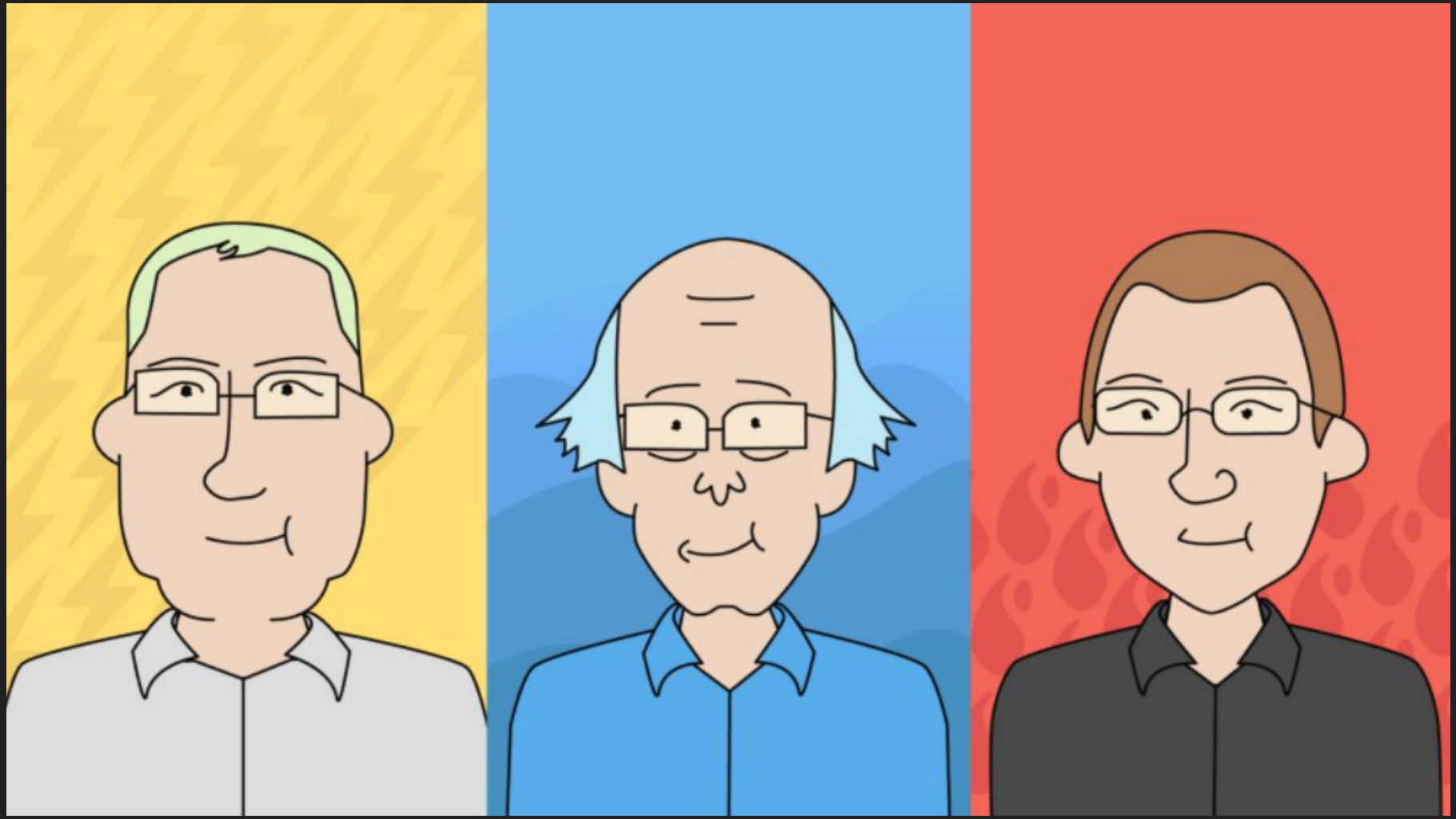
El Lenguaje de Programación Rust

Felipe Vallejo

Temas por abordar

1. Historia
2. Para quien es Rust?
3. Cargo y rust-analyzer
4. IDEs
5. Comunidad
6. Diferencias más destacadas con C++
7. rustfmt
8. Reglas de Ownership y Borrowing
9. Consumo de energia
10. Awesome Rust
11. Colecciones
12. Programación Orientada a Objetos
13. Puntero a NULL en Rust
14. Que otros lenguajes están tratando de reemplazar a C++?
15. Mini Proyecto
16. Conclusion
17. Bibliografia

Historia



```
fn main() {
    log "hello, world";
}

fn max(int x, int y) -> int {
    if (x > y) {
        ret x;
    } else {
        ret y;
    }
}
```

```
fn main(){
    println!("hello, world");
}

fn max(x: i32, y: i32) -> i32 {
    if x > y {
        x
    } else {
        y
    }
}
```

```
let x = @5; // GC'd pointer
let y = ~5; // unique pointer
let z = &5; // borrowed pointer
```

LLVM currently supports compiling of Ada, C, C++, D, Delphi, Fortran, Haskell, Julia, Objective-C, Rust, and Swift using various front ends.

Announcing Rust 1.0

May 15, 2015 · The Rust Core Team

Today we are very proud to announce the 1.0 release of Rust, a new programming language aiming to make it easier to build reliable, efficient systems. **Rust combines low-level control over performance with high-level convenience and safety guarantees**. Better yet, it achieves these goals without requiring a garbage collector or runtime, making it possible to use Rust libraries as a "drop-in replacement" for C. If you'd like to experiment with Rust, the "Getting Started" section of the Rust book is your best bet (if you prefer to use an e-reader, Pascal Hertleif maintains unofficial e-book versions as well).

Elon Musk @elonmusk · Feb 22

Replying to @elonmusk and @gdb

Compilers can be way better too. Not enough effort spent there.

118 128 2,789

jack @jack · Feb 22

Replying to @elonmusk and @gdb

rust has fixed this dramatically

145 232

Elon Musk
@elonmusk

Replying to @jack and @gdb

I'm a fan of Rust. Clearly scales well, given that Discord uses it.

jack
@jack

rust is a perfect programming language

1:20 AM · Dec 24, 2021 · Twitter for iPhone

In October 2022, a pull request for accepting the implementation for Rust for Linux was approved by Torvalds.[8]

Programming, scripting, and markup

For the sixth-year, Rust is the most loved language, while Python is the most wanted language for its fifth-year.

Rust Foundation

Para quien es Rust?


FRONTEND
BACKEND
WEEKEND

NSA recommends that organizations use memory safe languages when possible and bolster protection through code-hardening defenses such as compiler options, tool options, and operating system configurations.

In 2019, a Microsoft security engineer reported that 70 percent of all security vulnerabilities were caused by memory safety issues. In 2020, a team at Google similarly reported that 70 percent of all "severe security bugs" in Google Chromium were caused by memory safety problems.

- Programadores de C++
- Programadores funcionales
- Programadores de Js o Python

Pinned

Signal-Android Public

A private messenger for Android.

Java 23k 5.6k

Signal-Desktop Public

A private messenger for Windows, macOS, and Linux.

TypeScript 12.9k 2.3k

Signal-iOS Public

A private messenger for iOS.

Swift 9.5k 2.6k

Signal-Server Public

Server supporting the Signal Private Messenger applications on Android, Desktop, and iOS

Java 7.9k 1.9k

libsignal Public

Home to the Signal Protocol as well as other cryptographic primitives which make Signal possible.

Rust 1.5k 197

Cargo y rust-analyzer


C++ FANS
"IMAGINE TAKING 1 HOUR TO RUN A PROGRAM"
imgflip.com
PYTHON FANS
"IMAGINE SPENDING 1 HOUR TO FIND A COMPILER"

Mac / Linux

```
$ curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
```

Windows

```
C:\Users\LuisVallejo>gsudo choco install rust
```

```
cargo new hello_cargo
# New directory created, cargo has created the files inside as well
cd hello_cargo
# cargo has generated two files: Cargo.toml and a src folder with main.rs
# it also initialized a new Git repo along with our .gitignore file
```

```
PS C:\Users\LuisVallejo\Documents\hello_cargo\src> cargo check
    Checking hello_cargo v0.1.0 (C:\Users\LuisVallejo\Documents\hello_cargo)
    Finished dev [unoptimized + debuginfo] target(s) in 0.83s
PS C:\Users\LuisVallejo\Documents\hello_cargo\src> cargo build
   Compiling hello_cargo v0.1.0 (C:\Users\LuisVallejo\Documents\hello_cargo)
    Finished dev [unoptimized + debuginfo] target(s) in 0.60s
PS C:\Users\LuisVallejo\Documents\hello_cargo\src> cargo run
    Finished dev [unoptimized + debuginfo] target(s) in 0.01s
     Running `C:\Users\LuisVallejo\Documents\hello_cargo\target\debug\hello_cargo.exe`
Hello, world!
```

```
[package]
name = "hello_cargo"
version = "0.1.0"
edition = "2021"

# See more keys and their definitions at
#https://doc.rust-lang.org/cargo/reference/manifest.html

[dependencies]
```

crates.io

Browse All Crates | 菲利 ▾

The Rust community's crate registry

Press 'S' to focus this searchbox...

