Trust Rust! (For Safe Systems Programming)

Pramode C.E

25 November, 2017

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

Alan Perlis.

Figure 1: Robert Morris Jr

Morris Worm: 1988
(https://en.wikipedia.org/wiki/Morris_worm)

How to smash the stack for fun and profit: 1996 (http://phrack.org/issues/49/14.html#article)

The most recent MacOS release fixed 32 vulnerabilities caused due to memory safety (https://support.apple.com/en-us/HT208221).

The Most recent Google Chrome release fixed 10 memory safety issues (https://chromereleases.googleblog.com/2017/10/stable-channel-update-for-desktop.html) and Firefox fixed 38 (https://www.mozilla.org/en-US/security/advisories/mfsa2017-24/).

Google's OSS-Fuzz and Project Zero have uncovered a large number of memory safety related vulnerabilities.

October 2, 2017: Yet more DNS and DHCP vulnerabilities: https://security.googleblog.com/2017/10/behind-masq-yet-more-dns-and-dhcp.html

Almost all of them are memory safety issues! Example:

- CVE-2017-14493 Stack based overflow
- CVE-2017-14495 Lack of “free()”
- CVE-2017-14492 Heap based overflow

Intel® Manageability Engine Firmware 11.0.x.x/11.5.x.x/11.6.x.x/11.7.x.x/11.10.x.x/11.20.x.x

CVE ID	CVE Title	CVSSv3 Vectors
CVE-2017-5705	Multiple buffer overflows in kernel in Intel Manageability Engine Firmware 11.0/11.5/11.6/11.7/11.10/11.20 allow attacker with local access to the system to execute arbitrary code.	8.2 High AV:L/AC:L/PR:H/UI:N/S:C/C:H/I:H/A:H
CVE-2017-5708	Multiple privilege escalations in kernel in Intel Manageability Engine Firmware 11.0/11.5/11.6/11.7/11.10/11.20 allow unauthorized process to access privileged content via unspecified vector.	7.5 High AV:L/AC:H/PR:L/UI:N/S:C/C:H/I:H/A:N
CVE-2017-5711	Multiple buffer overflows in Active Management Technology (AMT) in Intel Manageability Engine Firmware 8.x/9.x/10.x/11.0/11.5/11.6/11.7/11.10/11.20 allow attacker with local access to the system to execute arbitrary code with AMT execution privilege.	6.7 Moderate AV:L/AC:L/PR:H/UI:N/S:U/C:H/I:H/A:H
CVE-2017-5712	Buffer overflow in Active Management Technology (AMT) in Intel Manageability Engine Firmware 8.x/9.x/10.x/11.0/11.5/11.6/11.7/11.10/11.20 allows attacker with remote Admin access to the system to execute arbitrary code with AMT execution privilege.	7.2 High AV:N/AC:L/PR:H/UI:N/S:U/C:H/I:H/A:H

Figure 2: Intel Management Engine Vulnerabilities

Figure 3: IoT Security

Figure 4:

Solutions?

- Programmer discipline?

Solutions?

```
// undef1.c
static void (*Do)();

static void EraseAll() {
  printf("remove all files ...\n");
}
void NeverCalled() {
  Do = EraseAll;
}
int main() {
  Do();
}
```

[More fun: https://blog.regehr.org/archives/213]

Solutions?

- Static Analysis tools?
- Languages like Java, Go, etc?

Why Rust?

- A modern, memory-safe replacement for C/C++ with excellent tooling.
- Memory safety achieved using innovative type system concepts (affine types) and *not* using Garbage Collection.
- Statically prevents data races in multi-threaded code.
- Many high level features (mostly borrowed from statically typed functional programming languages) without any run time overhead (so-called “zero cost abstractions”).

A bit of Rust history

- Started by Graydon Hoare as a personal project in 2006
- Mozilla foundation started sponsoring Rust in 2010
- Rust 1.0 released in May, 2015
- Regular six week release cycles
- Separate “stable” and “nightly” release channels provide access to experimental features without breaking stability.

