Red Hat Developer Tools 1

Rust 1.52.1 Toolset の使用

Rust 1.52.1 Toolset のインストールおよび使用
Red Hat Developer Tools 1 Rust 1.52.1 Toolset の使用

Rust 1.52.1 Toolset のインストールおよび使用

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.
概要

Rust Toolset は、Red Hat Enterprise Linux プラットフォームで開発者向けの Red Hat 製品です。Rust Toolsetguide では、この製品の概要、異なるバージョンの Rust ツールの呼び出しおよび使用方法、および詳細な情報を含むリソースへのリンクを説明します。
目次

多様性を受け入れるオープンソースの強化 .................................................. 3

第1章 RUST TOOLSET .................................................. 4
  1.1. RUST TOOLSET コンポーネント .......................................... 4
  1.2. 互換性 .............................................................................. 4
  1.3. RED HAT ENTERPRISE LINUX 7 で RUST TOOLSET へのアクセス .... 4
  1.4. RUST TOOLSET のインストール ............................................ 6
  1.5. ドキュメントのインストール .............................................. 6
    1.5.1. Rust ドキュメントのインストール .................................. 6
    1.5.2. Cargo ドキュメントのインストール .................................. 7
  1.6. 関連情報 ........................................................................... 7

第2章 CARGO ビルドツール .................................................. 8
  2.1. CARGO ディレクトリ構造およびファイル配置 ....................... 8
  2.2. RUST プロジェクトの作成 ............................................... 8
  2.3. RUST プロジェクトのライブラリーの作成 ............................... 9
  2.4. RUST プロジェクトのビルド ............................................. 9
  2.5. リリースモードでの RUST プロジェクトのビルド .................. 10
  2.6. RUST プログラムの実行 .................................................. 11
  2.7. RUST プロジェクトのテスト ............................................. 12
  2.8. リリースモードでの RUST プロジェクトのテスト .................. 12
  2.9. RUST プロジェクト依存関係の設定 .................................. 13
  2.10. RUST プロジェクトのドキュメントの構築 ............................. 14
  2.11. VENDORING RUST プロジェクトの依存関係 ..................... 15
  2.12. 関連情報 ........................................................................... 15

第3章 RUSTFMT フォーマットツール ........................................... 17
  3.1. RUSTFMT のインストール .................................................. 17
  3.2. RUSTFMT をスタンドアロンツールとして使用 ....................... 17
  3.3. CARGO での RUSTFMT の使用 .......................................... 18
  3.4. 関連情報 ........................................................................... 19

第4章 RUST TOOLSET を使用したコンテナーイメージ ....................... 20
  4.1. RUST TOOLSET のコンテナーイメージの作成 ....................... 20
  4.2. 関連情報 ........................................................................... 21

第5章 RUST 1.52.1 TOOLSET の変更点 ...................................... 22
Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、弊社の CTO、Chris Wright のメッセージを参照してください。
第1章 RUST TOOLSET

Rust Toolset は、Red Hat Enterprise Linux 上の開発者向けの Red Hat 製品です。これは、Rust プログラミング言語 rustc コンパイラ、Rust パッケージマネージャ Cargo、rustfmt フォーマットツール、および必要なライブラリーの rustc コンパイラを提供します。

Rust Toolset は、Red Hat Developer Tools for Red Hat Enterprise Linux 7 の一部として配布されています。Red Hat Enterprise Linux 8 では、Rust Toolset がモジュールとして利用できます。

1.1. RUST TOOLSET コンポーネント

以下のコンポーネントは、Rust Toolset の一部として利用できます。

<table>
<thead>
<tr>
<th>名前</th>
<th>バージョン</th>
<th>説明</th>
</tr>
</thead>
<tbody>
<tr>
<td>rust</td>
<td>1.52.1</td>
<td>LLVM 用の Rust コンパイラーフrontエンド。</td>
</tr>
<tr>
<td>cargo</td>
<td>1.52.1</td>
<td>Rust のビルドシステムおよび依存関係マネージャー。</td>
</tr>
<tr>
<td>rustfmt</td>
<td>1.52.1</td>
<td>Rust コードの自動フォーマットを行うためのツール。</td>
</tr>
</tbody>
</table>