Install Cargo

Getting Started

Instantly publish your crates and install them. Use the API to interact and find out more information about available crates. Become a contributor and enhance the site with your work.

23,195,125,306
Downloads

96,806
Crates in stock

New Crates

Most Downloaded

Just Updated

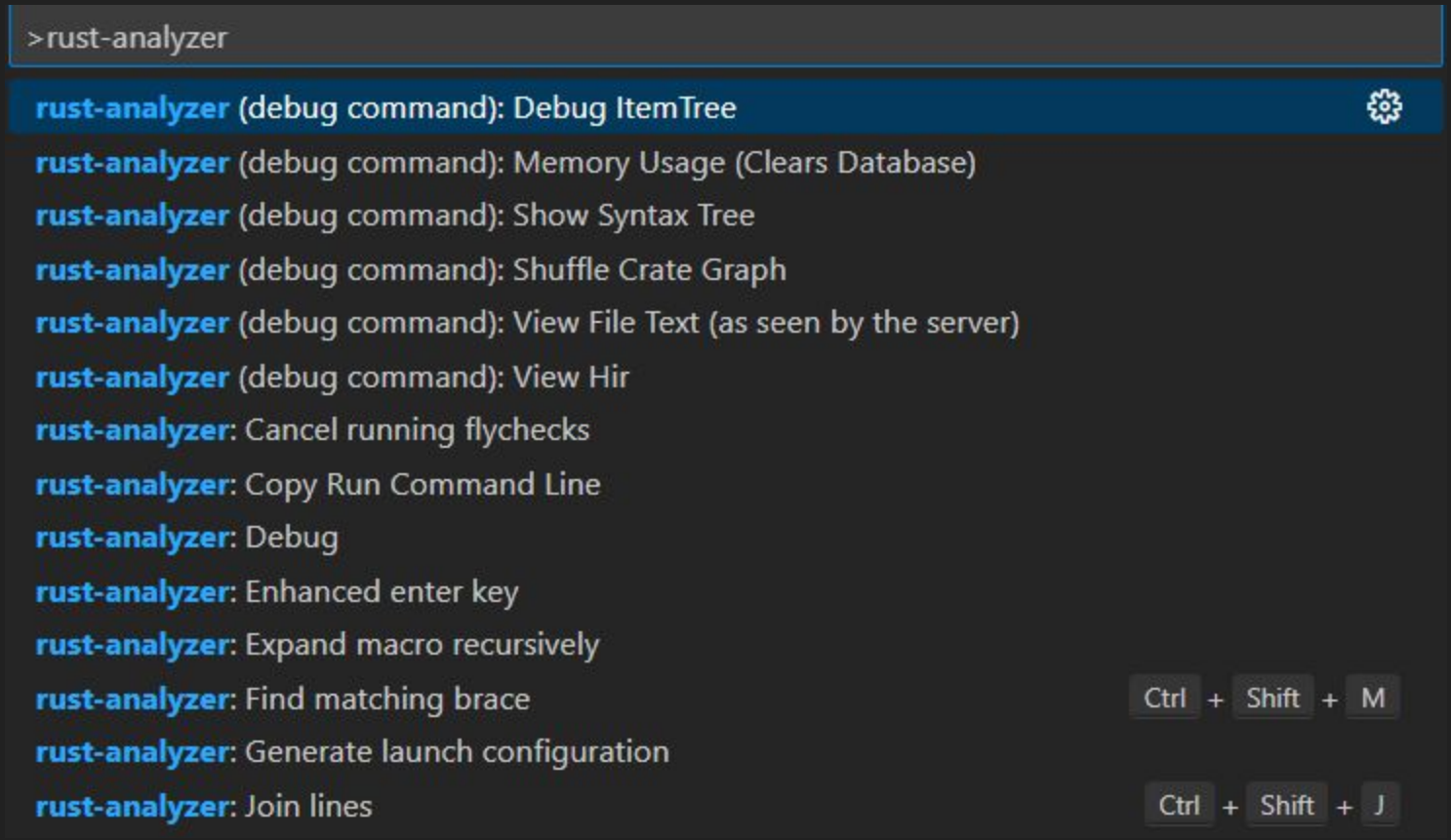
rust. analyzer

Bringing a *great* IDE experience
to the Rust programming language.

Source

About

rust-analyzer is an implementation of Language Server Protocol for the Rust programming language. It provides features like completion and goto definition for many code editors, including VS Code, Emacs and Vim.


>rust-analyzer
rust-analyzer (debug command): Debug ItemTree
rust-analyzer (debug command): Memory Usage (Clears Database)
rust-analyzer (debug command): Show Syntax Tree
rust-analyzer (debug command): Shuffle Crate Graph
rust-analyzer (debug command): View File Text (as seen by the server)
rust-analyzer (debug command): View Hir
rust-analyzer: Cancel running flychecks
rust-analyzer: Copy Run Command Line
rust-analyzer: Debug
rust-analyzer: Enhanced enter key
rust-analyzer: Expand macro recursively
rust-analyzer: Find matching brace
Ctrl + Shift + M
rust-analyzer: Generate launch configuration
rust-analyzer: Join lines
Ctrl + Shift + J

IDEs


IJ
Vim
CL


SO YOU THINK VS CODE IS THE BEST EDITOR
I do. And I'm tired of pretending it's not.

Comunidad

Ranking en GitHub

- Microsoft
- Google
- GitHub
- Meta
- FreeCodeCamp
- Alibaba
- Vuejs
- TensorFlow
- Apache
- Meta Research
- Rust Lang
- Tencent

- Go
- Swift
- TypeScript
- Rust
- Kotlin

Read the documentation for 15 minutes

Stack Overflow for 2 hours

The Rust Programming Language

by Steve Klabnik and Carol Nichols, with contributions from the Rust Community

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.

> **Want a more interactive learning experience? Try out a different version of the Rust Book, featuring: quizzes, highlighting, visualizations, and more**: https://rust-book.cs.brown.edu

Rust by Example

Rust is a modern systems programming language focusing on safety, speed, and concurrency. It accomplishes these goals by being memory safe without using garbage collection.

Rust by Example (RBE) is a collection of runnable examples that illustrate various Rust concepts and standard libraries. To get even more out of these examples, don't forget to install Rust locally and check out the official docs. Additionally for the curious, you can also check out the source code for this site.

rustlings 🦀 ❤️

Greetings and welcome to `rustlings`. This project contains small exercises to get you used to reading and writing Rust code. This includes reading and responding to compiler messages!

...looking for the old, web-based version of Rustlings? Try here

Alternatively, for a first-time Rust learner, there are several other resources:

- The Book - The most comprehensive resource for learning Rust, but a bit theoretical sometimes. You will be using this along with Rustlings!
- Rust By Example - Learn Rust by solving little exercises! It's almost like `rustlings`, but online

CIS 198: Rust Programming

University of Pennsylvania

Spring 2016

Home | Schedule | Policies | Final Projects

This schedule will be filled out as the semester progresses. (*Tentative*)

#	Date	Lecture	Assignment
1	1/20	• 00 - Course Introduction (Slides)	• HW00 - Due 1/25 • HW01 - Due 1/28
	1/21	*Rust 1.6 Release*	
2	1/27	• 01 - Ownership (Slides) • 02 - Structured Data (Slides)	• HW02 - Due 2/03
	2/01	*Add Period Ends*	
3	2/03	• 03 - Generics & Traits (Slides)	• HW03 - Due 2/10
4	2/10	• 04 - Closures (Slides) • 05 - std: Standard Library (Slides)	• HW04 - Due 2/17
5	2/17	• 06 - std: Pointer Types (Slides) • 07 - Misc: Syntax, Crates, std (Slides)	• HW05 - Due 2/26
	2/19	*Drop Period Ends*	
6	2/24	• 08 - I/O (Slides) • 09 - Networking (Slides)	• HW06 - Due 3/06
7	3/02	• 10 - Concurrency I (Slides)	• *None*
	3/03	*Rust 1.7 Release*	
	3/05–3/13	*Spring Break*	
8	3/16	• 11 - Concurrency II (Slides)	• HW07 - Due 3/23
9	3/23	• 12 - Unsafe Rust & FFI (Slides)	• Project Ideas & Proposals
	3/25	*Withdraw Period Ends*	

Diferencias más destacadas con C++

learn Python after C++

learn C++ after Python

Torre de Hanoi

```cpp
#include <iostream>
using namespace std;

void towerOfHanoi(int x, int from, int aux, int to){
    if(x > 0){
        towerOfHanoi(x-1, from, to, aux);
        printf("From %d to %d\n", from, to);
        towerOfHanoi(x-1, aux, from, to);
    }
}

int main(){
    int n = 3;
    towerOfHanoi(n, 1, 2, 3);
    return 0;
}
```

