HZ BOOKS
华章科技

Rust 连续多年荣膺 Stack Overflow 网站最受程序员欢迎的编程语言。

深入浅出Rust

DIVE INTO RUST

范长春 ◎ 著

本书使用通俗易懂的语言，辅以大量的代码示例，高屋建瓴地总结阐释了 Rust 的主要概念以及使用方法，并对背后的设计思路和原理做了深入浅出的剖析，全面深入地提炼了 Rust 的设计精华。

机械工业出版社
China Machine Press

深入浅出Rust

DIVE INTO RUST

范长春◎著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

深入浅出 Rust/ 范长春著 . —北京：机械工业出版社，2018.8

ISBN 978-7-111-60642-0

I. 深… II. 范… III. 程序语言 - 程序设计 IV. TP312

中国版本图书馆 CIP 数据核字（2018）第 181199 号

深入浅出 Rust

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）

责任编辑：陈佳媛　　责任校对：李秋荣

印　　刷：北京诚信伟业印刷有限公司　　版　　次：2018 年 8 月第 1 版第 1 次印刷

开　　本：186mm × 240mm　1/16　　印　　张：25.25

书　　号：ISBN 978-7-111-60642-0　　定　　价：89.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：（010）88379426　88361066　　投稿热线：（010）88379604

购书热线：（010）68326294　88379649　68995259　　读者信箱：hzit@hzbook.com

Preface 前　言

A language that doesn't affect the way you think about programming is not worth knowing.

——Alan Perlis

Rust 简介

Rust 是一门新的编程语言。

我想，大部分读者看到本书，估计都会不约而同地想到同样的问题：现存的编程语言已经多得数不清了，再发明一种新的编程语言有何意义？难道现存的那么多编程语言还不够用吗，发明一种新的编程语言能解决什么新问题？

俗话说，工欲善其事，必先利其器。在程序员平时最常用的工具排行榜中，编程语言当仁不让的是最重要的“器”。编程语言不仅是给程序设计者使用的工具，反过来，它也深刻地影响了设计者本身的思维方式和开发习惯。

卓越的编程语言，可以将优秀的设计、先进的思想、成功的经验，自然而然地融入其中，使更多的使用者开阔眼界、拓展思路，受益无穷。

A programming language is a tool that has profound influence on our thinking habits.

——Edsger Dijkstra

所以说关于这个问题，我认为，如果与现有的各种语言相比，新设计的语言有所进步、有所发展、有所创新，那么它的出现就很有意义。

最近这些年，的确涌现出了一大批编程语言，可以说是百花争艳、繁华似锦。但是在表面的繁荣之下，我们是否可以自满地说，编程语言的设计和发展已经基本成熟、趋于完美了

呢？恐怕不尽然吧！

那些优秀的编程语言中，不少都有自己的“绝活”。有的性能非常高，有的表达力非常强，有的擅长组织大型程序，有的适合小巧的脚本，有的专注于并发，有的偏重于科学计算，等等，不一而足。即便如此，新兴的 Rust 语言面市后依旧展现出了它的独特魅力，矫矫不群，非常值得大家关注。

作为多年来鲜有的新一代系统编程语言，它的设计准则是“安全，并发，实用”。Rust 的设计者是这样定位这门语言的：

Rust is a system's programming language that runs blazingly fast, prevents segfaults, and guarantees thread safety.

安全

是的，安全性很重要。Rust 最重要的特点就是可以提供内存安全保证，而且没有额外的性能损失。

在传统的系统级编程语言（C/C++）的开发过程中，经常出现因各种内存错误引起的崩溃或 bug。比如空指针、野指针、内存泄漏、内存越界、段错误、数据竞争、迭代器失效等，血泪斑斑，数不胜数。这些问题不仅在教科书中被无数次提起，而且在实践中也极其常见。因此，各种高手辛苦地总结了大量的编程经验，许多代码检查和调试工具被开发出来，各种代码开发流程和规范被制定出来，无数人呕心沥血就是为了系统性地防止各类 bug 的出现。尽管如此，我们依然无法彻底解决这些问题。

教科书解决不了问题，因为教育不是强制性的；静态代码检查工具解决不了问题，因为传统的 C/C++ 对静态代码检查不友好，永远只能查出一部分问题；软件工程解决不了问题，因为规范依赖于执行者的素质，任何人都会犯错误。事后 debug 也不是办法，解决 bug 的代价更高。

鉴于手动内存管理非常容易出问题，因此先辈们发明了一种自动垃圾回收的机制（Garbage Collection），故而程序员在绝大多数情况下不用再操心内存释放的问题。新发明的绝大多数编程语言都使用了基于各种高级算法的自动垃圾回收机制，因为它确实方便，解放了程序员的大脑，使大家能更专注于业务逻辑的部分。但是到目前为止，不管使用哪种算法的 GC 系统，在性能上都要付出比较大的代价。要么需要较大的运行时占用较大内存，要么需要暂停整个程序，要么具备不确定性的时延。当然，在现实的许多业务场景中，这点开销是微不足道的，因此问题不大。可是如果在性能敏感的领域，这是完全不可接受的。