1.2. 互換性

Rust Toolset は、Red Hat Enterprise Linux 7 および Red Hat Enterprise Linux 8 では、以下のアーキテクチャーで利用できます。

- AMD アーキテクチャーおよび Intel 64 ビットアーキテクチャー
- 64 ビット ARM アーキテクチャー (RHEL 8 のみ)
- IBM Power Systems (リトルエンディアン)
- IBM Power Systems、Big Endian（RHEL 7 のみ）
- 64 ビット IBM Z

1.3. RED HAT ENTERPRISE LINUX 7 で RUST TOOLSET へのアクセス

Red Hat Enterprise Linux 7 に Rust Toolset をインストールするには、まず Red Hat Developer Tools リポジトリおよび Red Hat Software Collections リポジトリにアクセスして有効にする必要があります。これらのリポジトリがすでにシステムに割り当てられている場合は、「Rust Toolset のインストール」を参照してください。

前提条件

- wget がインストールされている。
  
  wget をインストールするには、以下のコマンドを実行します。
# yum install wget

手順

1. 以下を実行して最新のサブスクリプションデータをダウンロードします。
   
   ```bash
   # subscription-manager refresh
   ```

2. 以下を実行してシステムを登録します。
   
   ```bash
   # subscription-manager register
   ```

   また、『システム登録および登録解除』に従って、システムを登録することもできます。

3. 利用可能なサブスクリプションの一覧を表示し、以下のコマンドを実行してプール ID を特定します。
   
   ```bash
   # subscription-manager list --available
   ```

   Pool ID で始まる行でプール ID を検索します。

4. **Red Hat Developer Tools** リポジトリへのアクセスを提供するサブスクリプションをシステムに割り当てるには、以下のコマンドを実行します。
   
   ```bash
   # subscription-manager attach --pool=_appropriate pool ID from the subscription_
   ```

   • サブスクリプションの適切なプール ID を、直前の手順で特定したプール ID に置き換えます。

5. 以下を実行して、どのサブスクリプションがシステムに割り当てられていることを確認します。
   
   ```bash
   # sudo subscription-manager list --consumed
   ```

6. 以下を実行して **rhel-7-バリアント-devtools-rpms** リポジトリを有効にします。
   
   ```bash
   # subscription-manager repos --enable rhel-7-variant-devtools-rpms
   ```

   • variant を、Red Hat Enterprise Linux システムのバリアント（サーバーまたはワークステーション）に置き換えます。
   
   サーバーを使用して、幅広い開発ツールにアクセスすることを検討してください。

7. 以下を実行して **rhel-バリアント-rhscl-7-rpms** リポジトリを有効にします。
   
   ```bash
   # subscription-manager repos --enable rhel-variant-rhscl-7-rpms
   ```

   • variant を、Red Hat Enterprise Linux システムのバリアント（サーバーまたはワークステーション）に置き換えます。

8. 以下を実行して、Red Hat Developer Tools GPG キーをシステムに追加します。
# cd /etc/pki/rpm-gpg
# wget -O RPM-GPG-KEY-redhat-devel https://www.redhat.com/security/data/a5787476.txt
# rpm --import RPM-GPG-KEY-redhat-devel

関連情報

- システムを登録し、サブスクリプションに関連付ける方法は、Red Hat Subscription Management のガイドを参照してください。

1.4. RUST TOOLSET のインストール

すべての開発およびデバッグツールや依存パッケージを含む Rust Toolset をインストールするには、以下の手順を実行します。Rust Toolset には、LLVM Toolset の依存関係があります。

前提条件

- Red Hat Enterprise Linux 7 では、Red Hat Developer Tools コンテンツセットへのアクセスを提供するサブスクリプションがシステムに割り当てられている。
  サブスクリプションを割り当てるには、『Red Hat Enterprise Linux 7 で Rust Toolset へのアクセス』を参照してください。

- 利用可能なすべての Red Hat Enterprise Linux 更新がインストールされている。

手順

- Red Hat Enterprise Linux 7 で、以下を実行して rust-toolset-1.52 コレクションをインストールします。
  ```
  # yum install rust-toolset-1.52
  ```