Rust

```rust
fn tower_of_hanoi(x: i32, from: i32, aux: i32, to: i32) {
    if x > 0 {
        tower_of_hanoi(x - 1, from, to, aux);
        println!("From {} to {}", from, to);
        tower_of_hanoi(x - 1, aux, from, to);
    }
}

fn main() {
    let x = 3;
    tower_of_hanoi(x, 1, 2, 3);
}
```

rustfmt

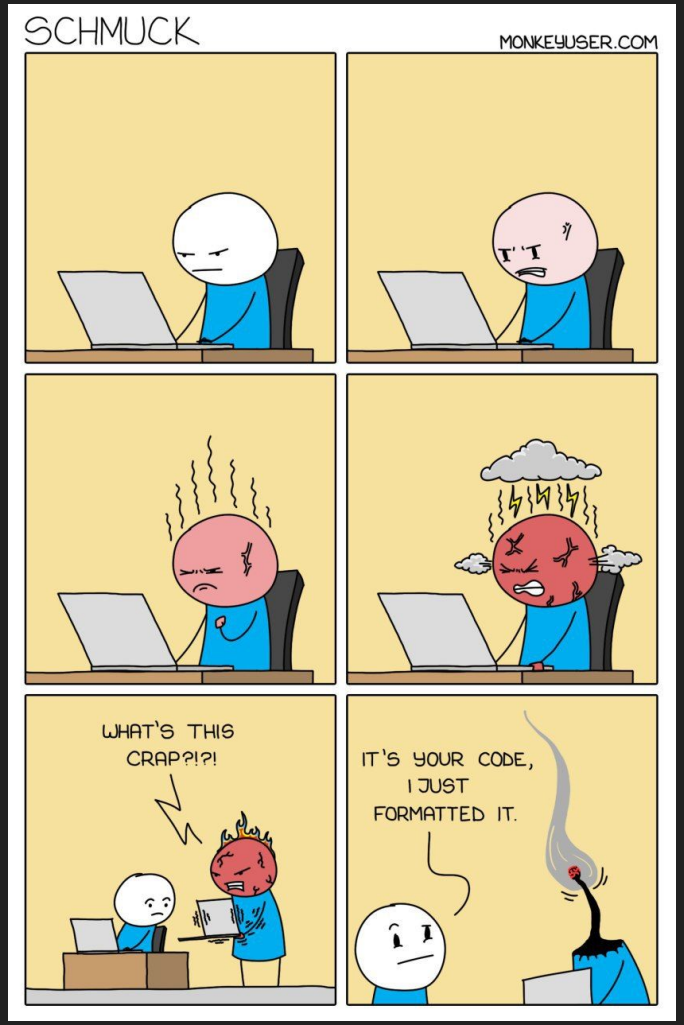
styleguide

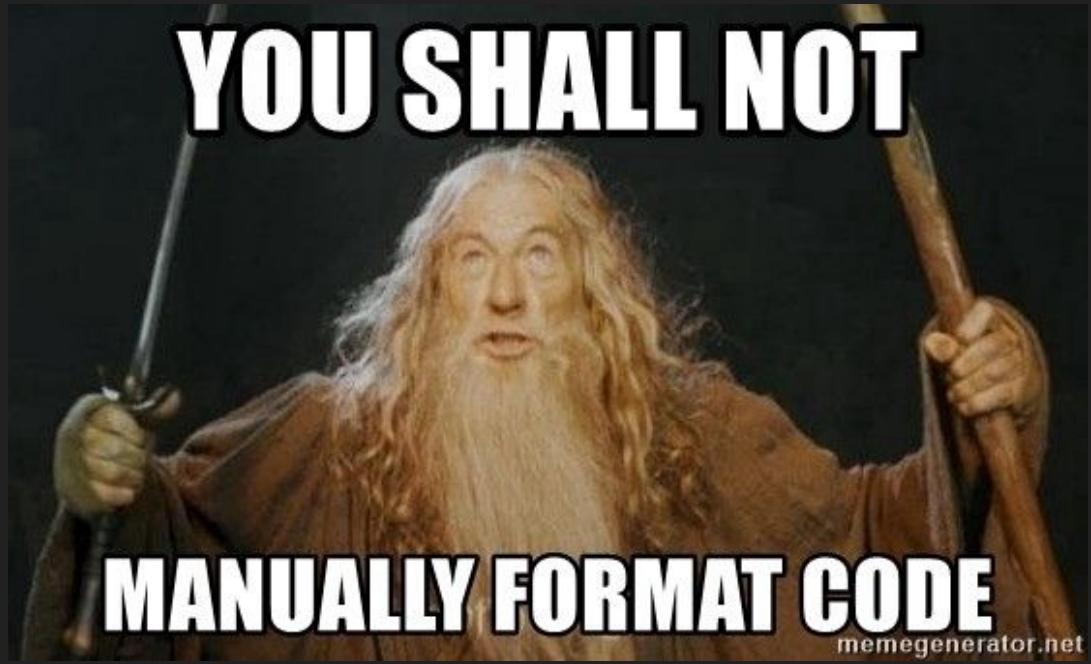
Google Style Guides

Every major open-source project has its own style guide: a set of conventions (sometimes arbitrary) about how to write code for that project. It is much easier to understand a large codebase when all the code in it is in a consistent style.

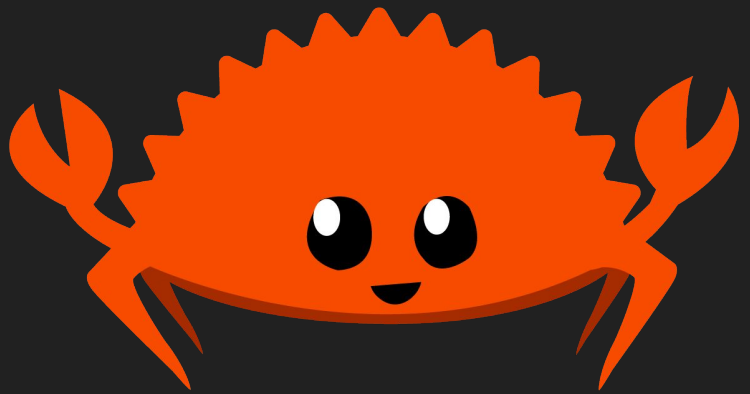
"Style" covers a lot of ground, from "use camelCase for variable names" to "never use global variables" to "never use exceptions." This project (google/styleguide) links to the style guidelines we use for Google code. If you are modifying a project that originated at Google, you may be pointed to this page to see the style guides that apply to that project.

This project holds the C++ Style Guide, C# Style Guide, Swift Style Guide, Objective-C Style Guide, Java Style Guide, Python Style Guide, R Style Guide, Shell Style Guide, HTML/CSS Style Guide, JavaScript Style Guide, TypeScript Style Guide, AngularJS Style Guide, Common Lisp Style Guide, and Vimscript Style Guide. This project also contains cpplint, a tool to assist with style guide compliance, and google-c-style.el, an Emacs settings file for Google style.

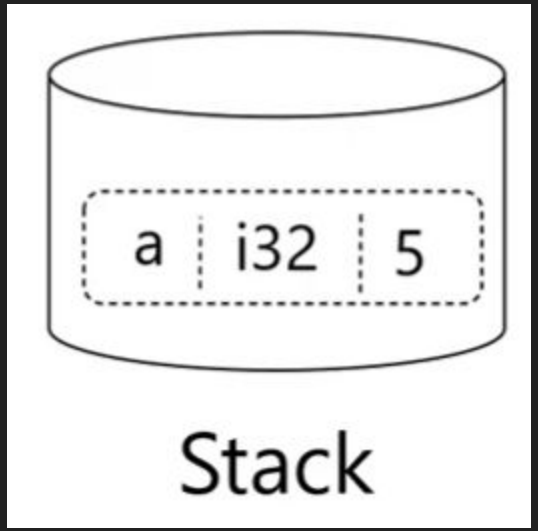

SCHMUCK
MONKEYUSER.COM
T'T
WHAT'S THIS CRAP?!?!
IT'S YOUR CODE, I JUST FORMATTED IT.


