LAAS CNRS

Rust & calcul scientifique

Rencontre thématique Calcul Scientifique 2024, INRIA

2024-10-10

Guilhem Saurel

cnrs

Université Fédérale Toulouse Midi-Pyrénées

Figure 1: IR en robotique humanoïde

Rust

- Performance
- Fiabilité
- Productivité

Tooling

- Un compilateur bienveillant
- Un gestionnaire de paquet moderne
- Du formattage et de l'analyse statique

Tooling

- Un compilateur bienveillant
- Un gestionnaire de paquet moderne
- Du formattage et de l'analyse statique

Tout intégré et par défaut: Cargo

Tooling: drawback

Cargo n'est bon que pour le rust.

Dès qu'on doit se linker à autre chose, il faut un gestionnaire de paquet plus général.

Fearless concurency

Safe Rust guarantees an absence of data races, which are defined as:

- two or more threads concurrently accessing a location of memory
- one or more of them is a write
- one or more of them is unsynchronized

Compile-time Ownership

- plusieurs accès en lecture seule

XOR

- un seul accès en écriture

Compile-time Ownership

- plusieurs accès en lecture seule

XOR

- un seul accès en écriture

Bonus: `const` by default

Borrow checker: drawback

Learning curve...

Memory safety

"No way to prevent this" say users of only language where this regularly happens

— https://xeiaso.net/blog/ x 12

Idiomes

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

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

Results

```
let file = File::create("data.txt");
match file {
    Err(e) => eprintln!("cant create file: {e:?}"),
    Ok(mut f) => {
        let ret = f.write_all(b"hello");
        match ret {
            Err(e) => eprintln!("cant write: {e:?}")
            Ok(_) => println!("file written."),
        }
    }
}
```

Results: shortcut

```rust
fn main() -> std::io::Result<()> {
    let mut file = File::create("data.txt")?;
    file.write_all(b"hello")?;
    println!("file written.");
    Ok(())
}
```

Rayon

```rust
fn sum_of_squares(input: &[i32]) -> i32 {
    input.iter()
         .map(|&i| i * i)
         .sum()
}
```

Rayon

```rust
fn sum_of_squares(input: &[i32]) -> i32 {
    input.iter()
         .map(|&i| i * i)
         .sum()
}

fn sum_of_squares(input: &[i32]) -> i32 {
    input.par_iter()
         .map(|&i| i * i)
         .sum()
}
```

Des projets en rust

Tooling d'autres langages

- python (ruff, uv)
- javascript (deno, parcel, oxc)

Système

- linux
- coreutils
- shell utils

Un dernier drawback

Avoid dynamic libraries for now.

Aller plus loin

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

Aller plus loin

- https://doc.rust-lang.org/book/
- https://doc.rust-lang.org/nomicon/
- https://wiki.2rm.cnrs.fr/AnfRust2024

Inscriptions ouvertes jusqu'à 18h !

Questions

Contact

Matrix: @gsaurel:laas.fr
Mail: gsaurel@laas.fr

Cette présentation

`https://homepages.laas.fr/gsaurel/ rust-hpc.pdf`
`https://gitlab.laas.fr/gsaurel/talks : rust-hpc.md`