WEBINAR

Moving from C++ to Rust

Craig Cuthbert | August 24, 2023

About ICS

Established in 1987, Integrated Computer Solutions, Inc. (ICS) delivers innovative software solutions with a full suite of services to accelerate development of successful next-gen products.

ICS is headquartered outside Boston in Waltham, Mass. with offices in California, Canada and Europe.

- UX and UI design services
- Full stack software development
- Cloud-native application development
- Medical regulatory compliance
- End-to-end software house

Our Markets

ICS specializes in software solutions for mission-critical and regulated industries, including:

Medical & Life Sciences

Automotive & Transportation

Industrial Controls & IoT

Aerospace & Defense

A Few of Our Customers

ICS

Agenda

- What is Rust?
- Discuss similarities and differences between C++ and Rust at a high level
- Look at common pain points in switching to Rust.
- Rust Pros and Cons
- Who is using Rust?
- Why would we want to use Rust vs C++?
- Language Features
- Cargo Build System
- Code Examples
- Future Directions
- Summary and Conclusions
- References

Introduction

Why would we want to use Rust vs C++?

Safety!

Speed Comparable to C/C++

Modern Language with Modern Tools

Fun!!

What is Rust?

- A big reason Rust was invented was to address some shortcomings of C and C++ while keeping similar performance.
- Multi-paradigm programming language designed for performance and safety.
- Although syntactically similar to C++ in some cases, it is an entirely different language with a steep learning curve.
- Can guarantee memory safety by using a “borrow checker” to validate references. In other words, null references are caught a compile time, not run time.
- Fully compiled, no runtime, comparable performance to C++.
- Does not use garbage collection; reference counting is optional.
- Originally developed by Mozilla research, now run by an independent non-profit foundation.

Pros of using Rust

- Modern Language with modern tools.
- Safety, Concurrency, Speed.
- Reduce or eliminate memory issues without sacrificing performance.
- Developer community likes it and it is growing in popularity.
- Open source libraries and adoption continues to grow.
- Eliminates complex C++ build scripts using the Cargo build system and package manager.
- Concurrency can be trivial compared to C++
- More errors are caught at compile time than in C++.
- The compiler will produce detailed error reports of any errors and warnings.
- Very good documentation.
- Testing, deployment and documentation are built into the language tools and not an “afterthought”

ICS

Cons of using Rust

- Steep learning curve.
- First step is to get the thing to compile!
- Need to approach Rust programming differently. Porting may not be straightforward (e.g if porting inheritance, it will need to be refactored using traits).
- Slower compile times.
- There are currently no mature GUI frameworks for Rust such as Qt.
- Qt can be used but requires unsafe code and/or generated bridge code. It may not be the best fit at the moment.
- Crossing boundaries (e.g. C++ to Rust or the reverse) can be difficult.

Who is using Rust?

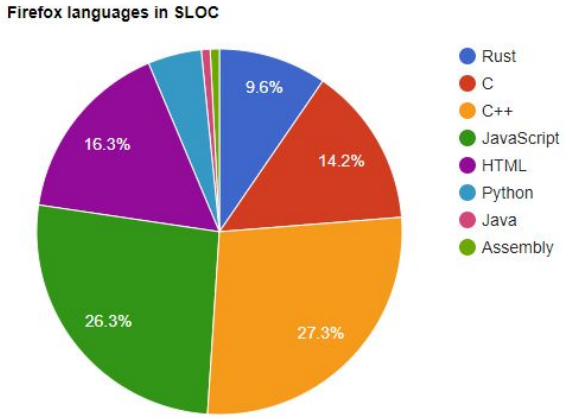
- Mozilla https://www.techrepublic.com/article/rust-not-firefox-is-mozillas-greatest-industry-contribution/
- Dropbox https://www.wired.com/2016/03/epic-story-dropboxs-exodus-amazon-cloud-empire/
- Figma https://www.figma.com/blog/rust-in-production-at-figma/
- Facebook https://engineering.fb.com/2021/04/29/developer-tools/rust/
- Discord https://github.com/serenity-rs/serenity
- Amazon https://aws.amazon.com/blogs/opensource/sustainability-with-rust/
- Microsoft https://learn.microsoft.com/en-us/windows/dev-environment/rust/rust-for-windows
- Cloudflare https://blog.cloudflare.com/introducing-oxy/