- Red Hat Enterprise Linux 8 で、以下を実行して rust-toolset モジュールをインストールします。
  ```
  # yum module install rust-toolset
  ```

1.5. ドキュメントのインストール

Rust プログラミング言語および Cargo のドキュメントは、インストール可能なドキュメントとして利用できます。ローカルマシンにドキュメントをインストールし、アクセスするには、以下の手順を実行します。

前提条件

- Rust がインストールされている。
  詳細は、「Rust Toolset のインストール」を参照してください。

1.5.1. Rust ドキュメントのインストール

手順

- rust-doc パッケージをインストールするには、以下のコマンドを実行します。
Red Hat Enterprise Linux 7 の場合:

```bash
# yum install rust-toolset-1.52-rust-doc
```

/opt/rh/rust-toolset-1.52/root/usr/share/doc/rust/html/index.html 以下のパスの下に、Rust Programming Language ブックを検索します。

/opt/rh/rust-toolset-1.52/root/usr/share/doc/rust/html/std/index.html のすべての Rust コードパッケージの API ドキュメントを検索します。

Red Hat Enterprise Linux 8 の場合:

```bash
# yum install rust-doc
```

/usr/share/doc/rust/html/index.html の以下のパスの下にある「The Rust Programming Language」を参照してください。
すべての Rust コードパッケージの API ドキュメントは、/usr/share/doc/rust/html/std/index.html のパスの下に検索します。

1.5.2. Cargo ドキュメントのインストール

- cargo-doc パッケージをインストールするには、以下を実行します。
  - Red Hat Enterprise Linux 7 の場合:

```bash
# yum install rust-toolset-1.52-cargo-doc
```

Cargo, Rust’s Package Manager ガイドは、/opt/rh/rust-toolset-1.52/root/usr/share/doc/cargo/html/index.html のパスの下に検索します。

Red Hat Enterprise Linux 8 の場合:

```bash
# yum install cargo-doc
```

/usr/share/doc/cargo/html/index.html の以下のパスの下にある Cargo, Rust’s Package Manager ブックを検索します。

1.6. 関連情報

- Rust プログラミング言語の詳細は、公式の Rust ドキュメントを参照してください。
第2章 CARGO ビルドツール

Cargo は、Rust コンパイラー rustc のビルドツールおよびフロントエンドであり、パッケージおよび依存関係マネージャーです。これにより、Rust プロジェクトは特定のバージョン要件で依存関係を宣言し、完全な依存関係のグラフを解決し、パッケージのダウンロード、およびビルド、プロジェクト全体のテストを行うことができます。

Rust Toolset には、Cargo 1.52.1 が同梱されています。

注記
すべてのコマンドの前に scl enable を使用しないようにするには、以下のコマンドを実行します。

```bash
$ scl enable rust-toolset-1.52 'bash'
```

2.1. CARGO ディレクトリー構造およびファイル配置

Cargo ビルドツールは、Cargo パッケージ内のディレクトリー構造とファイル配置を定義するために、セット規則を使用します。cargo new コマンドを実行すると、マニフェストとプロジェクトファイル両方のパッケージディレクトリー構造とテンプレートが生成されます。デフォルトでは、パッケージの root ディレクトリーで新しい Git リポジトリも初期化します。

バイナリープログラムの場合、Cargo.toml という名前のテキストファイルを含むディレクトリー project_name と、main.rs という名前のテキストファイルを含むサブディレクトリー src を作成します。

関連情報
- Cargo ディレクトリー構造の詳細は、Cargo Book - パッケージレイアウト を参照してください。
- Rust コード組織の詳細は、「The Rust Programming Language - Managing Growing Projects with Packages, Crates, and Modules 」を参照してください。

2.2. RUST プロジェクトの作成

cargo new コマンドで Cargo 規則に従って設定される新しい Rust プロジェクトを作成します。プロジェクトコードを編集するには、メインの実行可能ファイル main.rs を編集して、src サブディレクトリーに新しいソースファイルを追加します。Cargo 規則の詳細は、「Cargo directory structure and file placements 」を参照してください。

手順
- 以下のコマンドを実行して Rust プロジェクトを作成します。
  - Red Hat Enterprise Linux 7 の場合:
    ```bash
    $ scl enable rust-toolset-1.52 'cargo new --bin project_name'
    ```
  - project_name をプロジェクト名に置き換えます。
  - Red Hat Enterprise Linux 8 の場合:
$ cargo new --bin project_name

