Pemrograman Rust

Bambang Purnomosidi D. P. <bambangpdp@gmail.com>

Version 0.0.1, 2020-06-02

Daftar Isi

Pengantar

Sebuah buku tentang Bahasa Pemrograman Rust.

Penghargaan

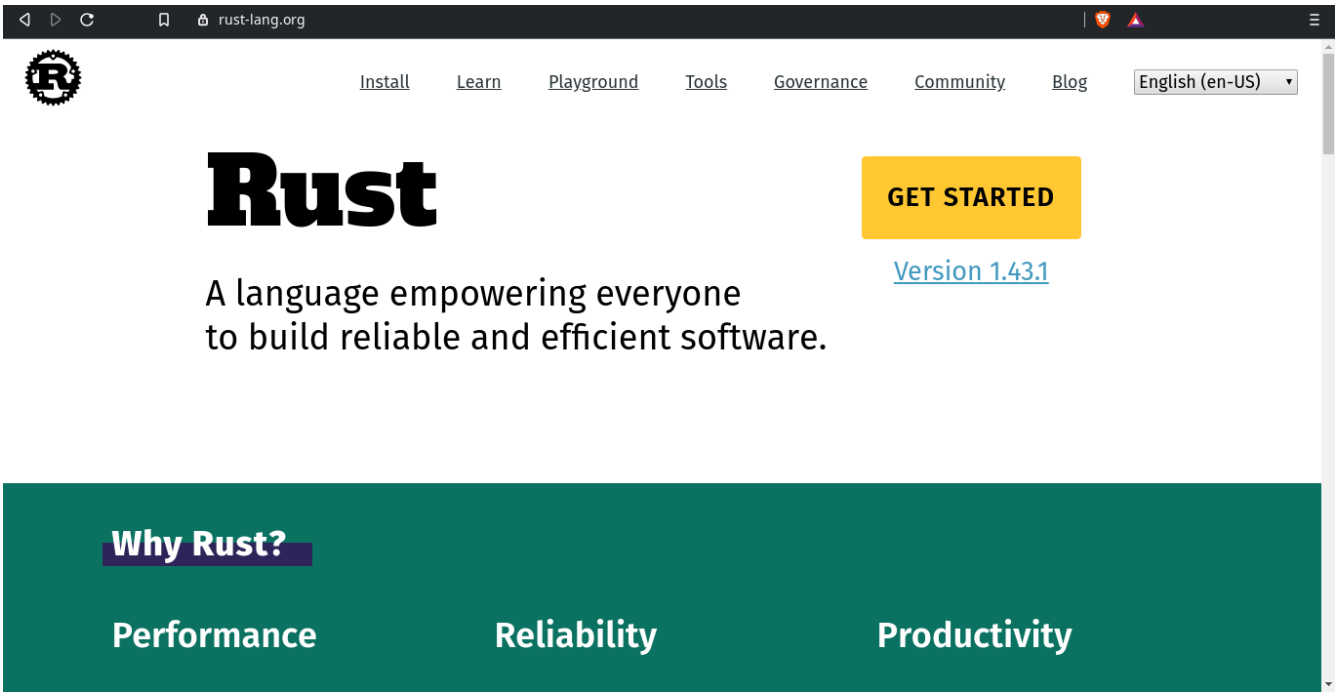
Kami mengucapkan terima kasih atas partisipasi dari rekan-rekan semua.

Bagian I: Pengenalan dan Ekosistem

Bagian ini menjelaskan tentang gambaran umum dari Rust serta persiapan untuk membangun aplikasi menggunakan Rust

Bab 1. Mengenal Rust

Bab ini membahas tentang gambaran umum dari bahasa pemrograman serta peranti pengembangan Rust. Dengan mempelajari bab ini, pembaca diharapkan bisa memahami gambaran umum dari Rust sehingga bisa memberikan semacam pemahaman terhadap ruang lingkup masalah-masalah pemrograman yang bisa diselesaikan menggunakan Rust serta posisi Rust di antara berbagai bahasa pemrograman dan peranti pengembangan lainnya.