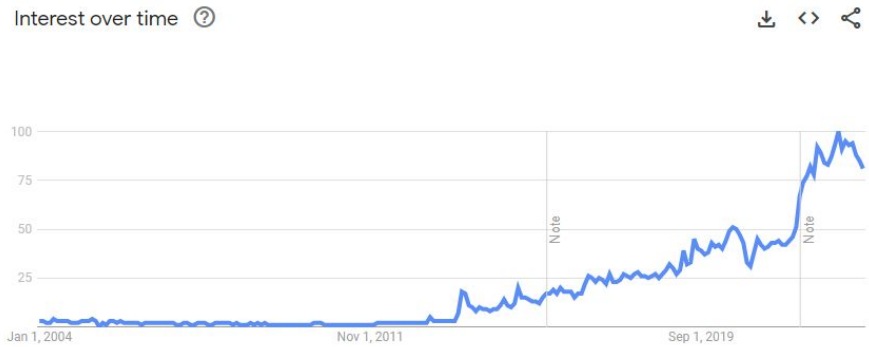



ICS

How much Rust is in Firefox?

Rust was invented by Mozilla for the Firefox browser to deal with the classic issues in C++. Since its inception, interest has been steadily climbing…





Language Features

Some similarities to C++:

- Data structures are similar to C++ structs.
- Control structures such as if, else, for, while are similar to C++ although the syntax is slightly different (e.g. no brackets around the condition).
- Enums are more powerful and can contain variant data values and methods.
- Break and Continue are similar to C++.
- Arrays, Hash Maps, Iterators etc. have similar function, but also differences.
- Has Lambda functions (closures).
- As in C++ we can create macros, but the syntax is strange (dev complaints).
- Data types can be declared explicitly or can be inferred by the compiler similar to the C++ auto keyword.

Simple Code Example

```
// Hello world plus factorial and for loop
fn factorial(i: u64) -> u64 {
    match i {
        0 => 1,
        n => n * factorial(n - 1),
    }
}
fn main() {
    println!("Hello, World!");
    for i in 0..20 {
       println!("factorial({}) = {}", i, factorial(i));
    }
}
```


ICS

CODE DEMO

SIMPLE EXAMPLE

ICS

Cargo Build System

- Tool to help manage Rust projects.
- Manages building code, downloading dependencies the code needs, and building those dependencies (and more)
- Conceptually similar to qmake or cmake (but doesn't generate make files).

In the simplest case, Cargo.toml just contains:

```
[package]
name = "hello_world"
version = "0.0.1"
authors = [ "Linus Torvalds <ltorvalds@linuxfoundation.org>" ]
```

Usually source code is in a src subdirectory, e.g. src/main.rs

Cargo Build System

```
% cargo build
   Compiling simple_example v0.0.1
(C:\Source\Rust\webinar\simple_example)
    Finished dev [unoptimized + debuginfo] target(s) in 0.26s

% cargo run
    Finished dev [unoptimized + debuginfo] target(s) in 0.02s
     Running `target\debug\simple_example.exe`
Hello, World!
```

Trivia: Rust programmers refer to themselves as "Rustaceans".

CODE DEMO

HELLO WORLD

ICS

Language Features

Differences from C++ - Ownership and Memory Management:

- An ownership system where all values have a unique owner, and the scope of the value is the same as the scope of the owner.
- Since there is only one owner, any reference is tied to the lifetime of the owner. If this is ambiguous the compiler will error.
- Lifetimes are tied to code blocks unless the lifetime is explicitly set by the developer. In other words, if a lifetime is ambiguous, the program will not compile.
- The Rust compiler enforces these rules at compile time and also checks that all references are valid. Dangling/null pointers are thus eliminated.
- Memory and other resources are managed through the resource acquisition is initialization (RAII) convention, with optional reference counting using a “Reference Counted Smart Pointer” or Rc<T>.

Language Features

Ownership and Memory Management:

- Type declaration for variables declared with the keyword let.
- Values can be passed by immutable reference, by mutable reference, or by value. Variables default to immutable. Can add ‘mut’ keyword to change.
- At all times, there can either be multiple immutable references or one mutable reference (an implicit readers-writer lock).
- Race and deadlock conditions are limited by compile time restrictions.
- Does not use an automated garbage collection system, but also does not require explicit deletion of pointers.
- “unsafe” code which can subvert some of these restrictions may be written using the language's unsafe keyword.