- project_name をプロジェクト名に置き換えます。

関連情報

- プロジェクトの設定および依存関係の追加に関する詳細は、「Rust プロジェクトの依存関係の設定」を参照してください。

2.3. RUST プロジェクトのライブラリーの作成

Cargo ビルドツールを使用して Rust プロジェクトのライブラリーを作成するには、以下の手順を実行します。

前提条件

- 既存の Rust プロジェクト。
  Rust プロジェクトの作成方法に関する詳細は、「Rust プロジェクトの作成」を参照してくださ

手順

- Rust プロジェクトのライブラリーを作成するには、以下のコマンドを実行します。
  Red Hat Enterprise Linux 7 の場合:
  $ scl enable rust-toolset-1.52 'cargo new --lib project_name'
  - project_name を Rust プロジェクトの名前に置き換えます。
  Red Hat Enterprise Linux 8 の場合:
  $ cargo new --lib project_name
  - project_name を Rust プロジェクトの名前に置き換えます。

2.4. RUST プロジェクトのビルド

Cargo ビルドツールを使用して Rust プロジェクトをビルドします。Cargo はプロジェクトのすべての依存関係を解決し、不足している依存関係をダウンロードし、rustc コンパイラを使用してコンパイルします。

デフォルトで、プロジェクトはデバッグモードでビルドおよびコンパイルされます。リリースモードでプロジェクトをコンパイルする方法は、「Building a Rust project in release mode」を参照してください。

前提条件

- 既存の Rust プロジェクト。
  Rust プロジェクトの作成方法に関する詳細は、「Rust プロジェクトの作成」を参照してくださ

手順
Cargo が管理する Rust プロジェクトを構築するには、プロジェクトディレクトリーで実行します。

- Red Hat Enterprise Linux 7 の場合:
  
  $ scl enable rust-toolset-1.52 'cargo build'

- Red Hat Enterprise Linux 8 の場合:
  
  $ cargo build

Rust プロジェクトがビルドされ、コンパイルされている。

検証

- Cargo が管理する Rust プログラムを構築するには、プロジェクトディレクトリーで実行します。

- Red Hat Enterprise Linux 7 の場合:
  
  $ scl enable rust-toolset-1.52 'cargo check'

- Red Hat Enterprise Linux 8 の場合:
  
  $ cargo check

ヒント

cargo build コマンドの代わりに cargo check コマンドを使用して、実行ファイルを構築する必要がないときに Rust プログラムを構築することができることを検証することを検討してください。cargo check コマンドは、cargo build コマンドを使用する完全なプロジェクトビルドよりも高速です。

2.5. リリースモードでの RUST プロジェクトのビルド

Cargo ビルドツールを使用して、リリースモードで Rust プロジェクトをビルドします。リリースモードはソースコードの最適化で、コンパイルしたバイナリーの実行時間を短縮する一方で、コンパイル時間を長くすることができます。このモードを使用して、リリースと実稼働環境に適した最適化されたアーティファクトを生成します。

Cargo はプロジェクトのすべての依存関係を解決し、不足している依存関係をダウンロードし、rustc コンパイラを使用してコンパイルします。

デバッグモードでプロジェクトをコンパイルする方法は、「Rust プロジェクトの構築」を参照してください。

前提条件

- 既存の Rust プロジェクト。
  
  Rust プロジェクトの作成方法に関する詳細は、「Rust プロジェクトの作成」を参照してください。

手順

- リリースモードでプロジェクトをビルドするには、以下を実行します。
Red Hat Enterprise Linux 7の場合:
$ scl enable rust-toolset-1.52 'cargo build --release'

Red Hat Enterprise Linux 8の場合:
$ cargo build --release

検証

- Cargo が管理する Rust プログラムを構築するには、プロジェクトディレクトリーで実行します。
  - Red Hat Enterprise Linux 7の場合:
    $ scl enable rust-toolset-1.52 'cargo check'
  - Red Hat Enterprise Linux 8の場合:
    $ cargo check