YOU SHALL NOT
MANUALLY FORMAT CODE
memegenerator.net

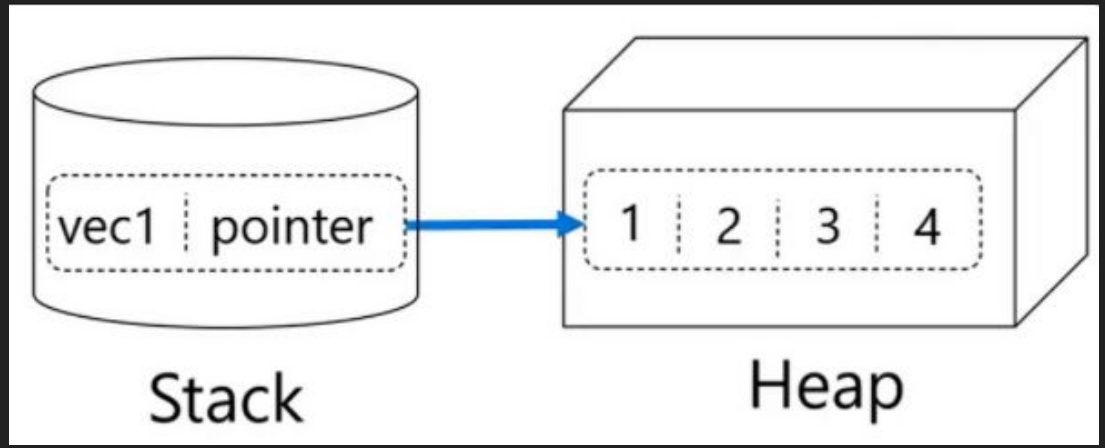
Ownership



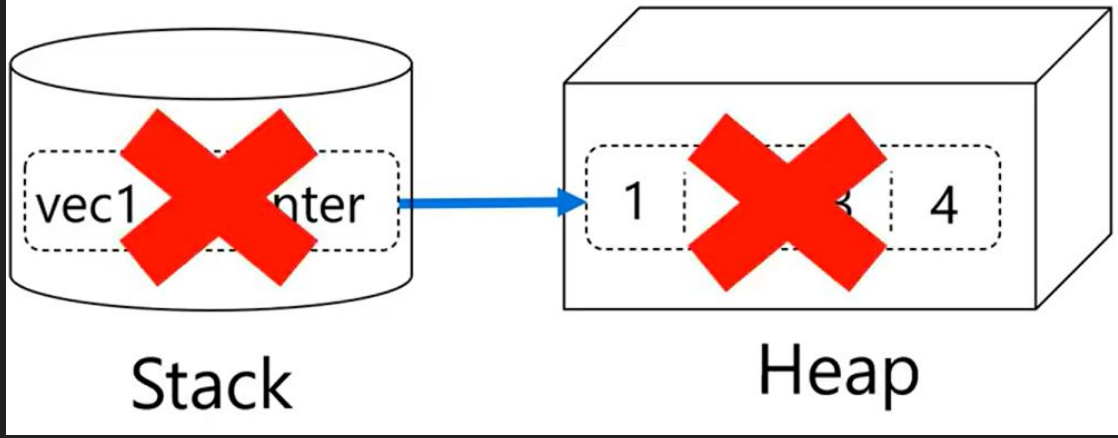
```
let a: i32 = 5;
```



```rust
// mut significa que puede cambiar su tamano mientras el programa corre
// mut = mutable
// por que puede mutar se almacena en el stack y en el heap
let mut vec1 = vec![1, 2, 3];
// si agrego lo siguiente mientras el programa corre veremos que
vec1.push(4);
println!("{:?}", vec1);
>> [1, 2, 3, 4]
```



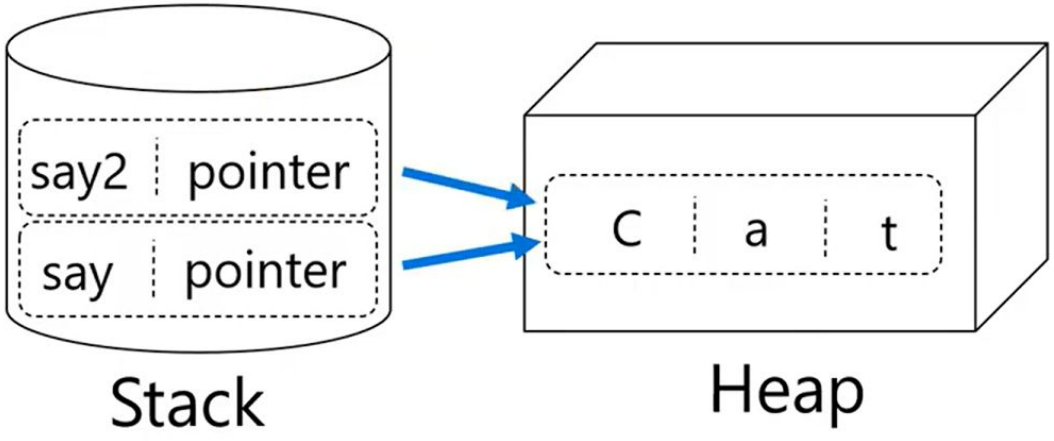
```
let mut vec1 = vec![1, 2, 3];

vec1.push(4);

println!("{:?}", vec1);

drop(vec1);
```



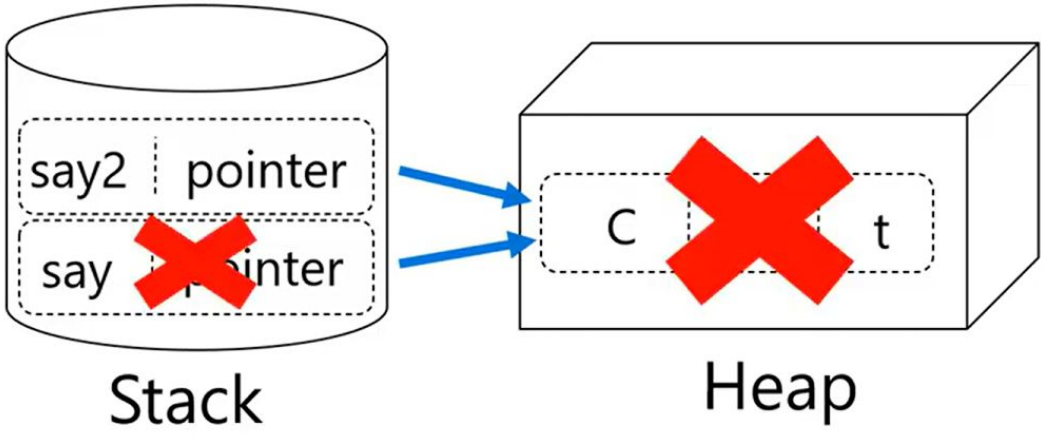
```
let mut say = String::from("Ca");

say.push_str("t");

println!(say);
```

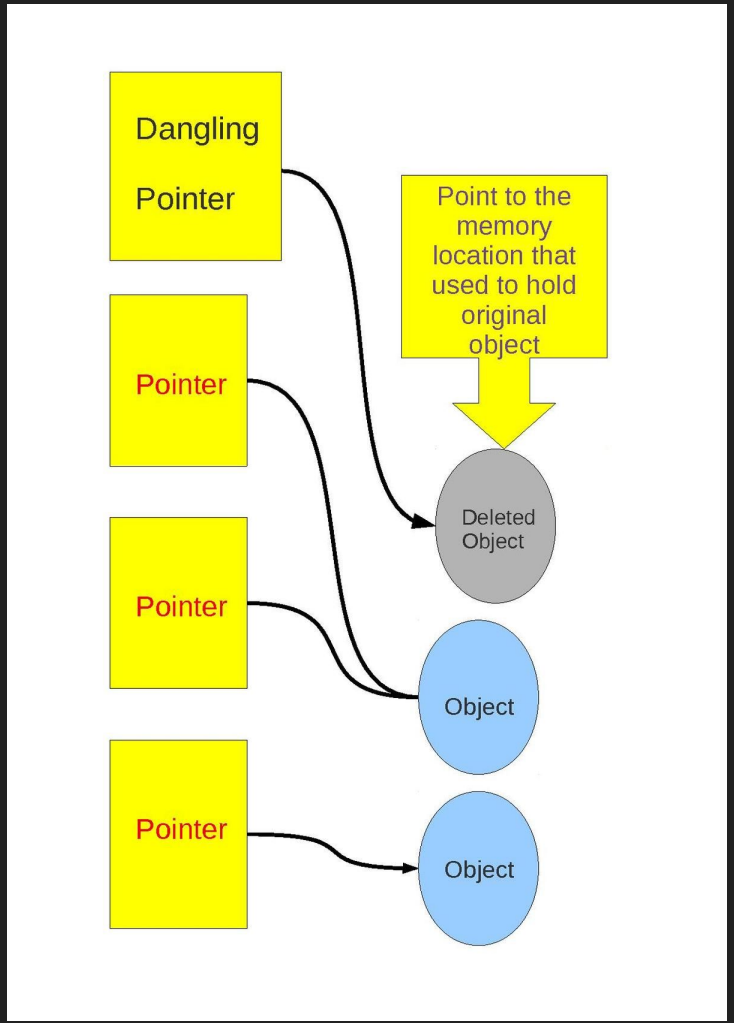
```
$ cargo run

Cat
```

```
let mut say = String::from("Ca");

say.push_str("t");

let say2 = say;
```



```
let mut say = String::from("Ca");

say.push_str("t");

let say2 = say;

drop(say);
```