Core language features

- Memory safety without garbage collection
  - Ownership
  - Move Semantics
  - Borrowing and lifetimes
- Static Typing with Type Inference
- Algebraic Data Types (Sum and Product types)
- Exhaustive Pattern Matching
- Trait-based generics
- Iterators
- Zero Cost Abstractions
- Concurrency without data races
- Efficient C bindings, minimal runtime

Firefox Quantum: Rust in action

Figure 5: Firefox Quantum

Firefox Quantum: Stylo CSS

https://hacks.mozilla.org/2017/08/inside-a-super-fast-css-engine-quantum-css-aka-stylo/

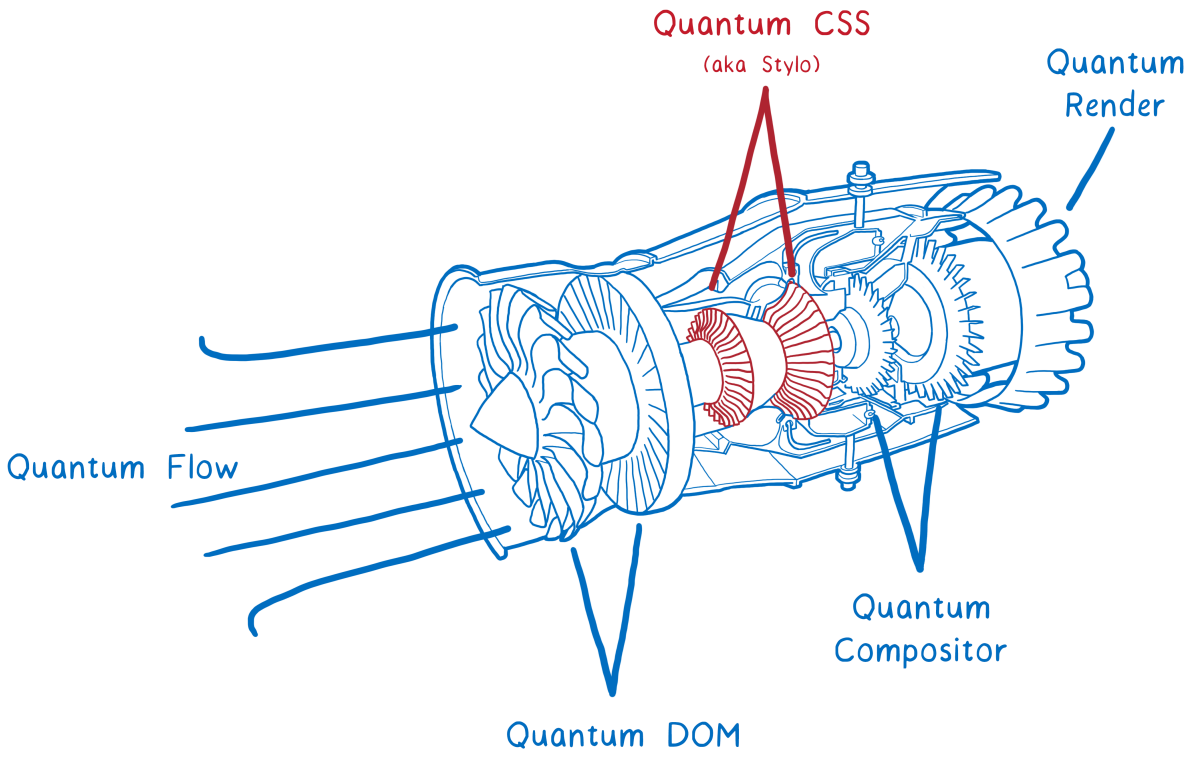

Figure 6:

Friends of Rust

[https://www.rust-lang.org/en-US/friends.html]

- Mozilla, Coursera, Dropbox, Canonical, npm, Atlassian, ...

Friends of Rust - Ather Energy

An electric scooter company from India!
(https://www.atherenergy.com/)

Figure 7: Ather Electric Scooter

Structure of this workshop

- Session 1: Understand Rust concepts not related to memory safety. Also, some fun demos!
- Session 2: Memory safety and Multithreaded programming.

Why Not Rust?

- Complex language with a steep learning curve. Not suitable for beginners.

Why Not Rust?

- Difficult to express many data structure patterns in “safe” Rust.
- Very young . . . library ecosystem not as mature as that of older languages (but this will improve with time).

Why Not Rust?