很遗憾，到目前为止，在系统级编程语言中，我们依然被各种内存安全问题所困扰。这些年来，许多新的语言特性被发明出来，许多优秀的编程范式被总结出来，许多高质量的代码库被开发出来。但是内存安全问题依然像一个幽灵一样，一直徘徊在众多程序员的头顶，无法摆脱。再多的努力，也只能减少它出现的机会，很难保证完整地解决掉这一类错误。

Rust 对自己的定位是接近芯片硬件的系统级编程语言，因此，它不可能选择使用自动垃圾回收的机制来解决问题。事实证明，要想解决内存安全问题，小修小补是不够的，必须搞清楚导致内存错误的根本原因，从源头上解决。Rust 就是为此而生的。Rust 语言是可以保证内存安全的系统级编程语言。这是它的独特的优势。本书将用大量的篇幅详细介绍“内存安全”。

并发

在计算机单核性能越来越接近瓶颈的今天，多核并行成了提高软件执行效率的发展趋势。一些编程语言已经开始从语言层面支持并发编程，把“并发”的概念植入到了编程语言的血液中。然而，在传统的系统级编程语言中，并行代码很容易出错，而且有些问题很难复现，难以发现和解决问题，debug 的成本非常高。线程安全问题一直以来都是非常令人头痛的问题。

Rust 当然也不会在这一重要领域落伍，它也非常好地支持了并发编程。更重要的是，在强大的内存安全特性的支持下，Rust 一举解决了并发条件下的数据竞争（Data Race）问题。它从编译阶段就将数据竞争解决在了萌芽状态，保障了线程安全。

Rust 在并发方面还具有相当不错的可扩展性。所有跟线程安全相关的特性，都不是在编译器中写死的。用户可以用库的形式实现各种高效且安全的并发编程模型，进而充分利用多核时代的硬件性能。

实用

Rust 并不只是实验室中的研究型产品，它的目标是解决目前软件行业中实实在在的各种问题。它的实用性体现在方方面面。

Rust 编译器的后端是基于著名的 LLVM 完成机器码生成和优化的，它只需要一个非常小巧的运行时即可工作，执行效率上可与 C 语言相媲美，具备很好的跨平台特性。

Rust 摈弃了手动内存管理带来的各种不安全的弊端，同时也避免了自动垃圾回收带来的效率损失和不可控性。在绝大部分情况下，保持了“无额外性能损失”的抽象能力。

Rust 具备比较强大的类型系统，借鉴了许多现代编程语言的历史经验，包含了众多方便的语法特性。其中包括代数类型系统、模式匹配、闭包、生成器、类型推断、泛型、与 C 库

ABI 兼容、宏、模块管理机制、内置开源库发布和管理机制、支持多种编程范式等。它吸收了许多其他语言中优秀的抽象能力，海纳百川，兼容并蓄。在不影响安全和效率的情况下，拥有不俗的抽象表达力。

有意思的地方是，在程序语言设计领域，按照传统思路，有些设计目标是互相冲突的。而 Rust 的优异之处在于，它能游刃有余地游走在各种设计目标之间，扬长避短，保持良好的妥协和平衡。

本书结构

本书将详细描述 Rust 语言的基本语法，穿插讲解一部分高级使用技巧，并尽量以更容易理解的方式向读者解释其背后的设计思想。语法只是基础，并非本书的重点，笔者更希望读者能理解到这些语法设计背后的理念，读完本书之后，可以从中受到一点启发，对程序语言有更多的认识，从而对编程本身有更深的理解。

Learning a language that is significantly different than you are used to is certainly tough at first, but it's a great way to expand your horizons a bit.

在本书中，笔者尽量避免要求读者有很多的基础知识。当然，如果读者对其他的一种或多种编程语言有所了解更佳，其中包括 C/C++ 的基础知识、内存错误、手动内存管理、自动垃圾回收、多线程并发和同步、操作系统相关的基础概念等。

本书共分为五个部分。

第一部分介绍 Rust 基本语法。因为对任何程序设计语言来说，语法都是基础，学习这部分是理解其他部分的前提。

第二部分介绍属于 Rust 独一无二的内存管理方式。它设计了一组全新的机制，既保证了安全性，又保持了强大的内存布局控制力，而且没有额外性能损失。这部分是本书的重点和核心所在，是 Rust 语言的思想内核精髓之处。

第三部分介绍 Rust 的抽象表达能力。它支持多种编程范式，以及较为强大的抽象表达能力。

第四部分介绍并发模型。在目前这个阶段，对并行编程的支持是新一代编程语言无法绕过的重要话题。Rust 也吸收了业界最新的发展成果，对并发有良好支持。

第五部分介绍一些实用设施。Rust 语言有许多创新，但它绝不是高高在上、孤芳自赏的类型。设计者们在设计过程中充分考虑了语言的工程实用性。众多在其他语言中被证明过的

