Rust : Structuration du code

Qualité de Développement — R5.A.08

C. Raïevsky

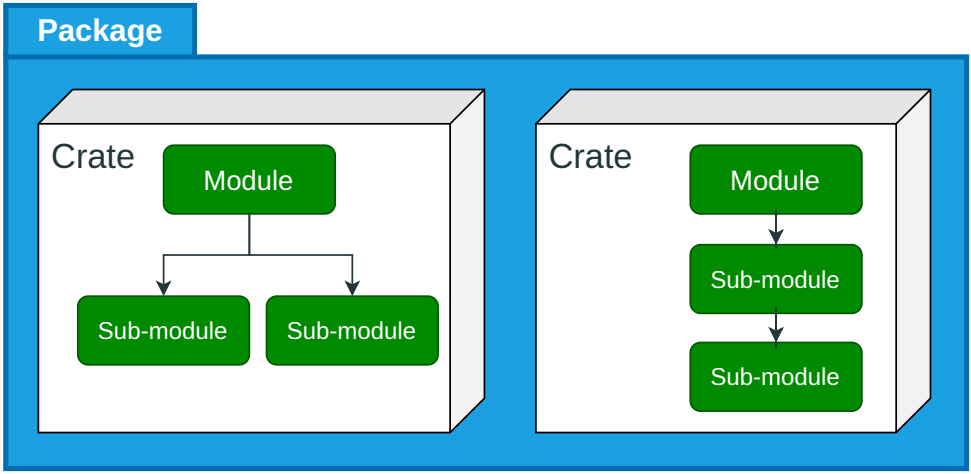
2023-2024

IUT A Valence
Université Grenoble Alpes

Département Informatique

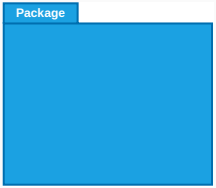
BUT Informatique 3ème année

Packages - Crates - Modules



Package

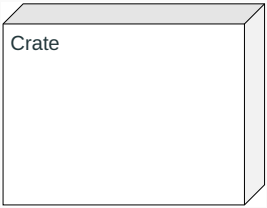
- Ce que crée `cargo new`
- Configuré dans un "Cargo.toml"
- Contient au moins un *crate*



La documentation

Crate

- Unité de compilation de Rust
- Constitué de modules
- Un fichier racine
- Potentiellement d'autres fichiers
- Deux types possibles : Exécutable ou bibliothèque

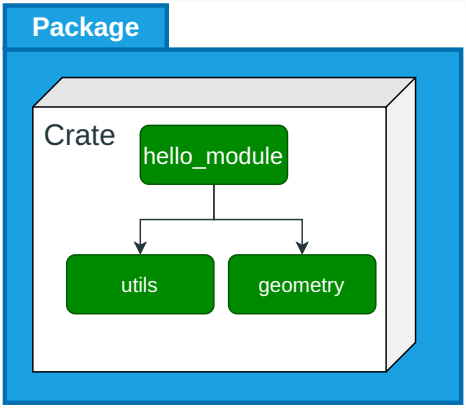


Fichier racine

- Binary crate (`cargo new`) → main.rs
- Library crate (`cargo new --lib`) → lib.rs

Ajout de modules à un crate

Ajout de modules : `mod`



Dans le fichier racine : **main.rs** ou **lib.rs**

```
1 mod utils;
2 mod geometry;
```

Code du module - Fichier ou répertoire

Fichier

Un fichier du nom du module :

```
hello
├── Cargo.lock
├── Cargo.toml
└── src
    ├── main.rs
    └── utils.rs
```

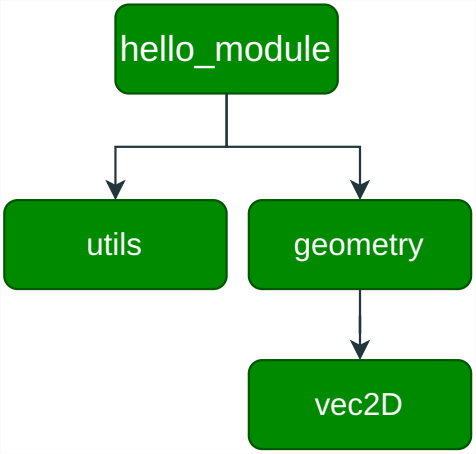
Répertoire

Un répertoire du nom du module et un fichier "`mod.rs`"

```
hello
├── Cargo.lock
├── Cargo.toml
└── src
    ├── main.rs
    └── utils
        └── mod.rs
```

Module

- Unités logicielles cohérentes
- Organisées en hiérarchie
- Contiennent
  - struct et/ou
  - fonctions associées



Imports : directive `use`

Pour utiliser un modules :

- `use nom_module;`
- Le module doit être visible

Il est possible de préciser un élément du module

- Une fonction du module : `use nom_module::nom_fonction;`
- Un type du module :
  - `use nom_module::NomType;`
  - `use nom_module::NomEnum;`

Imports

main.rs

```
1  mod utils;
2  use utils::print;
3  use utils::Point;
4  // These two lines can be replaced by:
5  // use utils::*;
6
7  fn main() {
8      let un_point = Point::origin();
9      print(un_point);
10 }
```

utils.rs

```
1  pub fn print(p: Point) {
2      println!("Le point: ({},{})", p.x,
3  }
4
5  pub struct Point {
6      x: f64,
7      y: f64,
8  }
9
10 impl Point {
11     pub fn origin() -> Self {
12         Self {x: 0.0, y:0.0}
13     }
14 }
```

Visibilité

Trois règles de base

- Un item est privé par défaut
- Un item publique est accessible au monde extérieur
- Un item privé n'est visible que
  - dans le module où il est déclaré
  - dans les sous modules