SVec

Programmation Générique en Rust

12 janvier 2026

Aujourd'hui le TP de rust est particulièrement difficile. On va descendre dans les profondeurs et utiliser *unsafe*.

On va implémenter quelques fonctions sur une structure de vecteur optimisée pour les petites tailles : le *SMallVec* de la crate *smallvec*. Cette structure a beaucoup d'utilisations pratiques car elle permet de gros gains de performances lorsqu'un programme doit manipuler de nombreux petits vecteurs. Elle est également connue pour une faille de sécurité découverte par un outil d'analyse statique en 2021.

Notre structure à nous, le *SVec* est définie de la manière suivante :

```rust
pub struct SVec<T> {
    empiles: [MaybeUninit<T>; 4],
    tas: Option<Box<[MaybeUninit<T>]>>,
    taille: usize,
    capacite: usize,
}
```

On voit ici que les 4 premiers éléments du vecteur sont stockés directement dans le *SVec* (potentiellement sur la pile) puis, les éléments suivants sont stockés dans le tas. Les gains de performance viennent du fait que si moins de 4 éléments sont utilisés alors la *Box* n'est pas allouée car l'*Option* reste à *None*. L'allocation mémoire n'aura lieu qu'au dernier moment, lorsqu'elle devient réellement nécessaire. La *taille* stocke le nombre d'éléments réellement stockés et la capacité le nombre d'éléments que l'on peut potentiellement contenir sans avoir besoin d'allouer plus de mémoire. On vous demande d'implémenter le trait *Debug*, les méthodes *push* et *pop* ainsi qu'un itérateur (réalisant un transfert de propriété).

Note : Le tp est *long*, le trait *Debug* et la méthode *push* devraient déjà beaucoup vous occuper. Pour le trait *Debug*, il est suggéré de passer par la méthode `debug_list().entries`. Attention, pour le *push*, en plus de la gestion pile/tas, il faut également considérer les redimensionnements du tableau contenu dans la *Box*, en sachant que les éléments du *SVec* de type *T* n'implémentent pas *Copy*. Pour allouer la boxe, il est suggéré de passer par `new_uninit_slice`. Lors du redimensionnement, vous aurez besoin de faire une copie mémoire, qui peut ici être efficacement implémentée via une copie bas-niveau entre deux tranches mémoires disjointes.