优秀实践被吸收了进来，有利于提升实际工作效率。

为了内容的完整性，本书并没有严格按照知识点顺序组织内容，少数地方会直接使用后续章节中的知识点。笔者相信对读者来说，这不是一个很大的障碍，各位读者在碰到这种情况的时候，可以自行前后参照来理解。

总结和勘误

在计算机程序设计语言的领域中，一代又一代的语言潮起潮落，其兴起和衰落的节奏往往并非取决于技术本身的发展。对于 Rust 这门新出现的语言来说，以后究竟会有多大的影响，是否会成为取代某种语言的“新时代的宠儿”，实在难以预测，而且毫无必要预测。

笔者认为，Rust 语言是最近若干年内系统级编程语言领域的集大成者之一。不论其最终发展如何，它的许多设计思想和令人惊叹的特性都值得大家学习。在本人的学习过程中，也时常为某些精彩的设计发出由衷的赞叹。

Rust 语言是一门优秀的语言，同时也是门槛较高的一门语言，要完全掌握它不是一件很容易的事。因此，笔者并不希望将本书写成语言特性的逐一简单罗列，而更希望向读者解释清楚这些语言特性背后的设计思想。

所幸的是，Rust 语言是完全开源的，不仅代码是开源的，而且整个设计过程、思辩讨论都是对社区完全开放的。它的许多非常有价值的学习资料，如同星星点点，散落在各个地方，包括官方文档、邮件列表、讨论组、GitHub、个人博客等。在学习和写作的过程中，能有幸一窥新语言创造者们的心路历程，也是难得的机缘。

要想把 Rust 语言的方方面面讲好、讲透，实在是一个无比繁重的任务。动笔之际，方知“看人挑担不吃力，自家挑担压断脊”，诚惶诚恐，战战兢兢，生怕有误人子弟之嫌。笔者水平有限，如有错漏，在所难免，欢迎读者批评指正。笔者将会在 GitHub 上发布最新的勘误列表，网址为 https://github.com/F001/rust_book_feedback。读者可以在这个项目中新建 bug 提交问题，也可以通过邮件（rust-lang@qq.com）与笔者联系。同时也欢迎读者关注微信公众号：Rust 编程，后面还会发布更多关于 Rust 的文章。

致谢

感谢 Rust 设计组，为软件开发行业创造了一份宝贵的财富。

感谢我所在的公司 synopsys 给予的大力支持。

感谢梁自泽导师对我的培养。

感谢林春晓博士拨冗为本书做了最后一轮审校。

感谢妻子的包容和呵护，否则本书不可能面世。

感谢杨绣国编辑细致的工作。

感谢各位老师、同学和同事对我的支持，正是因为你们的帮助，才使我技术水平更上一层楼。

范长春（F001）

中国，武汉，2018 年 3 月

Contents 目　录

PART1 · 第一部分

基础知识

在这一部分中，我们将对Rust语言的主要语法特性做一个循序渐进的介绍。Rust语言的基本语法特性并不复杂，它也并没有贪多求全地堆砌大量华而不实的语法特性。相反，它在吸收各种优秀语法规则的同时也做了裁剪，去芜存菁，张弛有度。

第 1 章 · CHAPTER1

与君初相见

Rust 编程语言的官方网站是 https://www.rust-lang.org/。在官网主页上，我们可以看到，在最显眼的位置，写着 Rust 语言最重要的特点：

Rust is a systems programming language that runs blazingly fast, prevents segfaults, and guarantees thread safety.

Rust 语言是一门系统编程语言，它有三大特点：**运行快、防止段错误、保证线程安全**。

系统级编程是相对于应用级编程而言。一般来说，系统级编程意味着更底层的位置，它更接近于硬件层次，并为上层的应用软件提供支持。系统级编程语言一般具有以下特点：

- 可以在资源非常受限的环境下执行；
- 运行时开销很小，非常高效；
- 很小的运行库，甚至于没有；
- 可以允许直接的内存操作。

目前，C 和 C++ 应该是业界最流行的系统编程语言。Rust 的定位与它们类似，但是增加了安全性。C 和 C++ 都是编译型语言，无须规模庞大的运行时（runtime）支持，也没有自动内存回收（Garbage Collection）机制。

本章主要对 Rust 做一个简单的介绍，准备好一些基本概念以及开发环境。

1.1 版本和发布策略

Rust 编程语言是开源的，编译器的源码位于 https://github.com/rust-lang/rust 项目中，语言设计和相关讨论位于 https://github.com/rust-lang/rfcs 项目中。对于想深入研究这门语言的读者来说，这是一个非常好的消息，大家可以通过研读开放的源代码和技术文档了解到很多