SecHack365

開発駆動 仲山ゼミ
川端 薫

Rustに向けたmruby VM

“Ruri”

~小さいRuby(mruby)をRustに載せよう！~

要旨

1. プラグインシステム、設定ファイル、テンプレート処理など、動的要素を添加するために使う言語は意外と沼りがちで、さらにRustはそもそも選択肢が少ない
2. mruby(Ruby)は書き味的に（実は）Rustと相性がいい＋記法が柔軟で内部DSLが作りやすいため、設定データの記述からゴリゴリのスクリプト記述まで対応できる
3. Ruriは、Rustプログラムと連携しやすいmruby実行環境を、バイトコードVMを純粋なRustでゼロから実装することで実現し、1の課題への選択肢の一つを示した

mruby: 省リソース環境にも実装できるRuby派生言語

mrubyは、まつもとゆきひろ氏により開発が始まった軽量なRuby派生言語（通常のRubyとは異なる）
Rubyの書き味を保ちつつ、省リソース環境含む
さまざまな環境での利用を想定した設計がなされている
「Rubyっぽいけど結局機能が足りない」といった類のものにしないことを明示的に目標に掲げている

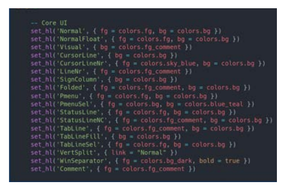
背景

なお、似た現象は数十年前からから言われ続けている（cf. グリーンスパンの第10法則）

ソフトウェアの規模が大きくなってくると、実行時に挙動を変えられるような動的要素を足したくなりがち

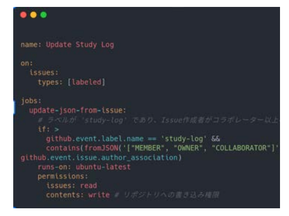
- ユーザーによる設定をできるようにしたい
- ちょっとしたテンプレート機能を入れたい
- ちいさなプラグインを書けるようにしたい
- ある言語の強力なライブラリをより軽量な言語経由で呼び出したい

しかし、安易に動的な要素を組み込むと破滅する！



Luaなどのスクリプト言語は
強力だが冗長

データ記述言語は
ロジックを入れるとカオスになる
（例：GitHub Actions）



独自DSLは
開発・維持・学習コストが高すぎる...
（玉石混交になりがち）

結局、強力すぎる/弱すぎる道具で、常に微妙なストレスを感じつつ記述し続けることになりがち

パワフルで書きやすい＋汎用的＋特定用途での簡潔な記述ができる柔軟性がある処理系がほしい！

が、現状それを満たす選択肢がなかなかない！
特にRust向けはLua系や独自言語以外の選択肢にかなり乏しい

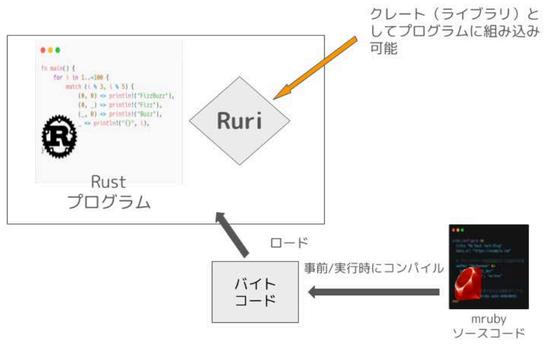
実際の利用例；
- ネットワーク機器の設定
- ゲームのシナリオの動的な記述
- ホットリロード可能なファームウェア(!?)

など

Ruriのアプローチ

Rustへ組み込めるmrubyの処理系（VM部分）を実装し、

Rustプログラムにクレート（ライブラリ）を追加するだけでmrubyバイトコードを実行可能に



RustxRubyの相性

- （実は）Rubyは埋め込み言語としてかなり優秀
  - 書きやすいことに加え、ブロック構文等で記法を柔軟に変更可能（DSL的簡潔さが実現できる）
  - さらに式指向など、埋め込み言語として嬉しい機能がかなり多い
  - 処理系が巨大すぎる問題はあるが、mrubyでは解決されている
- Rustは堅牢・可搬かつエコシステムが手厚い
  - 幅広い分野で充実したエコシステムが利用可能
  - cargo docによりドキュメントまわりもかなり体験がよい」
  - WASM等への対応が強い

Key Features

1. **設定・埋め込み用途で簡潔できれいな記述ができる**
   a. Ruby由来の柔軟性により実現
2. **ホスト言語と連携し、強力なエコシステムが利用できる**
   a. 純粋なRustで実装し、Rust連携が容易
3. **汎用言語としての十分な表現力**
   a. mrubyはRubyの表現力面でのパワフルさを継承しており、スクリプト言語としてパワフル

Rustの堅牢さ(＋厚いエコシステム)にRubyの柔軟さを組み合わせられる、
新たな選択肢の一つとしてRuriを提示する

実装

- **純粋なRustで実装され、Rustエコシステムに乗っかれる**
  - ホスト言語で実装しているのでRust⇔mruby連携が楽
- **WASMターゲット等での利用も想定し、そこそこ省依存**
  (no_std+allocなので、アロケータを準備できる環境では動く)
- **Rustの抽象化機構を素直に使う比較的シンプルな実装**
  - 極力Unsafeを削減し、Rust的な抽象化を利用

利用例

- 柔軟性を活かし、簡潔な記法で設定データを宣言的に記述
- Rustソフトウェアのプラグインシステム
- 強力なクレートの機能をそのまま使ってスクリプティング
- ちょっとしたロジックをmrubyで記述して実行中にホットリロード
- (その他、普通のRustが動く環境だとだいたい動かせる)

etc..

さらに、これらすべてを一貫した形で
(mrubyで)提供できる

SecHackでの１年間

6月: 当初は独自のRustむけ非同期ランタイムを作ろうとしていた（実はRustの非同期処理にはランタイムと呼ばれるクレートが別途必要）

9月: 紆余曲折の末方針を転換、mrubyのVMを書き始める。 本家mrubyのソースコードをプリントして読んだりしていた。9月末のイベントではフィボナッチ数列辺りまで動いたのでVMで発表

10~12月: このあたりからだんだんちゃんとしたVMっぽくなりはじめる。ドーナツが回るプログラムを移植したりしていた

1月: 第5回イベントではメタデータからテンプレート処理までのすべてに渡って、Rustの資産を活用しつつmrubyで書けるSSGをデモとして発表

やったこと＆これからやりたいこと

1月末時点で基本的な機能（メソッド呼び出しなど）は実装をすませたので、今後はメタプログラミングに直結する機能の実装と効率の改善や、Rustによる標準ライブラリ相当の機能の提供を行いたい。また、高速化も可能な範囲で行いたいが、実装をシンプル・Rustらしく保つという観点から、Rustの抽象化との折り合いが悪い機能(例えばNaN Boxingなど)の実装は現状考えていない。

SecHackに関わったすべての方々にこの場で謝意を表させていただきます
スタッフ/トレーナー/トレーニー/外部講師/修了生の皆様、1年間本当にお世話になりました！