Brandon Myre et Antonii Spasov

Technologies émergentes

Rust

Analyse d’une technologie émergente

Travail présenté à

Sylvain Labranche

Département d’informatique

Collège Montmorency

15 décembre 2023

Contents

[Introduction 3](#_Toc151033250)

[Cargo et Crates 3](#_Toc151033251)

[Stack vs Heap 4](#_Toc151033252)

Antonii Spasov et Brandon Myre

Tp Synthèse

**Rust**

Introduction :

Rust est un tout nouveau langage de programmation, il est sorti le 15 mai 2015. Il a eu un grand coup popularité à cause de sa facilité à utiliser et de sa haute performance, il est facilement reconnaissable par son logo de crabe orange et il a été voté le langage de programmation le plus aimé dans les dernières années, en plus, d’être un des langages qui gagne le plus d’utilisation. Contrairement à plusieurs langages de programmation Rust n’utilise pas d’objet, il utilise des struct qui est essentiellement la même chose, mais qui contient certaines différences que nous couverons plus tard. Le gros attrait de Rust est sa rapidité à cause du contrôle de la mémoire comme C et C++, mais il est beaucoup plus sécure. Rust à certaines fonctionnalités qui permet de contrôler la mémoire, tout en gardant les risques de fuite de mémoire à un minimum, ce qui permet d’avoir des performances à environ 10% de C++ qui est le langage de programmation le plus rapide sur le marché. De plus en plus Rust est utilisé dans des OS, des programmes où que la sécurité est importante, des programmes avec beaucoup de données et des programmes qui a besoin de « scalability » et de concurrence. On peut utiliser Rust en créent un ficher .rs et le compilant avec rustc, mais il est plus populaire de l’utiliser dans un environnement cargo avec les « crates » ce qui ressemble au npm de javascript.

Cargo et Crates :

Stack vs Heap :

Le processeur a une très courte et petite mémoire. On peut dire, qu’il a très peu attentif. Peut-être qu’il peut se rappeler d’une valeur antérieure, mais il ne se souviendra pas certainement pas des détails d’une application complète. Quand un enfant effectue des exercices, il lui faut un espace pour y placer ses effets. Il lui faut un agenda ou une étagère. Ainsi, il sera capable de se souvenir quels devoirs il doit faire et à quelle place il a placer son ballon.

Dans le cas de notre processeur, on veut lui introduit une sorte bac à sable. Le bac a le seul but d’être un entrepôt de données. Maintenant, il est possible de stocker des données. Quand on obtient une valeur, elle est immédiatement entreposée dans le bac. Par la suite, elle est accessible pour d’autre manipulation. Maintenant, notre processeur possède ce que l’on appelle une mémoire.

Danger de la mémoire :

Cependant, cette mémoire n’est pas organisée. C’est-à-dire, qu’il n’y a pas d’ordre d’emplacement des choses. Si on met un ballon dans un coin, alors ce coin devient occupé. Si on écrit une phrase en début de cahier, alors cette espace devient occupée. Cependant, dans le cas où l’on écrit par-dessus cette phrase, alors on vient d’effacer la donnée précédente de cette partie de la mémoire. Maintenant, des mesures de contrôle d’organisation de la mémoire sont nécessaires. Des mesures de contrôleurs d’écriture et de lecture doivent être implémenté.

Une approche organisée (Stack)

Dans ce cas, on essaye de garder notre information en petit bout bien organiser pour former une file d’exécution. Le premier arrivé sera le premier à être exécuter. Tout comme dans une liste de choses à faire, on rogne l’élément fait et on passe au suivant.

Une approche anarchique (Heap)

Dans ce cas, on essaye de mettre les données de tailles variables dans un espace plus large sans fille d’attente.

En programmation rouillé, on peut utiliser les deux approches. Cependant, l’approche Stack et très souvent à préférer.

Désambiguation :

Stack =

Imaginez-vous une file d’attente d’une clinique médicale. Dans ce cas, Les patients sont acceptés au comptoir par ordre. On n’accepte qu’un client à la fois. Le premier qui est arrivé sera servi en premier. Le deuxième qui est arrivé sera servie en deuxième et ainsi de suite. Ainsi, un nouveau patient se met en arrière de la file d’attente. Le premier patient dans la file d’attente est envoyé au comptoir. Puis, le deuxième prend sa place.

Heap =

Imaginez-vous une urgence à l’hôpital.