University of Waterloo. He has been programming for 4 years and has worked on a few projects. He enjoys competitive programming and working with exciting new languages such as Rust.

# www.PacktPub.com

## Support files, eBooks, discount offers, and more

For support files and downloads related to your book, please visit `www.PacktPub.com`.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at `www.PacktPub.com` and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at `service@packtpub.com` for more details.

At `www.PacktPub.com`, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.


`https://www2.packtpub.com/books/subscription/packtlib`

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can search, access, and read Packt's entire library of books.

## Why subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print, and bookmark content
- On demand and accessible via a web browser

## Free access for Packt account holders

If you have an account with Packt at `www.PacktPub.com`, you can use this to access PacktLib today and view 9 entirely free books. Simply use your login credentials for immediate access.

# Table of Contents

**Preface** **v**

**Chapter 1: Starting with Rust** **1**

**The advantages of Rust** **2**
**The trifecta of Rust – safety, speed, and concurrency** **3**
Comparison with other languages 5
**Using Rust** **5**
Servo 6
**Installing Rust** **7**
**The Rust compiler – rustc** **7**
**Our first program** **8**
**Working with Cargo** **10**
**The developer tools** **13**
Using Sublime Text 14
Other tools 15
**Summary** **16**

**Chapter 2: Using Variables and Types** **17**

**Comments** **17**
**Global constants** **18**
Printing with string interpolation 20
**Values and primitive types** **22**
Consulting Rust documentation 23
**Binding variables to values** **23**
Mutable and immutable variables 25
**Scope of a variable and shadowing** **26**
**Type checking and conversions** **27**
Aliasing 28
**Expressions** **29**

**The stack and the heap** **30**
**Summary** **34**
**Chapter 3: Using Functions and Control Structures** **35**
**Branching on a condition** **35**
**Looping** **37**
**Functions** **39**
Documenting a function 41
**Attributes** **42**
Conditional compilation 43
**Testing** **43**
Testing with cargo 45
**Summary** **45**
**Chapter 4: Structuring Data and Matching Patterns** **47**
**Strings** **48**
**Arrays, vectors, and slices** **50**
Vectors 52
Slices 53
Strings and arrays 54
**Tuples** **55**
**Structs** **56**
**Enums** **58**
Result and Option 59
**Getting input from the console** **60**
**Matching patterns** **62**
**Summary** **65**
**Chapter 5: Generalizing Code with Higher-order Functions and Parametrization** **67**
**Higher-order functions and closures** **67**
**Iterators** **70**
**Consumers and adapters** **72**
**Generic data structures and functions** **74**
**Error handling** **77**
Panics 78
Failures 78
**Methods on structs** **79**
**Traits** **82**
**Using trait constraints** **84**
**Built-in traits and operator overloading** **87**
**Summary** **87**

**Chapter 6: Pointers and Memory Safety** **89**
**Pointers and references** **89**
The stack and the heap 89
Lifetimes 90
Copying values and the Copy trait 93
Pointers 95
References 96
Using ref in a match 98
**Ownership and borrowing** **99**
**Boxes** **102**
**Reference counting** **105**
**An overview of pointers** **107**
**Summary** **107**
**Chapter 7: Organizing Code and Macros** **109**
**Modules and crates** **109**
Building crates 110
Defining a module 111
The visibility of items 112
Importing modules and file hierarchy 114
Importing external crates 115
Exporting a public interface 117
Adding external crates to a project 118
The test module 119
**Macros** **121**
Why do we use macros? 121
Developing macros 122
Repetition 124
Creating a new function 124
Using macros from crates 127
**Summary** **127**
**Chapter 8: Concurrency and Parallelism** **129**
**Concurrency and threads** **129**
Creating threads 130
Starting a number of threads 131
Panicking threads 133
Thread-safety 134
**The shared mutable state** **135**
The Sync trait 137
**Communication through channels** **138**
Sending and receiving data 139
Synchronous and asynchronous communication 141
**Summary** **142**

**Chapter 9: Programming at the Boundaries** **143**
**Program arguments** **143**
**Unsafe code** **145**
**Raw pointers** **146**
**Interfacing with C** **147**
Using a C library 149
Inlining assembly code 150
**Calling Rust from other languages** **151**
**Summary** **152**
**Appendix: Exploring Further** **153**
**Stability of Rust and the standard library** **153**
**The ecosystem of crates** **153**
**Other resources for learning Rust** **154**
Files and databases 154
Graphics and games 155
Web development 155
**Index** **157**

# Preface

Rust is the new open source and compiled programming language that finally promises software developers the utmost safety—not only type safety but memory safety as well. The compiler carefully checks all uses of variables and pointers so that common problems from C / C++ and other languages, such as pointers to wrong memory locations or null references, are a thing of the past. Potential problems are detected at compilation time so that Rust programs execute at speeds that are comparable with their C++ counterparts.

Rust runs with a very light runtime, which does not perform garbage collection. Again the compiler takes care of generating code that frees all resources at the right time. This means Rust can run in very constrained environments, such as embedded or real-time systems. When executing code concurrently no data races can occur, because the compiler imposes the same memory safety restrictions as when the code executes consecutively.

