Rust: Idioms and Design Patterns

Nicholas Cameron
Mozilla Research
ncameron@mozilla.com

Abstract

Rust is a new (1.0 was released in May 2015) systems programming language. It promises memory safety without garbage collection, and race-free concurrency with neither a runtime, nor having a blessed concurrency system built in to the language.

Although syntactically familiar, programming in Rust can be radically different from programming in older object-oriented languages. There is no free lunch for memory management - Rust has no garbage collection and is memory safe, but this comes with the price of a complex and sometimes constraining type system. Rust also relies heavily on closures, higher-order functions and other functional programming techniques. Finally, Rust's trait system is closer to Haskell's type classes than to traditional code reuse mechanisms such as inheritance. Idiomatic Rust can therefore be quite different to programming in C++, Java, or other languages.

This talk will cover some of the common idioms and design patterns encountered when programming in Rust. We'll work through simple idioms used in everyday programming for tasks such as object creation and customisation, resource management, and destruction. We'll then cover more complex patterns often used in generic data structures and libraries. We'll also discuss some of the underlying themes and why these idioms and patterns occur in Rust.

The talk will not assume previous experience with Rust. The audience will gain an understanding of programming with Rust and its strengths and weaknesses. For those learning (or intending to learn) Rust, the talk should provide a short-cut to intermediate and advanced programming skills.

Categories and Subject Descriptors CR-number [*subcategory*]: third-level

Keywords Rust, design patterns, programming

NOOL, October 27, 2015, Pittsburgh, Pennsylvania, USA.

http://dx.doi.org/10.1145/nnnnnnn.nnnnnnn