+33 (0)1 48 10 09 46
formation@ambient-it.net
140 Avenue Jean Lolive 93500 Pantin - France

Mis à jour le 06/03/2025

S'inscrire

Formation Rust

4 jours (28 heures)

Présentation

Notre formation Rust vous permettra de découvrir ce nouveau langage de programmation Open Source, conçu pour aider les développeurs à concevoir des applications ultra-rapides et sécurisées.

Avec ce cours, vous apprendrez à utiliser toutes les ressources qu'offre Rust, telles que le haut niveau d'abstractions, la sécurité garantie de la mémoire, ou encore des templates comme le trait-based generics, le Pattern matching, ainsi que l'inférence de type.

Grâce à notre formation, Rust, apprenez à éviter les erreurs de segmentation tout en assurant la sécurité des Threads. Vous pourrez créer des programmes qui ont la performance et le contrôle d'un langage de bas niveau tel que le C, avec la puissance d'un langage de haut niveau.

Comme dans toutes nos formations, celle-ci vous présentera la toute dernière version de Rust Programming Language, Rust 1.85.

Objectifs

- Découvrir, comprendre et utiliser le langage Rust
- Utiliser la programmation concurrente et la bonne gestion d'erreur avec Rust
- Interfacer Rust avec un autre langage (Python par exemple)
- Contrôler les flux basés sur la correspondance de motifs

Public visé

- Développeurs
- Architectes

Pré-requis

- Connaissances de base d’un système Unix
- Connaissance de programmation (C, C#, Java)

Pré-requis logiciels

- Installer Docker et Docker Compose
- Installation de gnuplot

Recommandations de lecture avant et après la formation

- Un article qui montre étude de cas sur l'écriture d'une application Rust avec seulement qu'une API Minimale
- Cet article présente le langage de programmation Rust ainsi que les raisons de sa popularité
- Une vidéo explicative en quelques minutes du langage de programmation Rust crée par Fireship

Programme de la formation Rust

Chaque chapitre contient ses propres exercices et chaque partie sera suivie d'un ou de plusieurs travaux pratiques.

Introduction à Rust

- Présentation du langage Rust
- Téléchargement et installation de l'environnement Rust
  - rustup
  - cargo
  - rustc
  - crates.io
- Mon premier programme
- Trouver la documentation
- Éditeurs pour Rust

LES BASES DE RUST

- Types basiques
  - Types numériques, entiers et ottants
  - Booléens
  - Caractères
  - Chaînes de caractères
- Types composés
  - Tableaux
  - Couples

- Variables et mutabilité
  - Notion de mutabilité
  - Constantes
  - Redénition d'une variable et shadowing
  - Variables statiques
- Fonctions
  - Syntaxe des fonctions
  - Instructions et expressions
- Blocs et portée
  - Blocs nécessaires
  - Blocs et durée de vie
  - Bloc d'assignation
- Structures de contrôle
  - if, else et else if
  - Boucle loop
  - Boucle while
  - Boucle for
- Espace de noms
  - Notion de crate
  - Imports depuis la std
  - Alias de type
- Tests unitaires : la commande cargo test

Les types de données composites

- Structures
  - Dénition
  - Assignation
  - Utilisation
  - Structures Tuple
  - Structures unitaires
- Énumérations
  - Forme commune à d'autres langages
  - Énumération avec données associées
- Implémentations de trait
  - Traits derivables
  - Implementations manuelles
- Méthodes
  - Constructeurs
  - Méthodes non-mutables
  - Méthodes mutables

Contrôles de flux basés sur la correspondance de motifs

- if let
  - Test du motif d'une structure
  - Test d'un variant d'une énumération
  - Opérateur OR et motifs multiples
  - Test d'une range
  - Motifs incomplets : introduction à la déstructuration
- while let

- match
  - Différence avec le switch case
  - Utilisation de la déstructuration
  - Ajout d'expression booléennes

Les types pour la gestion des erreurs

- Option
  - Prototype de Option
  - Création et utilisation d'une Option
  - Méthodes de la std sur Option
  - Aller plus loin avec les Options
- Result
  - Prototype de Result
  - Création et utilisation d'un Result
  - La gestion des erreurs

Concepts clefs du langage Rust

Ici, il n'est plus question de bases ; ce module servira à approfondir ou à introduire des concepts propres au langage Rust.

- Expressions
  - Initialisation conditionnelle d'une variable
  - Utilisation du pattern matching et expressions
  - Bloc d'expression
- Possession
  - Transfert de possession ou copie
  - Reprendre la possession
  - Problème de la modification locale
- Inférence de type
  - Différents cas d'inférence
  - Conflit de type
  - Limitations de l'inférence de type
- Références, emprunt et pointeurs
  - Références
  - Références implicites et sucre syntaxique
  - Mutabilité et exclusivité
  - Raw pointers
- Introduction à la durée de vie
  - Durées de vie implicites
  - Quand il faut être explicite
  - Lifetime static
- Type slice
  - Dénition
  - Utilisation
  - Itération
  - Méthodes
  - Accès mutables
- Généricité
  - Librairie standard et génériques
  - Type générique dans une structure ou une énumération

