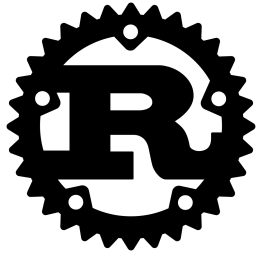


Rust Introduction

ECS 189C Lecture 8 – Spring 2024

What is Rust?

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

Systems Programming

- Operating Systems
- Device Drivers
- Embedded Systems
- Real Time Systems
- Networking
- Virtualization and Containerization
- Scientific Computing
- Video Games
- Graphics
- Web applications
- “Back-end” engineering in general

Some Principles of Systems Programming

Your code does not exist in a vacuum

- **Interaction** with hardware, operating system, or other running software

Everything can fail

- Some failures can be catastrophic

You need to know how fast your code will run

Systems Programming

→ You want to know how fast your code will run

```
for i in range(100):
    print(i)
    data.append(i)

...

for j in range(100):
    process_very_important_request(j)
```

Who uses Rust?

Amazon: https://aws.amazon.com/blogs/aws/firecracker-lightweight-virtualization-for-serverless-computing

Facebook: https://developers.libra.org/

Mozilla: Firefox

Discord: https://blog.discord.com/why-discord-is-switching-from-go-to-rust-a190bbca2b1f

Dropbox: backend infrastructure

Who uses Rust?

"For five years running, Rust has taken the top spot as the most loved programming language." -- StackOverflow

- https://insights.stackoverflow.com/survey/2020#technology-most-loved-dreaded-and-wanted-languages-loved

Now a top 20 language in most language popularity rankings, and one of the fastest-growing: https://www.reddit.com/r/rust/comments/hz7dfp/rust_is_now_a_top_20_language_in_all_of_the_5/

Second most used language for Advent of Code this year, after Python: https://app.powerbi.com/view?r=eyJrIjoiZTQ3OTlmNDgtYmZlMS00ZTJmLTkwYTgtMWQyMTkxNWI5NGM1IiwidCI6IjQwOTEzYjA4LTQyZTYtNGMxOS05Y2FiLTRmOWZlM2U0YzJmZCIsImMiOjI

Why Rust?

Speed and **Safety**

- Statically typed
- Compile-time checked for memory safety
- Trustworthy concurrency (mulithreaded, parallel, and distributed applications)

Examples of Safety: Static Typing

```
def process_client_request(s):
    // accidentally assume s is a string
    data = parse_data(s)
```

“Type Safety”: type errors are not possible

Note: C and C++ are not type safe!

Examples of Safety: Undefined Behavior

- `Buffer overflow`
- `Accessing an element beyond the end of an array`
- `Modifying a data structure while iterating over it`

Undefined behavior! ⇒ your program could do literally anything

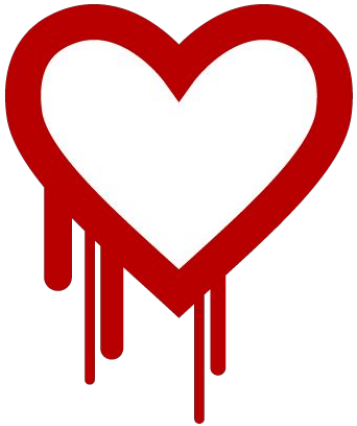
Almost everything is built on top of systems languages

- Java Virtual Machine
- CPython

Q: Why doesn't C/C++ prevent such things?

Examples of Safety: Memory safety

Heartbleed: https://xkcd.com/1354/



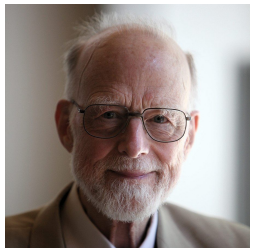
Microsoft: 70 percent of all security bugs are memory safety issues

Percentage of memory safety issues has been hovering at 70 percent for the past 12 years.

Examples of safety: null pointer dereference

"I call it my billion-dollar mistake. It was the invention of the null reference in 1965... My goal was to ensure that all use of references should be absolutely safe, with checking performed automatically by the compiler. But I couldn't resist the temptation to put in a null reference, simply because it was so easy to implement. This has led to innumerable errors, vulnerabilities, and system crashes, which have probably caused a billion dollars of pain and damage in the last forty years."

-- Sir Tony Hoare



Examples of Safety: Concurrency (Race Conditions)

A race condition is where the program behavior depends on the order of execution of uncontrollable events.

- different threads or processes
- different client requests
- different hardware events

Concurrency is hard to think about

1985-1987: Therac-25 https://en.wikipedia.org/wiki/Therac-25

Examples of Speed: “Zero Cost Abstraction”

“What you don’t use, you don’t pay for. And further: What you do use, you couldn’t hand code any better.”