ヒント

cargo build コマンドの代わりに cargo check コマンドを使用して、実行ファイルを構築する必要がないときに Rust プログラムを構築することができることを検証することを検討してください。cargo check コマンドは、cargo build コマンドを使用する完全なプロジェクトビルドよりも高速です。

2.6. RUST プログラムの実行

Cargo ビルドツールを使用して Rust プロジェクトを実行します。Cargo はまずプロジェクトを再構築し、作成された実行ファイルを実行します。開発中の場合、cargo run コマンドはビルドモードから独立して出力パスを正しく解決します。

前提条件

- ビルド済みの Rust プロジェクト。
  Rust プロジェクトのビルド方法は、「Rust プロジェクトのビルド」を参照してください。

手順

- Cargo によってプロジェクトとして管理される Rust プログラムを実行するには、プロジェクトディレクトリーで実行します。
  - Red Hat Enterprise Linux 7の場合:
    $ scl enable rust-toolset-1.52 'cargo run'
  - Red Hat Enterprise Linux 8の場合:
    $ cargo run
注記
プログラムがまだビルドされていない場合には、Cargo は実行前にプログラムを構築します。

2.7. RUST プロジェクトのテスト

Cargo ビルドツールを使用して Rust プログラムをテストします。Cargo はまずプロジェクトを再構築し、次にプロジェクトにあるテストを実行します。空き関数、モノミーフィックである関数のみをテストでき、引数を別のことに注意してください。関数の戻り値の型は () または Result< (), E> で、E: Error にする必要があります。

デフォルトでは、Rust プロジェクトはデバッグモードでテストされています。リリースモードでプロジェクトをテストする方法は、「リリースモードでの Rust プロジェクトのテスト」を参照してください。

前提条件

- ビルド済みの Rust プロジェクト。
  Rust プロジェクトのビルド方法は、「Rust プロジェクトのビルド」を参照してください。
- テストとしてマークされた関数。
  ソースコードで関数をテストとしてマークするには、関数の前に test 属性 #[test] を追加します。

手順

- Cargo によって管理される Rust プロジェクトでテストを実行するには、プロジェクトディレクトリリーで実行します。
  - Red Hat Enterprise Linux 7 の場合:
    ```
    $ scl enable rust-toolset-1.52 'cargo test'
    ```
  - Red Hat Enterprise Linux 8 の場合:
    ```
    $ cargo test
    ```

関連情報

- Rust プロジェクトでテストの実行に関する詳細は、「The Rust Reference - Testing attributes」を参照してください。

2.8. リリースモードでの RUST プロジェクトのテスト

Cargo ビルドツールを使用して、リリースモードで Rust プログラムをテストします。リリースモードはソースコードの最適化で、コンパイルしたバイナリーの実行時間を短縮する一方で、コンパイル時間を長くすることができます。このモードを使用して、リリースと実稼働環境に適した最適化されたアーティファクトを生成します。
Crate はまずプロジェクトを再構築し、次にプロジェクトにあるテストを実行します。空き関数、モノミーフィックである関数のみをテストでき、引数を別のことに注意してください。関数の戻り値の型は () または Result< (), E> で、E: Error にする必要があります。
デバッグモードでプロジェクトをテストする方法は、「Rust プロジェクトのテスト」を参照してください。

前提条件

- ビルド済みの Rust プロジェクト。
  Rust プロジェクトのビルド方法は、「Rust プロジェクトのビルド」を参照してください。

- テストとしてマークされた関数。
  ソースコードで関数をテストとしてマークするには、関数の前に test 属性 `#[test]` を追加します。

手順

- release モードで Cargo によって管理される Rust プロジェクトでテストを実行するには、プロジェクトディレクトリで実行します。
  - Red Hat Enterprise Linux 7 の場合:
    `$ scl enable rust-toolset-1.52 'cargo test --release'
  - Red Hat Enterprise Linux 8 の場合:
    `$ cargo test --release`