- Traits
  - Le trait From
  - Implémentation de From pour une énumération
- Panic, Safe et Unsafe Rust
  - Code unsafe
  - Prototype de panic

L'allocation dynamique

- Box
  - Introduction, pourquoi allouer dynamiquement ?
  - Box et Trait Deref
  - Cas d'utilisation d'une Box
- Collections - Vector et HashMap
  - Vectors : Tableau alloué dynamiquement
  - Hashmap : Dictionnaire
- Type String
  - Une chaîne de caractère allouée dynamiquement
  - Création de String
  - Méthodes de String
  - String et type &str
  - À propos de From et de Into
- Arc, compteurs atomiques de référence
  - Caractéristiques des Arc
  - Proof of concept
- À propos du trait Clone
  - Dérivation de Clone
  - Dérive récursif
  - Implémentation obligatoire de Clone pour Copy

La Programmation Fonctionnelle

Rust combine des aspects de programmation fonctionnelle, tels que les fermetures et l'immutabilité, enrichissant son modèle impératif et concurrent.

- Opérateur Iry
  - Introduction
  - Syntaxe det eet
- Closures
  - Différence avec une fonction, la capture de l'environnement
  - Capture par référence
  - Transfert de possession
- Qu'est-ce que l'approche fonctionnelle
  - Présentation des Iterateurs
  - Principe interne de l'itérateur
  - Exemple d'implémentation d'un Iterateur "custom"
  - Les méthodes sur les Iterateurs
  - Quelques exemples d'utilisation des itérateurs
- Itérateurs, mutabilité, exclusivité et RefCell
  - Un problème de borrow checker
  - Contournement grace à RefCell

La programmation concurrente

Rust excelle en programmation concurrente grâce à son système de propriété et d'emprunt, assurant la sécurité des threads sans verrouillage explicite.

- Garanties grâce aux traits Send et Sync
- Partage d'une ressource entre plusieurs threads
  - Utilisation de Arc
  - Mutex, lock et unlock
- MPSC ou Multiple Producer Single Consumer
  - Qu'est-ce qu'un MPSC
  - Initialisation
  - Création des threads et utilisation

Création de trait

- Définition d'un trait
- Polymorphisme statique
- Règle de l'orphelin

FFI

- Appel du Rust depuis C/C++
- Appel du C/C++ depuis Rust

Arborescence de projet et création de crate

- Utilisation d'une crate
  - Consultation de crates.io
  - Intégration d'une crate au projet
  - La documentation des crates
  - Utilisation des crates dans vos projets Rust
- Création d'une librairie
  - Création d'un boilerplate par defaut avec cargo
  - Renseignement du fichier cargo.toml
  - Utilisation de la librairie
- Fichiers, Dossiers et Visibilité
  - Ajout de fichier
  - Sous-dossiers, présentation des deux techniques

Module complémentaire (+1 jour) : Approfondissement

Les bases de la programmation concurrente

- Définition : Mandelbrot
- Parsing pair : command-line arguments
- Mapping pixels : complex numbers
- Tracer : Plotting the set

- Écriture d'un fichier image
- Mandelbrot : Concurrent program
- Exécuter : Mandelbrot plotter
- Sécurité

Unsafe code

- Unsafe blocks
- Exemple : ASCII string type
- Unsafe functions, traits, Raw pointers
- Exemple: RefWithFlag
- Nullable pointers
- Type sizes & alignments
- Pointer arithmetic
- Moving into and out of memory
- Exemple : GapBuffer
- Interop : Foreign functions, calling function lib C & C++
- Common data representations
- Déclaration : foreign functions & variables
- Raw interface libgit2, Safe interface libgit2

Pour aller plus loin

Sociétés concernées

Cette formation s’adresse à la fois aux particuliers ainsi qu’aux entreprises, petites ou grandes, souhaitant former ses équipes à une nouvelle technologie informatique avancée ou bien à acquérir des connaissances métiers spécifiques ou des méthodes modernes.

Positionnement à l'entrée en formation

Le positionnement à l'entrée en formation respecte les critères qualité Qualiopi. Dès son inscription définitive, l'apprenant reçoit un questionnaire d’auto-évaluation nous permettant d’apprécier son niveau estimé sur différents types de technologies, ses attentes et objectifs personnels quant à la formation à venir, dans les limites imposées par le format sélectionné. Ce questionnaire nous permet également d’anticiper certaines difficultés de connexion ou de sécurité interne en entreprise (intraentreprise ou classe virtuelle) qui pourraient être problématiques pour le suivi et le bon déroulement de la session de formation.

Méthodes pédagogiques

Stage Pratique : 60% Pratique, 40% Théorie. Support de la formation distribué au format numérique à tous les participants.

Organisation

Le cours alterne les apports théoriques du formateur soutenus par des exemples et des séances de réflexions, et de travail en groupe.

Validation

À la fin de la session, un questionnaire à choix multiples permet de vérifier l’acquisition correcte des compétences.

Sanction

Une attestation sera remise à chaque stagiaire qui aura suivi la totalité de la formation.