Dangling
Pointer
Point to the memory location that used to hold original object
Pointer
Pointer
Pointer
Deleted Object
Object
Object

```
error[E0382]: borrow of moved value: `say`
 --> main.rs:5:20
  |
2 |     let mut say = String::from("He");
  |         ------- move occurs because `say` has type `String`, which does not implement the `Copy` trait
3 |     say.push_str("llo");
4 |     let say2 = say;
  |                --- value moved here
5 |     println!("{}", say);
  |                    ^^^ value borrowed here after move
  |
```

Compiler: Code compiled successfully 0 errors
RAM with a runtime error:

error C2664: 'void std::vector<block,std::allocator<_Ty>>::push_back(const block &)': cannot convert argument 1 from 'std::_Vector_iterator<std::_Vector_val<std::_Simple_types<block>>>' to 'block &&'

Puntero colgante

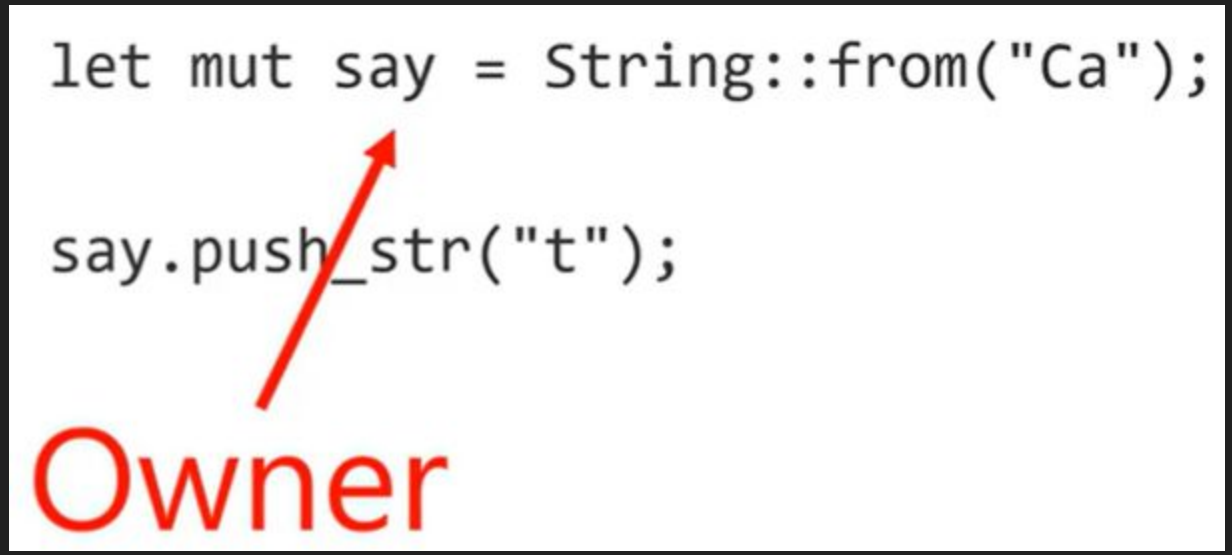
- C++ y C: Los programadores deben asignar y anular memoria.
- Python, JavaScript, Ruby: Recolector de basura.

In computer science, overhead is **any combination of excess or indirect computation time, memory, bandwidth, or other resources that are required to perform a specific task**.


GET THAT THING OUT OF MY FACE!
Me
Rust
chomp
@reactoverflow
Rust's ownership rule is awesome
chicken thoughts

Reglas de Ownership

Cada variable tiene un dueño

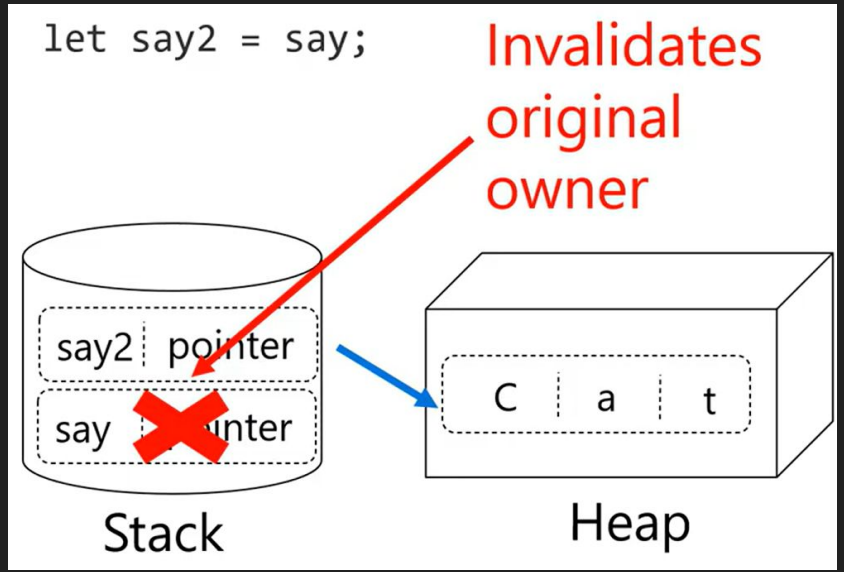


Solo puede haber un dueño por variable

Solo una variable puede apuntar al heap.

```
let mut say = String::from("Ca");

say.push_str("t");

let say2 = say;
```

Can only be one owner at a time



Qué problemas ayuda a evitar el ownership?

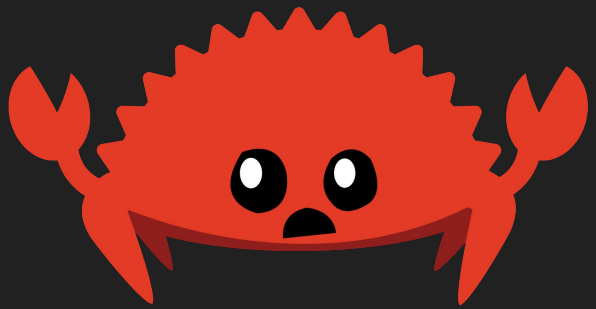
1. Puntero Colgante
2. Doble liberación: Liberar memoria que ya ha sido liberada
3. Fuga de memoria

~~NULL POINTERS~~

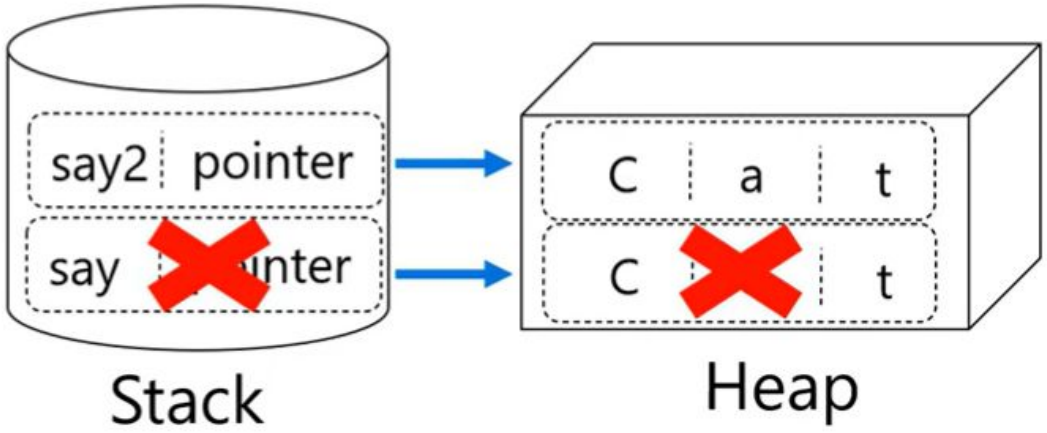
~~DANGLING POINTERS~~

~~RACE CONDITIONS~~

Borrowing



```
let say = String::from("Cat");

let say2 = say.clone();

drop(say);

println!(say2);
```