関連情報

- Rust プロジェクトでテストの実行に関する詳細は、「The Rust Reference - Testing attributes」を参照してください。

2.9. RUST プロジェクト依存関係の設定

Cargo ビルドツールを使用して Rust プロジェクトの依存関係を設定します。Cargo が管理するプロジェクトの依存関係を指定するには、プロジェクトディレクトリの Cargo.toml ファイルを編集して、プロジェクトを再構築します。Cargo は Rust コードパッケージとその依存関係をダウンロードし、ローカルで保存し、依存関係コードパッケージを含むすべてのプロジェクトのソースコードをビルドし、作成された実行ファイルを実行します。

前提条件

- ビルド済みの Rust プロジェクト。
  Rust プロジェクトのビルド方法は、「Rust プロジェクトのビルド」を参照してください。

手順

1. プロジェクトディレクトリで Cargo.toml ファイルを開きます。

2. 記述した [dependencies] セクションに移動します。
  各依存関係は、次の形式で新しい行に一覧表示されます。

```
crate_name = version
```

Rust コードパッケージは crates と呼ばれています。
3. 依存関係を編集します。

4. 以下を実行してプロジェクトを再構築します。
   - Red Hat Enterprise Linux 7 の場合:
     ```bash
     $ scl enable rust-toolset-1.52 'cargo build'
     ```
   - Red Hat Enterprise Linux 8 の場合:
     ```bash
     $ cargo build
     ```

5. 以下のコマンドを使用してプロジェクトを実行します。
   - Red Hat Enterprise Linux 7 の場合:
     ```bash
     $ scl enable rust-toolset-1.52 'cargo run'
     ```
   - Red Hat Enterprise Linux 8 の場合:
     ```bash
     $ cargo run
     ```

関連情報

- Rust 依存関係の設定に関する詳細は、「Jargo Book - 依存関係の指定」を参照してください。

2.10. RUST プロジェクトのドキュメントの構築

Cargo ツールを使用して、抽出用にマークされたソースコード内のコメントからドキュメントを生成します。ドキュメントのコメントは、公開関数、変数、メンバーに対してのみ抽出されることに注意してください。

前提条件

- ビルド済みの Rust プロジェクト。
  Rust プロジェクトのビルド方法は、「Rust プロジェクトのビルド」を参照してください。

- 設定された依存関係。
  依存関係の設定に関する詳細は、「Rust プロジェクト依存関係の設定」を参照してください。

- ドキュメントへの抽出についてマークされたソースコードのコメント。
  抽出のためにコメントをマークするには、3 つのスラッシュ（//）を使用してドキュメントのコメントをドキュメント化の頭に置きます。Cargo は、コメントの Markdown 言語をサポートします。

手順

- Cargo を使用してプロジェクトのドキュメンテーションを構築するには、プロジェクトディレクトリーで実行します。
  - Red Hat Enterprise Linux 7 の場合:
    ```bash
    $ scl enable rust-toolset-1.52 'cargo doc --no-deps'
    ```
Red Hat Enterprise Linux 8 の場合:

```bash
$ cargo doc --no-deps
```

注記
生成されたドキュメントに依存関係を含めるには、--no-deps オプションを省略します。
作成後にブラウザーで生成されたドキュメントを開くには、--open オプションを追加します。

関連情報
- Cargo を使用したドキュメントの構築の詳細は、「The Rust Programming Language - Making Useful Documentation Comments」を参照してください。

2.11. VENDORING RUST プロジェクトの依存関係
オフライン再分配用に Rust プロジェクトの依存関係のローカルコピーを作成し、Cargo ビルドツールを使用して再利用します。この手順は、ベンダー・プロジェクトの依存関係と呼ばれています。Windows オペレーティングシステムでプロジェクトをビルドするための Rust コードパッケージを含むベンダー依存関係は vendor ディレクトリーに置かれます。ベンダーの依存関係は、インターネットに接続せずに Cargo で使用することができます。

前提条件
- ビルド済みの Rust プロジェクト。
  Rust プロジェクトのビルド方法は、「Rust プロジェクトのビルド」を参照してください。
- 設定された依存関係。
  依存関係の設定に関する詳細は、「Rust プロジェクト依存関係の設定」を参照してください。

手順
- Cargo を使用して依存関係で Rust プロジェクトをベンダーにするには、プロジェクトディレクトリーで実行します。
  - Red Hat Enterprise Linux 7 の場合:
    ```bash
    $ scl enable rust-toolset-1.52 'cargo vendor'
    ```
  - Red Hat Enterprise Linux 8 の場合:
    ```bash
    $ cargo vendor
    ```