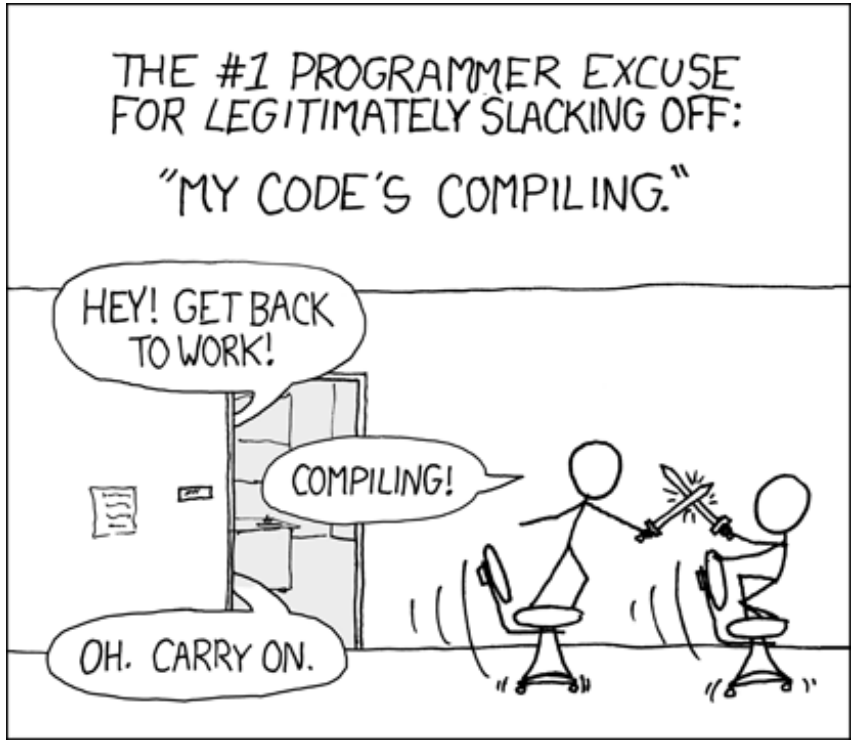

Figure 8:

Demo: Redox OS!

[https://www.redox-os.org/]

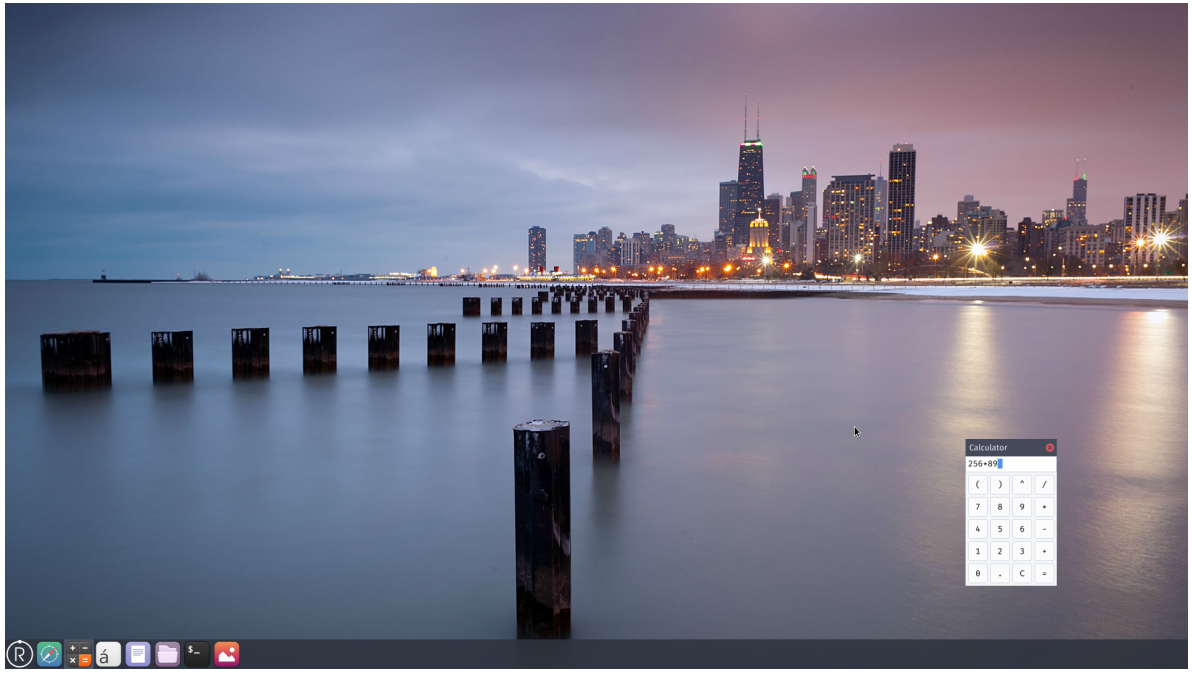

Figure 9: Redox OS - a pure Rust Unix-like OS

Demo: Rocket, a web framework!

