Zeste de savoir

La version stable de Rust 1.27.1 est désormais disponible !

22 mars 2019

Table des matières

1. Introduction

Rust est un langage de programmation système axé sur la *sécurité*, la *rapidité* et la *concurrence*.

Pour mettre à jour votre version stable, il suffit d'exécuter la commande habituelle.

```
$ rustup update stable
```

Si vous ne disposez pas de rustup, vous pouvez en obtenir une copie sur la page de téléchargement du site officiel. N'hésitez pas également à consulter la release note de la `1.27.1` sur GitHub !

2. Quoi de neuf ?

3. Patch : Faux positif sur les ressources empruntées

En Rust, les notions d'emprunt (borrowing) et de transfert (ownership) sont fondamentales et régissent la gestion des ressources. Ici, `foo` a l'*ownership* sur un objet `Foo`.

```
fn main() {
    let foo: Foo = Foo;
}
```

3. Patch : Faux positif sur les ressources empruntées

Nous pourrions nous en servir par le biais d'une référence `bar`, à quelques exceptions près.

```
##[derive(Debug)]
struct Foo;

fn main() {
    let foo: Foo = Foo;

    // On emprunte l'objet `foo` en lecture seule.
    let bar: &Foo = &foo;

    super_foo_factory(foo);

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

// Ici, la fonction prend un paramètre dont la ressource
// doit forcément être transférée.
fn super_foo_factory(f: Foo) -> Foo {
    /* ... On traite l'objet ... */
    f
}
```

```
error[E0505]: cannot move out of `foo` because it is borrowed
  --> src/main.rs:10:23
   |
8  |     let bar: &Foo = &foo;
   |                     --- borrow of `foo` occurs here
9  |
10 |     super_foo_factory(foo);
   |                       ^^^ move out of `foo` occurs here
```

Je pense qu'on ne peut pas faire plus explicite. Le compilateur est capable de différencier un emprunt (en l'occurrence `bar`) d'un transfert (que je n'ai pas illustré ici avec une autre variable, mais qui peu aisément l'être grâce au paramètre `f` de la fonction `super_foo_factory`).

Seulement, comme pour le dernier bug en date causé par une révision de l'expression `match`, le compilateur ne semblait plus différencier les deux notions.

```
fn main() {
    let a = vec![""].to_string()];
    a.iter().enumerate()
            .take_while(|(_, &t)| false)
            .collect::<Vec<_>>();
}
```

```
error[E0507]: cannot move out of borrowed content
    --> src/main.rs:4:30
     |
   4 |             .take_while(|(_, &t)| false)
     |                                  ^-
     |                                  ||
     |                                  |hint: to prevent move, use `ref t` or `ref mut t`
     |                                  cannot move out of borrowed content

error: aborting due to previous error
```

Ici, le compilateur nous renvoie une erreur affirmant que nous tentons de *transférer* une ressource *empruntée* et nous suggère d'effectuer un emprunt, ce qui est déjà le cas.

En réponse à cela, le problème a été corrigée dans la `1.27.1`.

i

Pour ceux qui se poseraient la question :

- `enumerate()` créé un itérateur contenant un tuple de deux valeurs (*index*, *valeur*) par élément ;
- `take_while()` créé un nouvel itérateur en accumulant les éléments matchant le prédicat (i.e. tant que le prédicat renvoie `true`, les éléments sont accumulés) ;
- `collect()` créé une nouvelle collection avec les éléments triés par la dernière méthode.

4. Patch : Exploitation possible du système de plugin de rustdoc

Ce 3 juillet, Red Hat a rapporté une vulnérabilité affectant le comportement de rustdoc. En effet, le système de plugin de rustdoc dispose d'un répertoire par défaut (i.e. `/tmp/rustdoc/plugins`) dans lequel les plugins seront chargés. Ce dernier pouvant être accédé en écriture, sans restrictions spécifiques sur la plupart des plateformes, il est possible d'y ajouter une bibliothèque dynamique pour ainsi exécuter du code arbitraire au lancement de l'outil.

Pour tenter de corriger progressivement la faille, il a été décidé que ce chemin par défaut serait supprimé et que l'utilisateur serait dans l'obligation de fournir *explicitement* un chemin dans lequel charger les plugins. L'équipe Rust prévoit de supprimer intégralement la fonctionnalité pour la version `1.28.0`.

5. Conclusion

6. Source

Le blog de l'équipe Rust

7. Voir aussi

7.1. Précédemment

— La version stable de Rust `1.27.0` est désormais disponible !

7.2. À suivre

— La version stable de Rust `1.27.2` est désormais disponible !