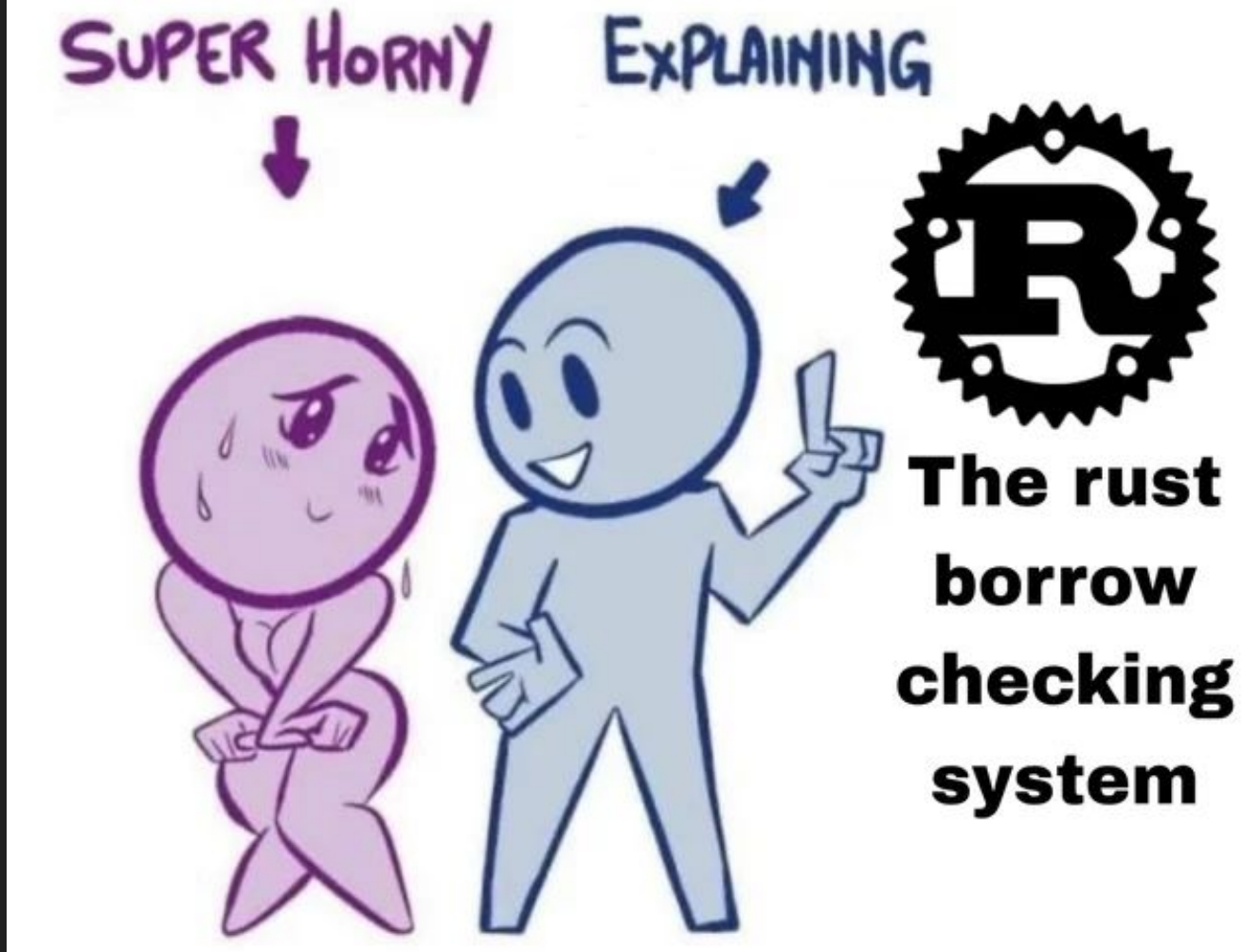


```
fn main() {
    let say = String::from("Cat");
    print_out(&say);
    println!("Again: {}", say);
}

fn print_out(to_print: &String) {
    println!("{}", to_print);
}
```

```
let say = String::from("Cat");
let say2 = &say;

println!("{}", say);

drop(say);

println!("{}", say2);
```

$ cargo run

ERROR


SUPER HORNY
EXPLAINING
The rust borrow checking system

```rust
use std::io;

▶ Run | Debug
fn main(){
    println!("What is your name?");
    let mut name: String = String::new();
    io::stdin().read_line(buf: &mut name).expect(msg: "Failed to read line");
    println!("Hello {}", name);
}
```

Consumo de Energía

Abstract

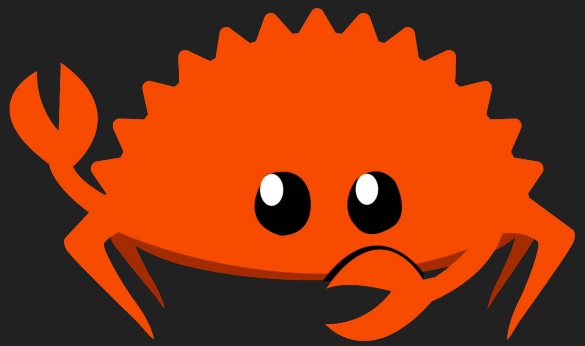
This paper presents a study of the runtime, memory usage and energy consumption of twenty seven well-known software languages. We monitor the performance of such languages using ten different programming problems, expressed in each of the languages. Our results show interesting findings, such as, slower/faster languages consuming less/more energy, and how memory usage influences energy consumption. We show how to use our results to provide software engineers support to decide which language to use when energy efficiency is a concern.

Table 4. Normalized global results for Energy, Time, and Memory

Total					
	Energy		Time		Mb
(c) C	1.00	(c) C	1.00	(c) Pascal	1.00
(c) Rust	1.03	(c) Rust	1.04	(c) Go	1.05
(c) C++	1.34	(c) C++	1.56	(c) C	1.17
(c) Ada	1.70	(c) Ada	1.85	(c) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(c) C++	1.34
(c) Pascal	2.14	(c) Chapel	2.14	(c) Ada	1.47
(c) Chapel	2.18	(c) Go	2.83	(c) Rust	1.54
(v) Lisp	2.27	(c) Pascal	3.02	(v) Lisp	1.92
(c) Ocaml	2.40	(c) Ocaml	3.09	(c) Haskell	2.45
(c) Fortran	2.52	(v) C#	3.14	(i) PHP	2.57
(c) Swift	2.79	(v) Lisp	3.40	(c) Swift	2.71
(c) Haskell	3.10	(c) Haskell	3.55	(i) Python	2.80
(v) C#	3.14	(c) Swift	4.20	(c) Ocaml	2.82
(c) Go	3.23	(c) Fortran	4.20	(v) C#	2.85
(i) Dart	3.83	(v) F#	6.30	(i) Hack	3.34
(v) F#	4.13	(i) JavaScript	6.52	(v) Racket	3.52
(i) JavaScript	4.45	(i) Dart	6.67	(i) Ruby	3.97
(v) Racket	7.91	(v) Racket	11.27	(c) Chapel	4.00
(i) TypeScript	21.50	(i) Hack	26.99	(v) F#	4.25
(i) Hack	24.02	(i) PHP	27.64	(i) JavaScript	4.59
(i) PHP	29.30	(v) Erlang	36.71	(i) TypeScript	4.69
(v) Erlang	42.23	(i) Jruby	43.44	(v) Java	6.01
(i) Lua	45.98	(i) TypeScript	46.20	(i) Perl	6.62
(i) Jruby	46.54	(i) Ruby	59.34	(i) Lua	6.72
(i) Ruby	69.91	(i) Perl	65.79	(v) Erlang	7.20
(i) Python	75.88	(i) Python	71.90	(i) Dart	8.64
(i) Perl	79.58	(i) Lua	82.91	(i) Jruby	19.84

	Energy consumed	Run-time
Imperative	125J	5585ms
Object-Oriented	879J	32965ms
Functional	1367J	42740ms
Scripting	2320J	88322 ms

Awesome Rust



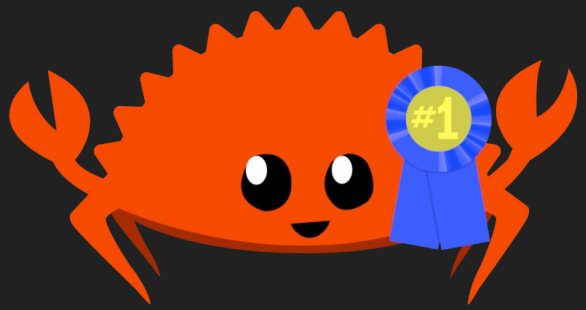
Awesome Rust

Rust passing | Track Awesome List

A curated list of Rust code and resources.

If you want to contribute, please read this.

Table of contents



Colecciones

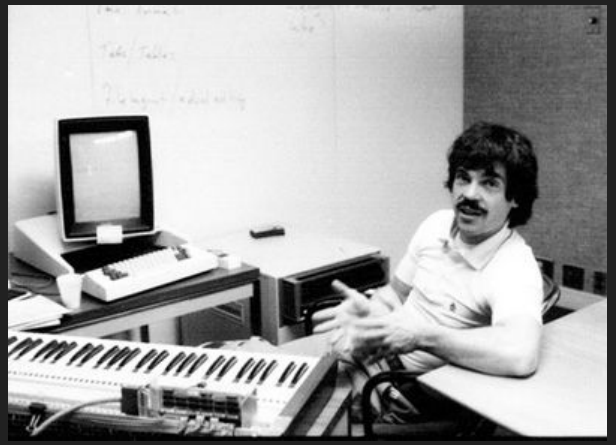
Rust's collections can be grouped into four major categories:

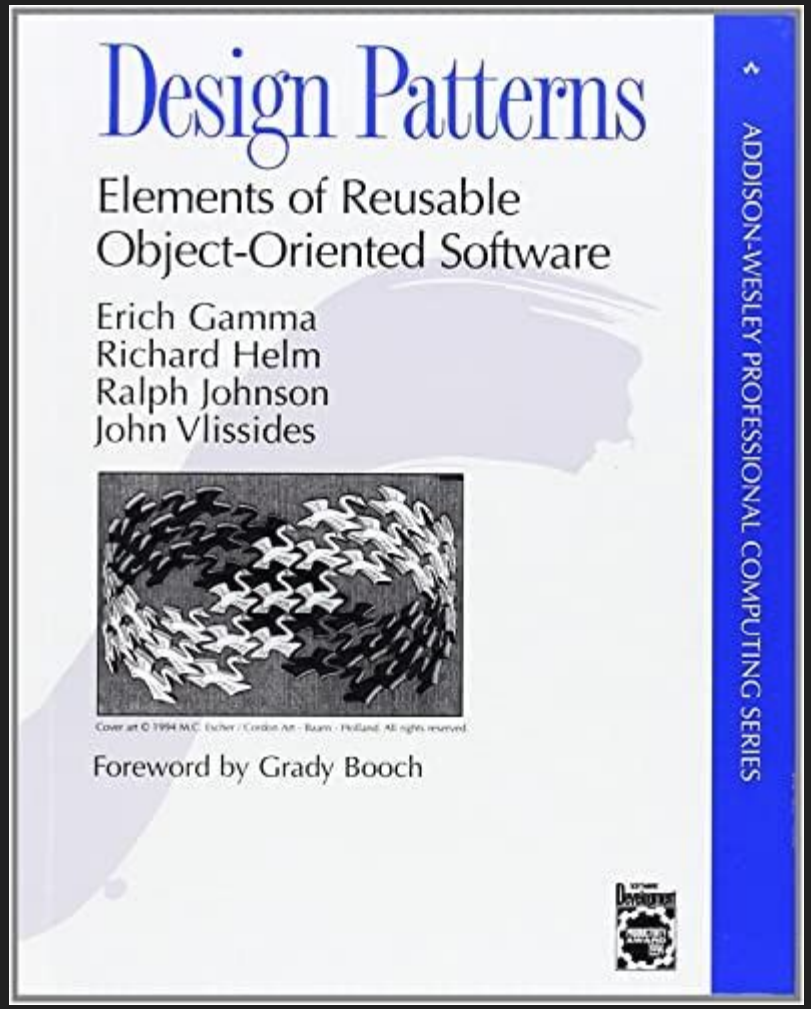
- Sequences: `Vec`, `VecDeque`, `LinkedList`
- Maps: `HashMap`, `BTreeMap`
- Sets: `HashSet`, `BTreeSet`
- Misc: `BinaryHeap`

```
use std::collections::LinkedList;

let list = LinkedList::from([1, 2, 3]);
```

```
use std::collections::HashMap;

let solar_distance = HashMap::from([
    ("Mercury", 0.4),
    ("Venus", 0.7),
    ("Earth", 1.0),
    ("Mars", 1.5),
]);
```

Programación Orientada a Objetos





Los programas orientados a objetos se componen de objetos. Un objeto contiene tanto los datos y los procedimientos que operan en los datos. Los procedimientos suelen llamarse métodos u operaciones.

```cpp
#include <iostream>
using namespace std;

class Room {
   private:
    double length;
    double breadth;

   public:
      void initData(double len, double brth) {
       length = len;
       breadth = brth;
    }

    double calculateArea() {
        return length * breadth;
    }
};

int main() {
    Room room1;
    room1.initData(42.5, 30.8);
    cout << "Area of Room =  " << room1.calculateArea() << endl;

    return 0;
}
```

Rust

```rust
struct Room {
    length: f64,
    breadth: f64,
}

impl Room {
    fn new() -> Self {
        Room {
            length: 0.0,
            breadth: 0.0,
        }
    }

    fn init_data(&mut self, length: f64, breadth: f64) {
        self.length = length;
        self.breadth = breadth;
    }

    fn calculate_area(&self) -> f64 {
        self.length * self.breadth
    }
}

▶ Run | Debug
fn main() {
    let mut room1: Room = Room::new();
    room1.init_data(length: 42.5, breadth: 30.8);
    println!("Area of Room = {}", room1.calculate_area());
}
```

```cpp
#include <iostream>
using namespace std;

class Animal {
   public:
    void eat() {
        cout << "I can eat!" << endl;
    }

    void sleep() {
        cout << "I can sleep!" << endl;
    }
};

class Dog : public Animal {
   public:
    void bark() {
        cout << "I can bark! Woof woof!!" << endl;
    }
};

int main() {
    Dog dog1;
    dog1.eat();
    dog1.sleep();
    dog1.bark();
    return 0;
}
```

```
obj counter(int i) {
    fn incr() {
        i += 1;
    }
    fn get() -> int {
        ret i;
    }
}

fn main() {
    auto c = counter(10);
    c.incr();
    log c.get();
}
```

Puntero a null en Rust

El diseño de un lenguaje de programación suele pensarse en términos de las características que se incluyen, pero las características que se excluyen también son importantes.

Yo lo llamo mi error del billón de dólares. En aquel momento, estaba diseñando el primer sistema de tipos completo para referencias en un lenguaje orientado a objetos. Mi objetivo era garantizar que todo uso de las referencias fuera absolutamente seguro, con una comprobación realizada automáticamente por el compilador. Pero no pude resistir la tentación de poner una referencia nula, simplemente porque era muy fácil de implementar. Esto ha llevado a innumerables errores, vulnerabilidades y caídas del sistema, que probablemente han causado mil millones de dólares y daños en los últimos cuarenta años.

El problema con los valores nulos es que si intentas utilizar un valor nulo como un valor no nulo, obtendrás un error de algún tipo. Debido a que esta propiedad nula o no nula es omnipresente, es extremadamente fácil cometer este tipo de error.

```
enum Option<T> {
    None,
    Some(T),
}
```

NOKIA Bell Labs

Que otros lenguajes están tratando de reemplazar a C++?

Carbon Language:
An experimental successor to C++

Why? | Goals | Status | Getting started | Join us

See our announcement video from CppNorth. Note that Carbon is not ready for use.

- JavaScript → TypeScript
- Java → Kotlin
- C++ → ***Carbon***

Mini Proyecto

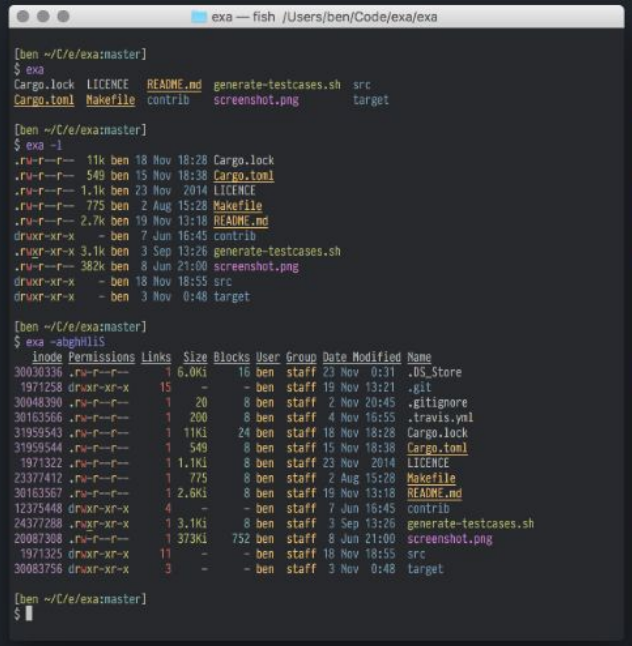

```
maverick@maverick-Inspiron-5548:~$ ls -1
1.c
a.out
ass8_1.c
binary.txt
cfile.c
c++file.cpp
cfile.o
cfile.so
client.c
Desktop
Documents
Downloads
end.txt
Exam
examples.desktop
FALCONN-1.2
fifo1.c
fifo2.c
first.txt
glove.cc
```

exa

exa is a modern replacement for *ls*.

README Sections: Options — Installation — Development

Unit tests passing | Say Thanks !



```
List of files:
foo.txt
bar.txt
foo1.txt
bar1.doc
foobar.txt
foo.doc
bar.doc
dataset.txt
purchase.db
purchase1.db
purchase2.db
purchase3.db
purchase.idx
foo2.txt
bar.txt
vivek@nixcraft-asus:/tmp$ grep 'purchase' demo.txt
purchase.db
purchase1.db
purchase2.db
purchase3.db
purchase.idx
```

```
~/minigrep$ cargo run 'ind' poema_i.txt
    Finished dev [unoptimized + debuginfo] target(s) in 0.03s
     Running `target/debug/minigrep ind poema_i.txt`
Perhaps I was blind to the facts, stabbed in the back
And in my mind I'ma blind man doin' time
Look to my future 'cause my past, is all behind me
But, it's my own kind doin' all the killin' here
```