2.12. 関連情報
- Cargo の詳細は、『Official Cargo Guide』を参照してください。
- Rust Toolset に含まれる man ページを表示するには、以下のコマンドを実行します。
  - Red Hat Enterprise Linux 7 の場合:
$ scl enable rust-toolset-1.52 'man cargo'

- Red Hat Enterprise Linux 8 の場合:

  $ man cargo
第3章 RUSTFMT フォーマットツール

`rustfmt` フォーマットツールを使用して、Rust プログラムのソースコードを自動的にフォーマットできます。`rustfmt` をスタンドアロンツールまたは Cargo で使用できます。

注記
すべてのコマンドの前に `scl enable` を使用しないようにするには、以下のコマンドを実行します。

```
$ scl enable rust-toolset-1.52 'bash'
```

3.1. RUSTFMT のインストール

`rustfmt` フォーマットツールをインストールするには、以下の手順を実行します。

前提条件

- Rust Toolset がインストールされている。 詳細は、「Rust Toolset のインストール」を参照してください。

手順

- 以下のコマンドを実行して `rustfmt` をインストールします。
  - Red Hat Enterprise Linux 7 の場合:
    ```
    # yum install rust-toolset-1.52-rustfmt
    ```
  - Red Hat Enterprise Linux 8 の場合:
    ```
    # yum install rustfmt
    ```

3.2. RUSTFMT をスタンドアロンツールとして使用

`rustfmt` をスタンドアロンツールとして使用して、Rust ソースファイルとそのすべての依存関係をフォーマットします。別の方法として、Cargo ビルドツールで `rustfmt` を使用します。詳細は、「Cargo での rustfmt の使用」を参照してください。

前提条件

- 既存の Rust プロジェクト。 Rust プロジェクトの作成方法に関する詳細は、「Rust プロジェクトの作成」を参照してください。

手順

- `rustfmt` をスタンドアロンツールとして使用して Rust ソースファイルをフォーマットするには、以下のコマンドを実行します。
  - Red Hat Enterprise Linux 7 の場合:
$ scl enable rust-toolset-1.52 'rustfmt source-file'

*source-file* は、ソースファイルの名前に置き換えます。
または、*source-file* を標準入力に置き換えることもできます。次に、*rustfmt* は出力を標準出力に提供します。

- Red Hat Enterprise Linux 8 の場合:

  $ rustfmt source-file

*source-file* は、ソースファイルの名前に置き換えます。
または、*source-file* を標準入力に置き換えることもできます。次に、*rustfmt* は出力を標準出力に提供します。

注記

デフォルトでは、*rustfmt* は詳細を表示したりバックアップを作成したりせずに、影響を受けるファイルを変更します。詳細を表示し、バックアップを作成するには、--write-mode の値を指定して *rustfmt* を実行します。

- Red Hat Enterprise Linux 7 の場合:

  $ scl enable rust-toolset-1.52 'rustfmt source-file' --write-mode value

- Red Hat Enterprise Linux 8 の場合:

  $ rustfmt source-file --write-mode value

### 3.3. CARGO での RUSTFMT の使用

Cargo で *rustfmt* ツールを使用して、Rust ソースファイルとそのすべての依存関係をフォーマットします。
別の方法として、*rustfmt* をスタンドアロンツールとして使用します。詳細は、「*rustfmt* をスタンドアロンツールとして使用」を参照してください。

前提条件

- 既存の Rust プロジェクト。
Rust プロジェクトの作成方法に関する詳細は、「Rust プロジェクトの作成」を参照してください。

手順

- Cargo コードパッケージ内のソースファイルをすべてフォーマットするには、以下のコマンドを実行します。

  - Red Hat Enterprise Linux 7 の場合:

    $ scl enable rust-toolset-1.52 'cargo fmt'

  - Red Hat Enterprise Linux 8 の場合:

    $ cargo fmt
注記

`rustfmt`フォーマットのオプションを変更するには、プロジェクトディレクトリーに設定ファイル`rustfmt.toml`を作成し、設定をファイルに追加します。