Examples of Speed: Garbage Collection

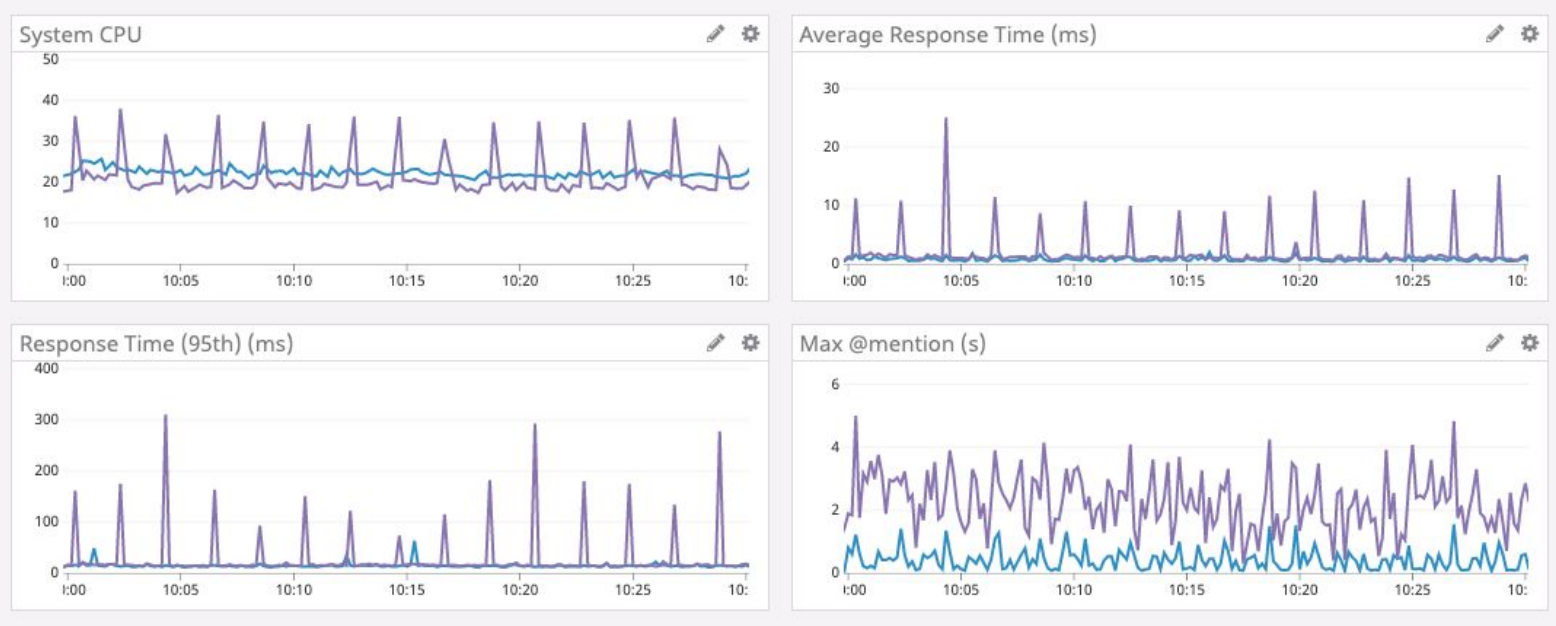
```
for i in range(100):
    print(i)
    data.append(i)

...

for j in range(100):
    process_very_important_request(j)
```

Examples of Speed: Garbage Collection

Why Discord is switching from Go to Rust:
https://blog.discord.com/why-discord-is-switching-from-go-to-rust-a190bbca2b1f



Examples of Rust Safety: Null-pointer references

The problems with null:

- Used to represent a missing value: String getValue(HashTable<String, String> t);
- Use to represent an error: void* malloc(size_t size)
- Very easy to ignore.

In Rust: optional values

Zero-cost abstraction!

```
// Optional Values:
enum Option<P> {
  Some(P),
  None
}
```

Examples of Rust Safety: Error Handling

Philosophy: if you don’t expose an error, library user won’t deal with it

```
enum Result<T, E> {
  Ok(T),
  Err(E),
}

fn read(&mut fd: FileDescriptor, buf: &mut [u8]) -> Result<usize, std::io::Error>;
```

Mini Demos

https://github.com/upenn-cis198/lecture1/tree/master/demos

Helpful resources: The Rust book

https://doc.rust-lang.org/stable/book/

Interactive version:

https://rust-book.cs.brown.edu/

Helpful resources:

another Rust book