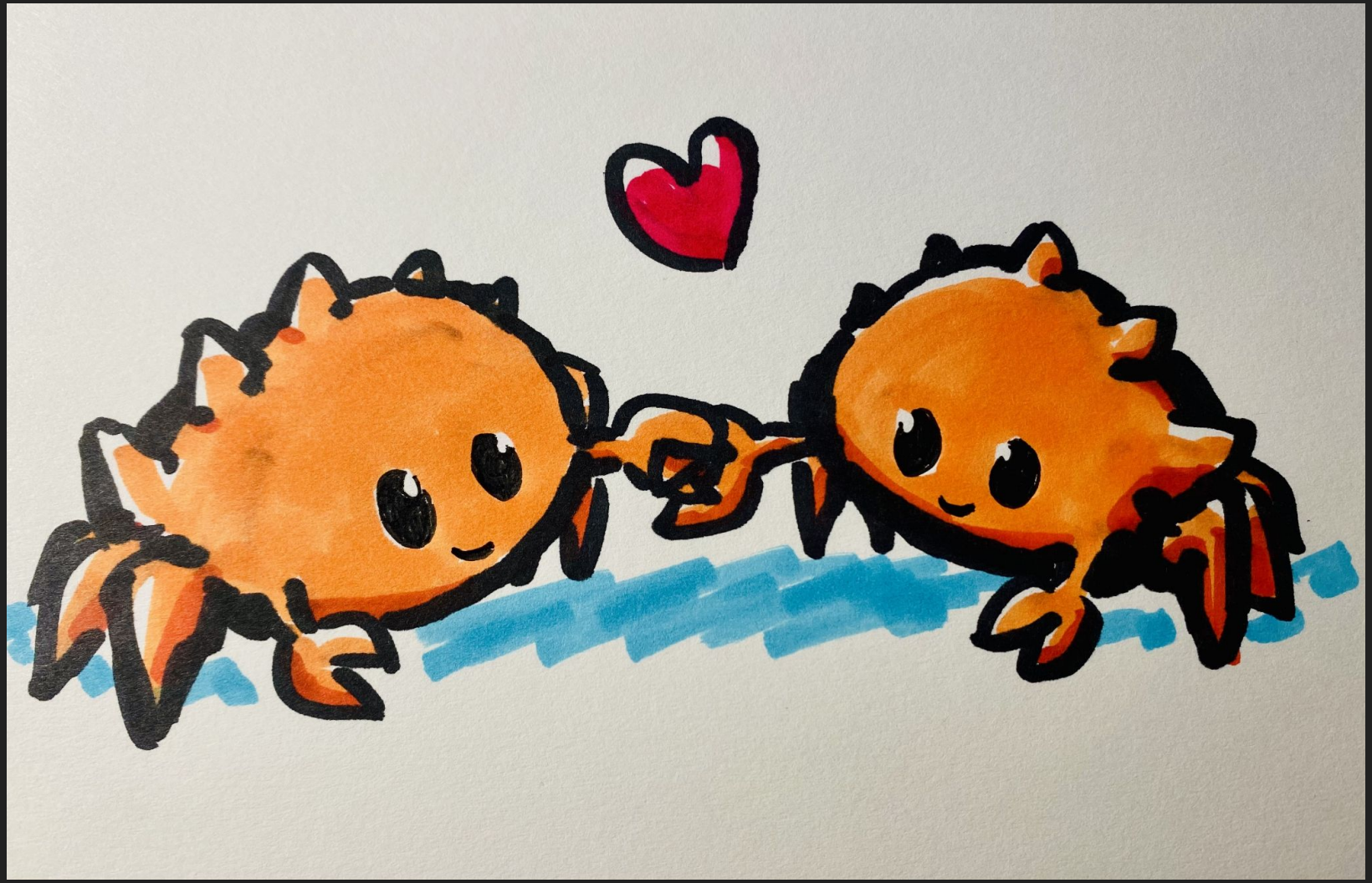
Rust posee un compilador bastante estricto

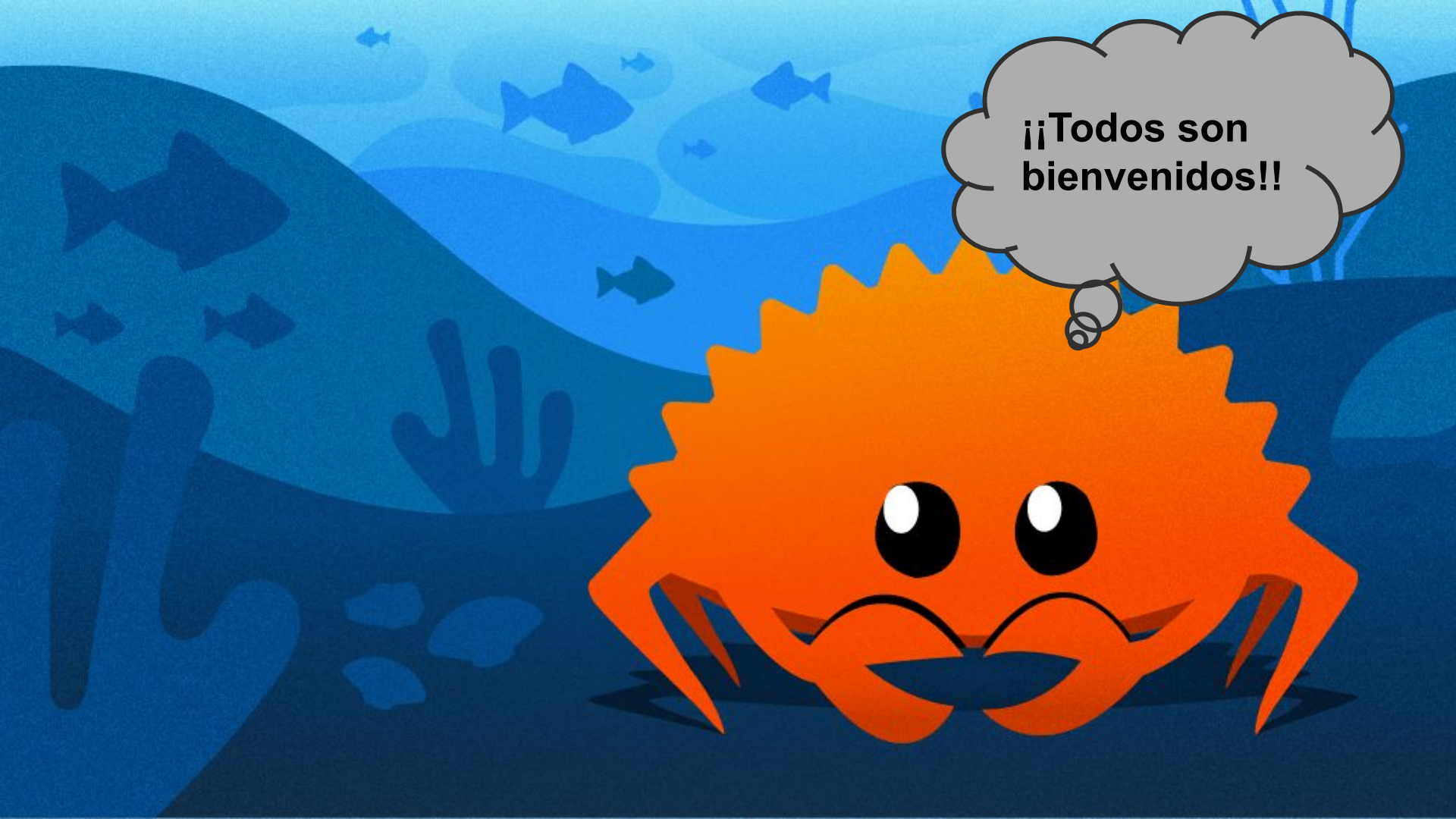
ABOUT RUST LATAM

Rust Latam Conference is the Latin America's leading event for and by the Rust movement, and one of Rust community's largest annual networking opportunities.

Two days of conference will offer interactive sessions, hands-on activities, and engaging talks to bring together 200+ passionate advocates, developers and enthusiasts of the Rust Programming Language from around the world. People and ideas from the Rust Latin America community will leap off the screen to learn, discuss, debate and address Rust in person.

Our second Rust Latam Conference is happening in Mexico City.




¡¡Todos son bienvenidos!!

Bibliografia

- *Build software better, together*. (n.d.). GitHub. https://github.com/collections/programming-languages
- *GitHub - rust-lang/rustlings: Small exercises to get you used to reading and writing Rust code!* (n.d.). GitHub. https://github.com/rust-lang/rustlings
- *GitHub - rust-lang/rust-by-example: Learn Rust with examples (Live code editor included)*. (n.d.). GitHub. https://github.com/rust-lang/rust-by-example
- *Experiment Introduction - The Rust Programming Language*. (n.d.). https://rust-book.cs.brown.edu/
- *I wonder, why Graydon Hoare, the author of Rust, stopped contributing into it and switched to Swift?* (2018, January 14). Reddit. https://www.reddit.com/r/rust/comments/7qels2/i_wonder_why_graydon_hoare_the_author_of_rust/
- *Energy efficiency across programming languages: how do energy, time, and memory relate?* (n.d.). ACM Digital Library. https://dl.acm.org/doi/10.1145/3136014.3136031
- Microsoft Developer. (2021, June 23). *Rust for Beginners*. YouTube. https://www.youtube.com/watch?v=PpWR6zungUk