From the preceding description, it is clear that Rust is applicable in all use cases where C and C++ were the preferred languages until now and that it will do a better job.

Rust is a very rich language; it has concepts (such as immutability by default) and constructs (such as traits) that enable developers to write code in a highly functional and object-oriented style.

The original goal of Rust was to serve as the language to write a new safe browser engine that was devoid of the many security flaws that plague existing browsers. This is the Servo project from Mozilla Research.

The goal of this book is to give you a firm foundation so that you can start to develop in Rust. Throughout the book, we emphasize the three pillars of Rust: safety, performance, and concurrency. We discuss the areas and the reasons why Rust differs from other programming languages. The code examples are not chosen ad hoc, but they are oriented as part of an ongoing project to build a game so that there is a sense of cohesion and evolution in the examples.

Throughout the book, I will urge you to learn by doing things; you can follow along by typing in the code, making the requested modifications, compiling, testing, and working out the exercises.

# What this book covers

*Chapter 1*, *Starting with Rust*, discusses the main reasons that led to the development of Rust. We compare Rust with other languages and indicate the areas in which it is most appropriate. Then, we guide you through the installation of all the necessary components for Rust's development environment.

*Chapter 2*, *Using Variables and Types*, looks at the basic structure of a Rust program. We discuss the primitive types, how to declare variables and whether they have to be typed, and the scope of variables. Immutability, which is one of the key cornerstones of Rust's safety strategy, is also illustrated. Then, we look at basic operations, how to do formatted printing, and the important difference between expressions and statements.

*Chapter 3*, *Using Functions and Control Structures*, shows you how to define functions and the different ways to influence program execution flow in Rust.

*Chapter 4*, *Structuring Data and Matching Patterns*, discusses the basic data types for programming, such as strings, vectors, slices, tuples, and enums. Then, we show you the powerful pattern matching that is possible in Rust and how values are extracted by de-structuring patterns.

*Chapter 5*, *Generalizing Code with Higher-order Functions and Parametrization*, explores the functional and object-oriented features of Rust. You will see how data structures and functions can be defined in a generic way and how traits can be used to define behavior.

*Chapter 6*, *Pointers and Memory Safety*, exposes the borrow checker, which is Rust's mechanism to ensure that only memory safe operations can occur. We discuss different kinds of pointers as well as how to handle runtime errors.

*Chapter 7*, *Organizing Code and Macros*, discusses the bigger code-organizing structures in Rust. We will also touch upon how to build macros in order to generate code and save time and effort.

*Chapter 8*, *Concurrency and Parallelism*, delves into Rust's concurrency model with its basic concepts of threads and channels. We also discuss a safe strategy for working with shared mutable data.

*Chapter 9*, *Programming at the Boundaries*, looks at how Rust can take command-line parameters to process. Then, we go on to look at situations where we have to leave the safety boundaries, such as when we interface with C or use raw pointers, and how Rust minimizes potential dangers when we do so.

*Appendix*, *Exploring Further*, talks about the Rust ecosystem and where the reader can find more information about certain topics, such as working with files, databases, games, and web development.

# What you need for this book

To run the code examples in the book, you will need the Rust system for your computer, which can be downloaded from `http://www.rust-lang.org/install.html`. This also contains the Cargo project and the package manager. To work more comfortably with the Rust code, a development environment such as Sublime Text can also be of use. *Chapter 1*, *Starting with Rust*, contains detailed instructions on how to set up your Rust environment.

# Who this book is for

This book is intended for developers who have some programming experience in C/C++, Java/C#, Python, Ruby, Dart, or a similar language and a basic knowledge of general programming concepts. It will get you up and running quickly, giving you all you need to start building your own Rust projects.

# Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "We can see that `main()` is a function declaration because it is preceded by the keyword `fn`, which is short and elegant like most Rust keywords."

A block of code is set as follows:

```
let tricks = 10;
let reftricks = &mut tricks;
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
let n1 = {
    let a = 2;
    let b = 5;
    a + b   // <-- no semicolon!
};
```

Any command-line input or output is written as follows:

```
[root]
name = "welcomec"
version = "0.0.1"
```

**New terms** and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "When working with Rust code, select **Tools** | **Build System** | **Rust**."

Warnings or important notes appear in a box like this.

Tips and tricks appear like this.

# Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to `feedback@packtpub.com`, and mention the book title via the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on `www.packtpub.com/authors`.

# Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

# Downloading the example code

You can download the example code files from your account at `http://www.packtpub.com` for all the Packt Publishing books you have purchased. If you purchased this book elsewhere, you can visit `http://www.packtpub.com/support` and register to have the files e-mailed directly to you.

# Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you could report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/submit-errata`, selecting your book, clicking on the **Errata Submission Form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded to our website or added to any list of existing errata under the Errata section of that title.

To view the previously submitted errata, go to `https://www.packtpub.com/books/content/support` and enter the name of the book in the search field. The required information will appear under the **Errata** section.

# Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

# Questions

You can contact us at `questions@packtpub.com` if you are having a problem with any aspect of the book, and we will do our best to address it.