# Préface

Rust est un langage de programmation système.

Cela mérite une explication de nos jours, car la programmation système n'est pas familière à la plupart des programmeurs en activité. Pourtant, cela sous-tend tout ce que nous faisons.

Vous fermez votre ordinateur portable. Le système d'exploitation le détecte, suspend tous les programmes en cours d'exécution, éteint l'écran et met l'ordinateur en veille. Plus tard, vous ouvrez l'ordinateur portable : l'écran et les autres composants sont à nouveau alimentés, et chaque programme est capable de reprendre là où il s'était arrêté. Nous tenons cela pour acquis. Mais les programmeurs système ont écrit beaucoup de code pour que cela se produise.

Programmation des systèmesest pour:

- Systèmes d'exploitation
- Pilotes de périphériques de toutes sortes
- Systèmes de fichiers
- Bases de données
- Code qui s'exécute dans des appareils très bon marché, ou des appareils qui doivent être extrêmement fiables
- Cryptographie
- Codecs multimédias (logiciels de lecture et d'écriture de fichiers audio, vidéo et image)
- Traitement des médias (par exemple, logiciel de reconnaissance vocale ou de retouche photo)
- Gestion de la mémoire (par exemple, implémentation d'un ramasse-miettes)
- Rendu du texte (la conversion du texte et des polices en pixels)
- Implémentation de langages de programmation de niveau supérieur (comme JavaScript et Python)
- La mise en réseau
- Virtualisation et conteneurs logiciels

- Simulations scientifiques

- Jeux

En bref, la programmation système est *limitée en ressources*programmation. C'est de la programmation lorsque chaque octet et chaque cycle CPU comptent.

La quantité de code système impliquée dans la prise en charge d'une application de base est stupéfiante.

Ce livre ne vous apprendra pas la programmation système. En fait, ce livre couvre de nombreux détails sur la gestion de la mémoire qui peuvent sembler inutilement abstrus au premier abord, si vous n'avez pas déjà fait de programmation système par vous-même. Mais si vous êtes un programmeur système chevronné, vous constaterez que Rust est quelque chose d'exceptionnel : un nouvel outil qui élimine les problèmes majeurs et bien compris qui ont tourmenté toute une industrie pendant des décennies.

# Qui devrait lire ce livre

Si vous êtes déjà un programmeur système et que vous êtes prêt pour une alternative au C++, ce livre est pour vous. Si vous êtes un développeur expérimenté dans n'importe quel langage de programmation, que ce soit C#, Java, Python, JavaScript ou autre chose, ce livre est également pour vous.

Cependant, vous n'avez pas seulement besoin d'apprendre Rust. Pour tirer le meilleur parti du langage, vous devez également acquérir une certaine expérience de la programmation système. Nous vous recommandons de lire ce livre tout en implémentant certains projets parallèles de programmation système dans Rust. Construisez quelque chose que vous n'avez jamais construit auparavant, quelque chose qui tire parti de la vitesse, de la simultanéité et de la sécurité de Rust. La liste des sujets au début de cette préface devrait vous donner quelques idées.

# Pourquoi nous avons écrit ce livre

Nous avons entrepris d'écrire le livre que nous aurions aimé avoir lorsque nous avons commencé à apprendre Rust. Notre objectif était

d'aborder les nouveaux grands concepts de Rust de front et de front, en les présentant clairement et en profondeur afin de minimiser l'apprentissage par essais et erreurs.

## Naviguer dans ce livre

Les deux premiers chapitres de ce livre présentent Rust et fournissent une brève visite guidée avant de passer aux types de données fondamentaux au chapitre 3 . Les chapitres 4 et 5 abordent les concepts fondamentaux de propriété et de références. Nous vous recommandons de lire ces cinq premiers chapitres dans l'ordre.

Les chapitres 6 à 10 couvrent les bases du langage : expressions ( chapitre 6 ), gestion des erreurs ( chapitre 7 ), crates et modules ( chapitre 8 ), structures ( chapitre 9 ) et énumérations et modèles ( chapitre 10 ). C'est bien de survoler un peu ici, mais ne sautez pas le chapitre sur la gestion des erreurs. Fais nous confiance.

Le chapitre 11 couvre les traits et les génériques, les deux derniers grands concepts que vous devez connaître. Les traits sont comme des interfaces en Java ou C#. Ils sont également le principal moyen pour Rust de prendre en charge l'intégration de vos types dans le langage lui-même. Le chapitre 12 montre comment les traits prennent en charge la surcharge d'opérateurs, et le chapitre 13 couvre de nombreux autres traits utilitaires.

Comprendre les traits et les génériques ouvre le reste du livre. Les fermetures et les itérateurs, deux outils puissants que vous ne voudrez pas manquer, sont traités respectivement dans les chapitres 14 et 15 . Vous pouvez lire les chapitres restants dans n'importe quel ordre, ou simplement y plonger au besoin. Ils couvrent le reste du langage : collections ( Chapitre 16 ), chaînes et texte ( Chapitre 17 ), entrée et sortie ( Chapitre 18 ), concurrence ( Chapitre 19 ), programmation asynchrone ( Chapitre 20 ), macros ( Chapitre 21 ), unsafe code ( Chapitre 22 ) et appeler des fonctions dans d'autres langages ( Chapitre 23 ).

## Conventions utilisées dans ce livre

Les conventions typographiques suivantes sont utilisées dans ce livre :

*Italique*

Indique de nouveaux termes, URL, adresses e-mail, noms de fichiers et extensions de fichiers.

`Constant width`

Utilisé pour les listes de programmes, ainsi que dans les paragraphes pour faire référence à des éléments de programme tels que des noms de variables ou de fonctions, des bases de données, des types de données, des variables d'environnement, des instructions et des mots-clés.