3.4. 関連情報

- `rustfmt`のヘルプページを表示するには、以下を実行します。
  
  - Red Hat Enterprise Linux 7 の場合:
    
    ```bash
    $ scl enable rust-toolset-1.52 'rustfmt --help'
    ```
  
  - Red Hat Enterprise Linux 8 の場合:
    
    ```bash
    $ rustfmt --help
    ```

- `rustfmt`ツールを設定するには、ファイル`Configurations.md`を編集します。
  
  - Red Hat Enterprise Linux 7 では、以下のパスで検索します。
    
    `/opt/rh/rust-toolset-1.52/root/usr/share/doc/rust-toolset-1.52-rustfmt-1.52.1/Configurations.md`
  
  - Red Hat Enterprise Linux 8 の場合は、以下のパスにあります。
    
    `/usr/share/doc/rustfmt/Configurations.md`
第4章 RUST TOOLSET を使用したコンテナーイメージ

Dockerfile を使用して、Red Hat Universal Base Images(UBI)コンテナ上に独自の Rust Toolset コンテナーイメージをビルドできます。

4.1. RUST TOOLSET のコンテナーイメージの作成

Rust Toolset パッケージは、Red Hat Universal Base Images(UBI)リポジトリに含まれます。Rust のコンテナーイメージを作成するには、UBI を設定し、Dockerfile を編集して Rust をインストールします。コンテナーのサイズが小さくなるには、Rust Toolset 全体ではなく、個々のパッケージのみをインストールします。

前提条件

- 既存の Dockerfile。
- Dockerfile の作成に関する詳細は、「Dockerfile reference」ページを参照してください。
- UBI のセットアップ。
  - a. UBI を設定するには、Red Hat Container Catalog にアクセスします。
    - Red Hat Enterprise Linux 7 では、UBI 7 を検索します。
    - Red Hat Enterprise Linux 8 で、UBI 8 を検索します。
  - b. UBI を選択します。
  - c. Get this image をクリックして、手順に従います。

手順

- Rust Toolset を含むコンテナーを作成するには、以下の行を Dockerfile に追加します。
  - Red Hat Enterprise Linux 7 の場合:
    ```
    FROM registry.access.redhat.com/ubi7/ubi:latest
    RUN yum install -y rust-toolset-1.52
    ```
  - Red Hat Enterprise Linux 8 の場合:
    ```
    FROM registry.access.redhat.com/ubi8/ubi:latest
    RUN yum install -y rust-toolset
    ```
- 個々のパッケージのみを含むコンテナーイメージを作成するには、以下の行を Dockerfile に追加します。
  - Red Hat Enterprise Linux 7 の場合:
    ```
    RUN yum install rust-toolset-1.52-package-name
    ```
  - package-name を、インストールするパッケージ名に置き換えます。
  - Red Hat Enterprise Linux 8 の場合:
RUN yum install package-name

- package-name を、インストールするパッケージ名に置き換えます。

4.2. 関連情報

- Red Hat UBI イメージの詳細は、「コンテナーイメージの使用」を参照してください。

- Red Hat UBI リポジトリの詳細は、「Universal Base Images(UBI): Images, repositories, packages, and code」を参照してください。
第5章 RUST 1.52.1 TOOLSET の変更点

Rust Toolset がバージョン 1.52.1 にリベースされました。

Rust Toolset のバージョンが 1.49.0 から 1.52.1 に更新されました。以下は、主な変更点です。

- constant-value パラメーターを使用して汎用を定義できるようになりました。この変更により、整数、ブール値、または文字型の値、およびアレイに汎用的に記述でき、要素型や長さ上で汎用関数を書き込むことができます。さらに、新規標準ライブラリーの配列型 API std::array::into_iter を使用して、値別にアレイからアイテムを繰り返すことができるようになりました。

- 今回のリリースで、増分コンパイルがデフォルトで無効になりました。Rust 1.52.0 に追加された新規検証により、インクリメンタルビルドで誤コンパイルをトリガーするすべての rust バージョンに存在するバグが検出されました。ダウングレードによりセキュリティーの修正が行われないため、増分コンパイルを無効にするか、Rust 1.52.1 にアップグレードしてください。

更新の詳細については、アップストリームの Rust 1.52.1 リリースノートを参照してください。