Rust a 10 ans : comment un ascenseur en panne a changé le monde du logiciel pour toujours

Rust 1.0 a été livré en mai 2015. Voici comment cela s'est produit et pourquoi cela a marqué un tournant dans le monde du développement logiciel.

PAR STEVEN VAUGHAN-NICHOLS
Publié le 26/05/2025 à 17:56 | Mis à jour le 26/05/2025 à 17:57

5 min

© Elyse Betters Picaro / ZDNET

Eric S. Raymond, l'un des fondateurs de l'open-source, a déclaré : "**Tout bon logiciel commence par gratter au niveau de la démangeaison personnelle d'un développeur**". Ce fut certainement le cas de Graydon Hoare, un développeur de logiciels chez **Mozilla**, lorsqu'il a commencé à travailler sur **le langage de programmation Rust.**

En 2006, M. Hoare était agacé par **l'ascenseur de son immeuble qui ne cessait de tomber en panne**. Comme il l'a dit plus tard, "il est ridicule que nous, les informaticiens, ne soyons même pas capables de fabriquer un ascenseur qui fonctionne sans tomber en panne". Il soupçonnait que l'ascenseur tombait en panne à cause d'erreurs de mémoire dans son logiciel de contrôle, qui était probablement écrit en **C ou en C++**. Ces deux langages de systèmes populaires sont difficiles à coder, notamment parce qu'il est trop facile d'écrire un code semi-fonctionnel avec des erreurs de mémoire.

C'est pourquoi Hoare, qui en avait assez de se traîner tous les jours sur 21 étages, a commencé à concevoir un nouveau langage informatique. Il voulait créer un langage de programmation petit et rapide, sans risque d'erreurs de mémoire. Il l'a appelé **Rust**, d'après une famille de champignons très résistants qu'il a décrit comme "sur-ingéniée pour la survie".

Un tournant

Son objectif était de créer un langage sûr et **concomitant**. Contrairement à C et C++, Rust assure la sécurité de la mémoire grâce à son système de propriété unique. Il empêche les erreurs courantes telles que les débordements de mémoire tampon en s'assurant que chaque donnée a un propriétaire unique et qu'elle est automatiquement libérée lorsqu'elle sort du champ d'application. Cette approche permet d'éliminer des catégories entières de bogues de mémoire au moment de la compilation. Et Rust renforce encore la sécurité en détectant les courses de données avant l'exécution du code, ce qui permet aux développeurs d'écrire plus facilement des programmes à la fois sûrs et efficacement concomitants.

Cela n'a pas été facile. Alors qu'il s'agissait au départ d'un projet personnel, Mozilla a perçu le potentiel de Rust et a commencé à le sponsoriser officiellement en 2009. Le langage a

été annoncé publiquement en 2010, et après des années d'itération, **Rust 1.0 a été livré le 15 mai 2015**.

C'était il y a 10 ans.

La première version stable du langage de programmation Rust a discrètement marqué un tournant dans le monde du développement logiciel. Aujourd'hui, Rust n'est pas seulement une réussite technique, mais un exemple de la puissance de l'innovation communautaire, passant d'une expérience soutenue par Mozilla à un outil grand public adopté par les géants de la technologie et les communautés open-source.

Ce n'est qu'un début

La première version stable de Rust n'était qu'un début. Au cours de la décennie qui a suivi, le langage s'est développé à pas de géant. Le registre des paquets de Rust, crates.io, est passé d'environ 2 000 paquets ("crates") à la version 1.0 à plus de 180 000 aujourd'hui. La bibliothèque standard a triplé en taille, et la chaîne d'outils a mûri avec des fonctionnalités telles que rust-analyzer pour le support IDE et un gestionnaire de paquets, **Cargo**.

Dans le même temps, l'engagement de Rust en faveur de versions sans rupture et de cycles réguliers de six semaines a permis une innovation rapide sans sacrifier la fiabilité. Plus de 246 000 changements ont été fusionnés depuis la version 1.0, avec 6 700 contributeurs et près de 600 000 crates publics testés pour chaque version.