**`Constant width bold`**

Affiche les commandes ou tout autre texte qui doit être tapé littéralement par l'utilisateur.

*`Constant width italic`*

Affiche le texte qui doit être remplacé par des valeurs fournies par l'utilisateur ou par des valeurs déterminées par le contexte.

---

**NOTER**

Cette icône signifie une note générale.

---

# Utiliser des exemples de code

Du matériel supplémentaire (exemples de code, exercices, etc.) est disponible en téléchargement sur *https://github.com/ProgrammingRust* .

Ce livre est là pour vous aider à faire votre travail. En général, si un exemple de code est proposé avec ce livre, vous pouvez l'utiliser dans vos programmes et votre documentation. Vous n'avez pas besoin de nous contacter pour obtenir une autorisation, sauf si vous reproduisez une partie importante du code. Par exemple, écrire un programme qui utilise plusieurs morceaux de code de ce livre ne nécessite pas d'autorisation. La vente ou la distribution d'exemples tirés des livres d'O'Reilly nécessite

une autorisation. Répondre à une question en citant ce livre et en citant un exemple de code ne nécessite pas d'autorisation. L'incorporation d'une quantité importante d'exemples de code de ce livre dans la documentation de votre produit nécessite une autorisation.

Nous apprécions, mais nous ne demandons pas d'attribution. Une attribution comprend généralement le titre, l'auteur, l'éditeur et l'ISBN. Par exemple : « *Programming Rust, deuxième édition* par Jim Blandy, Jason Orendorff et Leonora FS Tindall (O'Reilly). Copyright 2021 Jim Blandy, Leonora FS Tindall et Jason Orendorff, 978-1-492-05259-3.

Si vous pensez que votre utilisation d'exemples de code ne respecte pas l'utilisation équitable ou l'autorisation donnée ci-dessus, n'hésitez pas à nous contacter à l' *adresse permissions@oreilly.com* .

# Apprentissage en ligne O'Reilly

---

**NOTER**

Depuis plus de 40 ans, *O'Reilly Media* fournit des formations, des connaissances et des connaissances en technologie et en affaires pour aider les entreprises à réussir.

---

Notre réseau unique d'experts et d'innovateurs partage leurs connaissances et leur expertise à travers des livres, des articles, des conférences et notre plateforme d'apprentissage en ligne. La plate-forme d'apprentissage en ligne d'O'Reilly vous donne un accès à la demande à des cours de formation en direct, des parcours d'apprentissage approfondis, des environnements de codage interactifs et une vaste collection de textes et de vidéos d'O'Reilly et de plus de 200 autres éditeurs. Pour plus d'informations, rendez-vous sur *http://oreilly.com* .

# Comment nous contacter

Veuillez adresser vos commentaires et questions concernant ce livre à l'éditeur :

- O'Reilly Media, Inc.
- 1005, autoroute Gravenstein Nord
- Sébastopol, Californie 95472
- 800-998-9938 (aux États-Unis ou au Canada)
- 707-829-0515 (international ou local)
- 707-829-0104 (télécopieur)

Nous avons une page Web pour ce livre, où nous énumérons les errata, des exemples et toute information supplémentaire. Vous pouvez accéder à cette page à *https://oreil.ly/programming-rust-2e* .

Envoyez un e-mail à *bookquestions@oreilly.com* pour commenter ou poser des questions techniques sur ce livre.

Visitez *http://www.oreilly.com* pour plus d'informations sur nos livres et cours.

Retrouvez-nous sur Facebook : *http://facebook.com/oreilly*

Suivez-nous sur Twitter : *http://twitter.com/oreillymedia*

Regardez-nous sur YouTube : *http://youtube.com/oreillymedia*

# Remerciements

Le livre que vous tenez a grandement bénéficié de l'attention de nos réviseurs techniques officiels : Brian Anderson, Matt Brubeck, J. David Eisenberg, Ryan Levick, Jack Moffitt, Carol Nichols et Erik Nordin ; et nos traducteurs : Hidemoto Nakada (中田 秀基) (japonais), M. Songfeng Li (chinois simplifié) et Adam Bochenek et Krzysztof Sawka (polonais).

De nombreux autres réviseurs non officiels ont lu les premières ébauches et ont fourni des commentaires inestimables. Nous tenons à remercier Eddy Bruel, Nick Fitzgerald, Graydon Hoare, Michael Kelly, Jeffrey Lim, Jakob Olesen, Gian-Carlo Pascutto, Larry Rabinowitz, Jaroslav Šnajdr, Joe Walker et Yoshua Wuyts pour leurs commentaires judicieux. Jeff Walden et Nicolas Pierron ont été particulièrement généreux de leur temps, révi-

sant presque tout le livre. Comme toute entreprise de programmation, un livre de programmation se nourrit de rapports de bogues de qualité. Merci.

Mozilla a été extrêmement accommodant avec le travail de Jim et Jason sur ce projet, même s'il ne relevait pas de nos responsabilités officielles et leur faisait concurrence pour attirer notre attention. Nous sommes reconnaissants aux managers de Jim et Jason : Dave Camp, Naveed Ihsanullah, Tom Tromey et Joe Walker, pour leur soutien. Ils ont une vision à long terme de ce qu'est Mozilla ; nous espérons que ces résultats justifient la confiance qu'ils ont placée en nous.

Nous tenons également à exprimer notre gratitude à tous ceux d'O'Reilly qui ont contribué à la réalisation de ce projet, en particulier nos éditeurs étonnamment patients Jeff Bleiel et Brian MacDonald, et notre éditeur d'acquisitions Zan McQuade.

Surtout, nos sincères remerciements à nos familles pour leur amour indéfectible, leur enthousiasme et leur patience.