A linguagem Rust e abstrações de alto nível



Brenno Lemos

- Syndelis
- @brenno@fosstodon.org







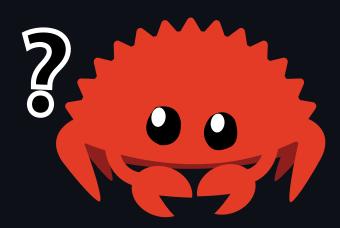
Antes de mais nada

Instale Rust e participe do live-coding

\$ curl https://sh.rustup.sh | sh

Por quê Rust?

- Padrão único de organização estrutural;
- Possui um gerenciador de pacotes oficial;
- Impossibilita* condições de corrida e vazamento de memória;
- É o inimigo № 1 do Segmentation Fault;



```
UFSJ S#inctude <stdlib.h>
    int main() {
        // Alocamos o vetor
         int *vec = (int*) malloc(
             50 * sizeof(int)
         );
         // Usamos o vetor...
         usa_vetor(vec);
         // Liberamos a memória
         free(vec);
```

</div> <div>

Rust

```
fn main() {
    // Alocamos o vetor
    let vec: Vec<i32> = Vec::new();
A linguagem Rust e abstrações de alto nível
    // Usamos o vetor
```

Índice - O que vamos aprender

- 1. A Sintaxe de Rust;
 - Comparando com C e Python;
- 2. Sistema de posse e empréstimo (ownership & borrowing system);
- 3. Estruturas e traços (structs & traits);
- 4. Implementação "cobertor" (blanket trait implementation);



```
<div class="columns"> <div>
```

- Similar ao C;
- Parênteses são opcionais e desencorajados;
- for genérico ao invés de numérico;
- return opcional na maioria dos casos;
- Tipagem pós-fixada ao invés de prefixada;
- Macros explícitos com ! ;

</div> <div style="font-size: 2em">

```
fn cinco_ou_maior(x: i32) -> i32 {
   if x > 5 { x } else { 5 }
}

fn main() {
   for i in 0..10 {
      println!(
        "{}",
        cinco_ou_maior(i)

A linguagem Rulte abstrações de alto nível
```