Hoare l'a récemment reconnu en déclarant : "**Rust est l'histoire d'une grande communauté de parties prenantes** qui se réunissent pour concevoir, construire, entretenir et développer une infrastructure technique partagée". Ces acteurs sont les développeurs, les concepteurs de langages, les auteurs et les formateurs, ainsi que les institutions qui soutiennent Rust. Ce qui les a réunis, a déclaré M. Hoare, c'est "un intérêt commun pour l'infrastructure".

"Le monde avait besoin d'une infrastructure robuste et fiable"

Par infrastructure, Hoare entend "un outil permettant de construire d'autres infrastructures : protocoles réseau, serveurs web, équilibreurs de charge, systèmes de télémétrie, bases de données, codecs, cryptographie, systèmes de fichiers, systèmes d'exploitation, machines virtuelles, interprètes, etc.

M. Hoare a ajouté : "Le monde avait besoin d'une infrastructure robuste et fiable. Et l'infrastructure dont nous disposions n'était pas à la hauteur de la tâche. En d'autres termes, elle tombait trop souvent en panne, de manière spectaculaire et coûteuse. Il existait des langages efficaces pour la construction d'infrastructures. Mais ils étaient très difficiles à utiliser et presque impossibles à utiliser en toute sécurité, en particulier lors de l'écriture de codes concomitants". Rust a été la réponse de Hoare.

J'appellerais cela de la programmation de systèmes.

La plomberie sous-jacente

Contrairement à d'autres langages populaires, tels que **Python**, **JavaScript** ou **Java**, Rust n'est pas destiné à écrire des programmes de haut niveau avec lesquels les utilisateurs finaux travaillent. Il est plutôt utilisé pour créer la plomberie sous-jacente dont tous les logiciels ont besoin pour fonctionner.

Ce n'est pas le genre de programmation que tout le monde pratique. Cependant, pour ceux qui travaillent avec les logiciels, les tuyaux et les raccords, Rust est très populaire. C'est ainsi que, **selon l'enquête Stack Overflow**, pour la huitième année consécutive, "Rust est le langage le plus admiré. Plus de 80 % des développeurs qui l'utilisent veulent l'utiliser à nouveau l'année prochaine".

J'avoue que j'en fais partie. Bien que je ne sois plus un développeur sérieux depuis des années, lorsque je code ces jours-ci, Rust est mon premier choix. Il fonctionne tout simplement, et je n'ai pas besoin de me préoccuper des détails de la mémoire comme je le faisais autrefois avec le C.

Je suis loin d'être le seul. Les **adoptants de Rust** ressemblent à un who's who de l'informatique moderne. Par exemple, Mozilla l'utilise dans Firefox, Google dans Android, Chrome OS et Fuchsia, et Microsoft dans ses bibliothèques Windows et Azure Confidential Compute.

Le noyau Linux, qui me tient le plus à cœur, intègre désormais Rust. Le voyage n'a pas été de tout repos. Comme l'a récemment déclaré Linus Torvalds, "je m'attendais à ce que les mises à jour de Rust soient plus rapides, mais une partie du problème réside dans le fait que les développeurs du noyau de longue date sont habitués au C et ne connaissent pas Rust. Ils ne sont pas vraiment enthousiastes à l'idée de devoir apprendre un nouveau langage qui est, à certains égards, très différent". Torvalds reste néanmoins un fervent partisan de **Rust-in-Linux.**

Apprendre Rust

Cela dit, l'utilisation et l'apprentissage de Rust ne se font pas sans heurts. Sa courbe d'apprentissage reste abrupte pour les nouveaux venus. Personnellement, je n'ai pas trouvé cela très difficile.

Si vous souhaitez apprendre Rust vous-même, je vous recommande de commencer par **The Rust Programming Language** (alias The Book), suivi de **Rust for Rustaceans**. Les sites web **Rust by Example** et **Welcome to Comprehensive Rust** de Google sont également utiles. Et gratuits de surcroît.

C'est ainsi que, dix ans après la sortie de la version 1.0, Rust a réalisé ce qui semblait autrefois impossible : rendre la programmation de systèmes à la fois sûre et agréable.
Source : "ZDNet.com"