Language Features

Ownership:

- Using a (non primitive type such as a string) variable directly with another variable or when passed to a function transfers the ownership. The previous variable is dropped (in this case the stack pointer s1 will be removed)

```
let s1 = String::from("abc");
let s2 = s1; // causes a "move" and s1 is now invalid
println!("s1: {}, s2: {}", s1, s2); // compile error
```

When s2 variable eventually goes out of scope at the end of this function, the s2 stack variable will be removed along with the "abc" heap allocation automatically.

ICS

Language Features

Ownership:

```
let s1 = String::from("abc");
let s2 = &s1; // "borrow" a reference
println!("s1: {}, s2: {}", s1, s2); // ok

let s1 = String::from("abc");
let s2 = s1.clone(); // deep copy
println!("s1: {}, s2: {}", s1, s2); // ok
```

CODE DEMO

LIFETIMES

ICS

Language Features

Types and Polymorphism:

- Composition is favored over inheritance.
- No inheritance of types, but inheritance of “Traits”.
- Polymorphism is done using traits (similar to an interface) .
- Functions can be given generic parameters, which usually require the generic type to implement a certain trait or traits.
- Object system is based around implementations, traits and structured types.

CODE DEMO

OBJECT EXAMPLE

ICS

Language Features

Language Extensions:

- Possible to extend the Rust language using the procedural macro mechanism.
- The println! macro is an example of a function-like macro.

Trivia: Rust has been voted the "most loved programming language" in the Stack Overflow Developer Survey every year since 2016.

ICS

Future Directions

- Linux has recently added support for Kernel development using Rust.
- Microsoft has created Rust for Windows, which allows creating Rust apps for Windows with bindings for WIN32 APIs.
- The Google Android team writing low level Android code in Rust.
- Facebook is using Rust in all aspects of development with hundreds of developers writing millions of lines of Rust code.
- Support for embedded platforms (including microcontrollers).
- Ability to compile to WebAssembly to run in a browser.

Fun fact: A 2021 survey of programmers by SlashData reported that Rust and Lua were the two fastest growing programming language communities in the past 12 months.

ICS

Summary and Conclusions

Two Linux security researchers estimate that about two-thirds of Linux kernel vulnerabilities come from memory safety issues. One Microsoft security expert estimates that 70% of the CVEs originating at Microsoft are memory safety issues. Rust, in theory, can avoid these with its inherently safer memory model.

Rust is a possible replacement for C++ on multiple platforms. Mostly driven by memory safety and the fact that it is more modern than C++. Comparable in performance and platform support and controlled by an independent foundation https://foundation.rust-lang.org/.

To get started visit https://rustup.rs/ and follow the instructions. I recommend using https://code.visualstudio.com/docs/languages/rust as the editor which has very good support for Rust.

Resources to learn more about Rust

- https://www.rust-lang.org/
- https://doc.rust-lang.org/book/title-page.html
- https://doc.rust-lang.org/stable/rust-by-example/
- https://crates.io/
- Online training sites such as Udemy and Pluralsight
  - https://www.udemy.com/course/ultimate-rust-crash-course/
  - https://www.udemy.com/course/ultimate-rust-2/
  - https://www.udemy.com/course/rust-fundamentals/
  - https://www.udemy.com/course/rust-programming-master-class-from-beginner-to-expert
  - https://www.pluralsight.com/courses/fundamentals-rust

ICS

Other References

Some helpful references

- https://doc.rust-lang.org/std/
- https://rust-lang.github.io/rust-clippy/master/index.html
- https://cheats.rs
- https://www.kernel.org/doc/html/latest/rust/index.html
- https://docs.microsoft.com/en-us/windows/dev-environment/rust/rust-for-windows
- https://github.com/rust-unofficial/awesome-rust#applications
- https://gist.github.com/zbraniecki/b251714d77ffebbc73c03447f2b2c69f
- https://www.zdnet.com/article/linus-torvalds-on-where-rust-will-fit-into-linux
- https://slint.dev/
- https://github.com/KDE/rust-qt-binding-generator

ICS

QUESTIONS?

ICS