The Husky Programming Language

by Xiyu Zhai

This version of the text assumes you're using Rust 1.62 (released 2022-06-30) or later. See the "Installation" section of Chapter 1 to install or update Rust.

The HTML format is available online at https://doc.rust-lang.org/stable/book/ and offline with installations of Rust made with `rustup`; run `rustup docs --book` to open.

Several community translations are also available.

This text is available in paperback and ebook format from No Starch Press.

Foreword

Husky is a general term for a dog used in the polar regions, primarily and specifically for work as sled dogs. It refers to a traditional northern type, notable for its cold-weather tolerance and overall hardiness. Modern racing huskies that maintain arctic breed traits (also known as Alaskan huskies) represent an ever-changing crossbreed of the fastest dogs.

Huskies have continued to be used in sled-dog racing, as well as expedition and trek style tour businesses, and as a means of essential transportation in rural communities.[5] Huskies are also kept as pets, and groups work to find new pet homes for retired racing and adventure-trekking dogs.

It wasn't always so clear, but the Rust programming language is fundamentally about *empowerment*: no matter what kind of code you are writing now, Rust empowers you to reach farther, to program with confidence in a wider variety of domains than you did before.

Take, for example, "systems-level" work that deals with low-level details of memory management, data representation, and concurrency. Traditionally, this realm of programming is seen as arcane, accessible only to a select few who have devoted the necessary years learning to avoid its infamous pitfalls. And even those who practice it do so with caution, lest their code be open to exploits, crashes, or corruption.

Rust breaks down these barriers by eliminating the old pitfalls and providing a friendly, polished set of tools to help you along the way. Programmers who need to "dip down" into lower-level control can do so with Rust, without taking on the customary risk of crashes or security holes, and without having to learn the fine points of a fickle toolchain. Better yet, the language is designed to guide you naturally towards reliable code that is efficient in terms of speed and memory usage.

Programmers who are already working with low-level code can use Rust to raise their ambitions. For example, introducing parallelism in Rust is a relatively low-risk operation: the compiler will catch the classical mistakes for you. And you can tackle more aggressive optimizations in your code with the confidence that you won't accidentally introduce crashes or vulnerabilities.

But Rust isn't limited to low-level systems programming. It's expressive and ergonomic enough to make CLI apps, web servers, and many other kinds of code quite pleasant to write — you'll find simple examples of both later in the book. Working with Rust allows you to build skills that transfer from one domain to another; you can learn Rust by writing a web app, then apply those same skills to target your Raspberry Pi.

This book fully embraces the potential of Rust to empower its users. It's a friendly and approachable text intended to help you level up not just your knowledge of Rust, but also your

reach and confidence as a programmer in general. So dive in, get ready to learn—and welcome to the Rust community!

— Xiyu Zhai

Introduction