[http://rocket.rs]

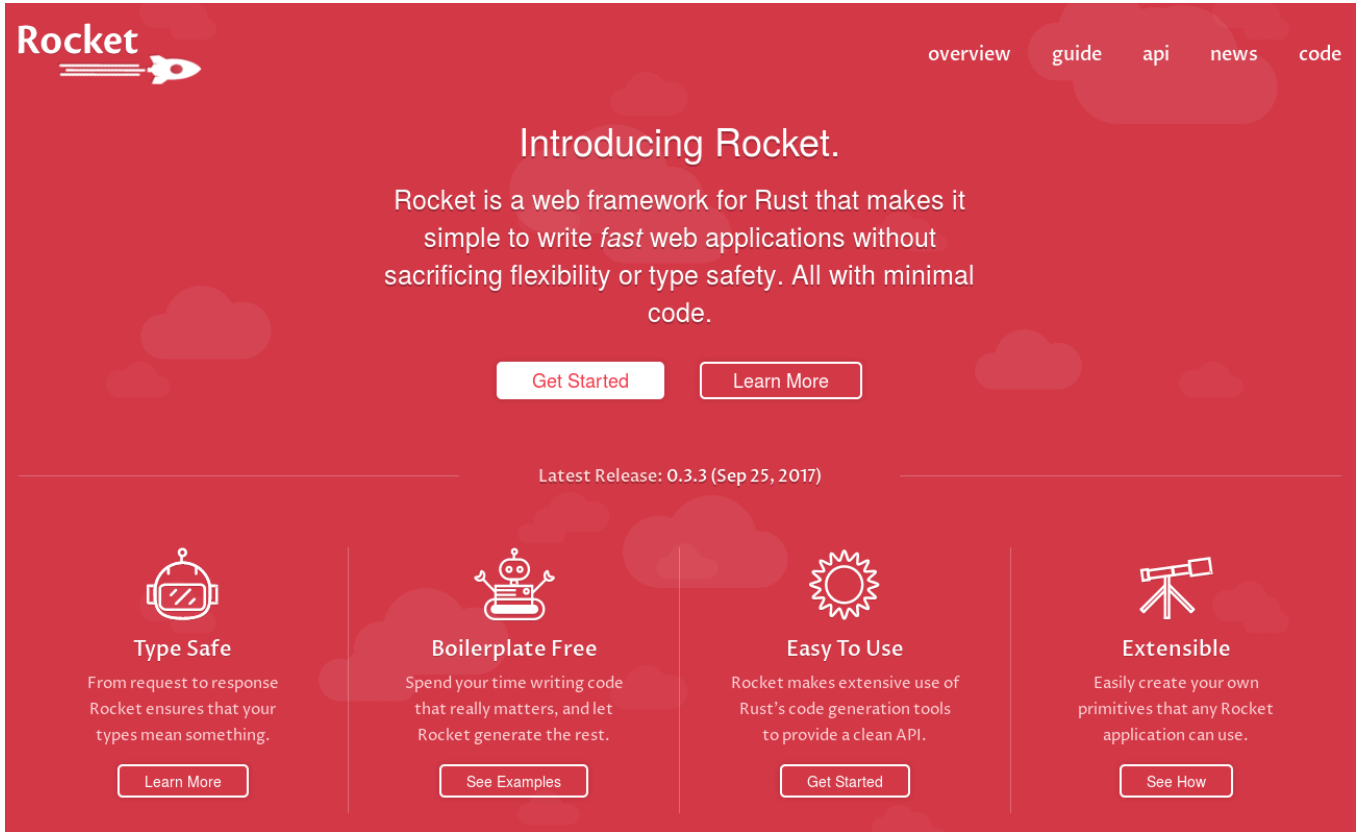

Figure 10: Rocket, a web framework

Hello, LED!

Folders: code/dynamic-typing; code/hello-led/, code/launchpad
For a change, we will write an interesting hello, world!

Rust in small embedded systems

- Runs perfectly on ARM Cortex-M microcontrollers
- AVR / MSP430 ports getting ready
- Main issue is availability of LLVM back-end
- The standard library will not be available

Rust in small embedded systems

- http://blog.japaric.io/fearless-concurrency/
  - Re-thinking the very concept of an RTOS.
  - Uses the Rust type system to provide deadlock free resource management in a concurrent environment.

Rust in small embedded systems

- https://www.tockos.org/
  - More of a traditional embedded OS capable of running multiple apps concurrently.
  - Uses the Rust type system to protect the core kernel against buggy/malicious device driver code.
  - Designed for use in security critical appliances, consumer IoT devices etc.

More language basics

Folder: code/basics

Tools Demo

Folder: code/tools

Benchmarking Python and Rust

Folder: code/benchmark

Sum types and pattern matching

Folder: code/sum-types; code/pattern-match

Product types and the newtype pattern

Folder: code/product-types; code/newtype-pattern

Traits and Generics

Folder: code/traits

Option/Result types and error handling

Folder: code/option

Collections

Folder: code/collections

API Design

Folder: code/apidesign

Interesting crates

Folder: code/interesting-crates

Resource Leaks

Folder: code/resource-leaks

Type-driven design

Folder: code/type-driven-design