1.1. Sejarah Singkat Rust

1. Rust pertama kali dibuat oleh salah seorang pegawai dari Mozilla yang bernama *Graydon Hoare* sekitar tahun 2006. Saat itu, Graydon membuat bahasa pemrograman baru dan *compiler* dari bahasa pemrograman tersebut menggunakan **OCaml**.
2. Mozilla mulai men-sponsori dan mendukung Rust untuk keperluan internal pada tahun 2009 (diumumkan tahun 2010).
3. Tahun 2010, pengembangan Rust menggunakan OCaml dihentikan, digantikan dengan *self-hosting compiler* (Rust dibuat dengan menggunakan Rust). Tahun 2011, Rust berhasil digunakan untuk mengkompilasi dirinya sendiri. Saat itu, Rust menggunakan *LLVM* sbagai *compiler backend*.
4. Versi stabil pertama (versi 1.0.0) dari Rust berhasil dirilis pada tanggal 15 Mei 2015.
5. Setelah itu, Rust menetapkan pola rilis pasti setiap 6 minggu, artinya setiap 6 minggu ada rilis baru untuk versi stabil, beta, dan *nightly*.

1.2. Paradigma Pemrograman Rust

1.3. Lisensi Rust

Semua artifak dari Rust (*compiler*, logo, situs Web, dan lain-lain) mempunyai lisensi ganda:

1. *MIT License*
2. *Apache License* - Versi 2.0

1.4. Software yang Dibangun Menggunakan Rust

Rust digunakan untuk membangun berbagai software, mulai dari *low level* sampai dengan *high level*. Istilah *low level* dan *high level* ini digunakan untuk menunjukkan kedekatan dengan akses mesin. *Low level* dikenal juga dengan istilah *system programming* (meski istilah ini bukan merupakan istilah yang kanonikal). Rust merupakan satu di antara sedikit bahasa pemrograman dan peranti pengembangan yang bisa digunakan utk semua level. Bagian ini menunjukkan beberapa software yang dibangun dengan menggunakan Rust.

1. **Servo** (https://servo.org/): *engine* penjelajah Web (*Web browser*) yang digunakan dalam browser **Mozilla Firefox** melalui proyek **Quantum**.
2. **Redox** (https://www.redox-os.org/): sistem operasi baru dengan teknologi *microkernel* dengan *userland* mirip Unix.
3. **Nushell** (https://www.nushell.sh/): shell.
4. **TiKV** (https://tikv.org/): basis data *key-value* transaksional yang terdistribusi.
5. **Deno** (https://deno.land/): *runtime* untuk JavaScript dan TypeScript.
6. **Discord** (https://discord.com/): salah satu peranti pengembangan yang digunakan untuk mengembangkan sistem Discord.

1.5. Kelebihan dan Kekurangan Rust

1.6. Domain Masalah dari Rust

Bab 2. Instalasi Rust

Untuk menggunakan Rust, tentu saja anda harus melakukan instalasi terhadap Rust dan ekosistem yang bisa digunakan untuk mendukung proses membangun software menggunakan Rust. Bab ini membahas tentang berbagai cara yang bisa digunakan untuk mulai menggunakan Rust. Selain itu, di bab ini juga akan dibahas tentang berbagai peranti pengembangan yang lazim digunakan sebagai hasil dari instalasi serta pengenalan penggunaannya.

2.1. Rilis Rust

Rust mempunyai 3 kategori rilis:

1. *Stable*: rilis stabil, dengan *test* yang dilakukan secara menyeluruh.
2. *Beta*: rilis versi ini merupakan rilis yang disiapkan untuk menjadi versi stabil berikutnya..
3. *Nightly*: rilis versi ini merupakan rilis yang berisi berbagai eksperimen yang mungkin bisa masuk ke versi stabil berikutnya (setelah melalui versi *Beta*), maupun tidak akan pernah dimasukkan ke rilis Rust.

Saat membangun aplikasi, pemrogram bebas untuk menggunakan kategori rilis manapun. Meskipun demikian, dianjurkan untuk menggunakan versi *Stable* karena fitur yang ada di dalamnya adalah fitur-fitur yang sudah stabil sehingga memudahkan pemrogram untuk me-*maintain* aplikasi yang dikembangkan.

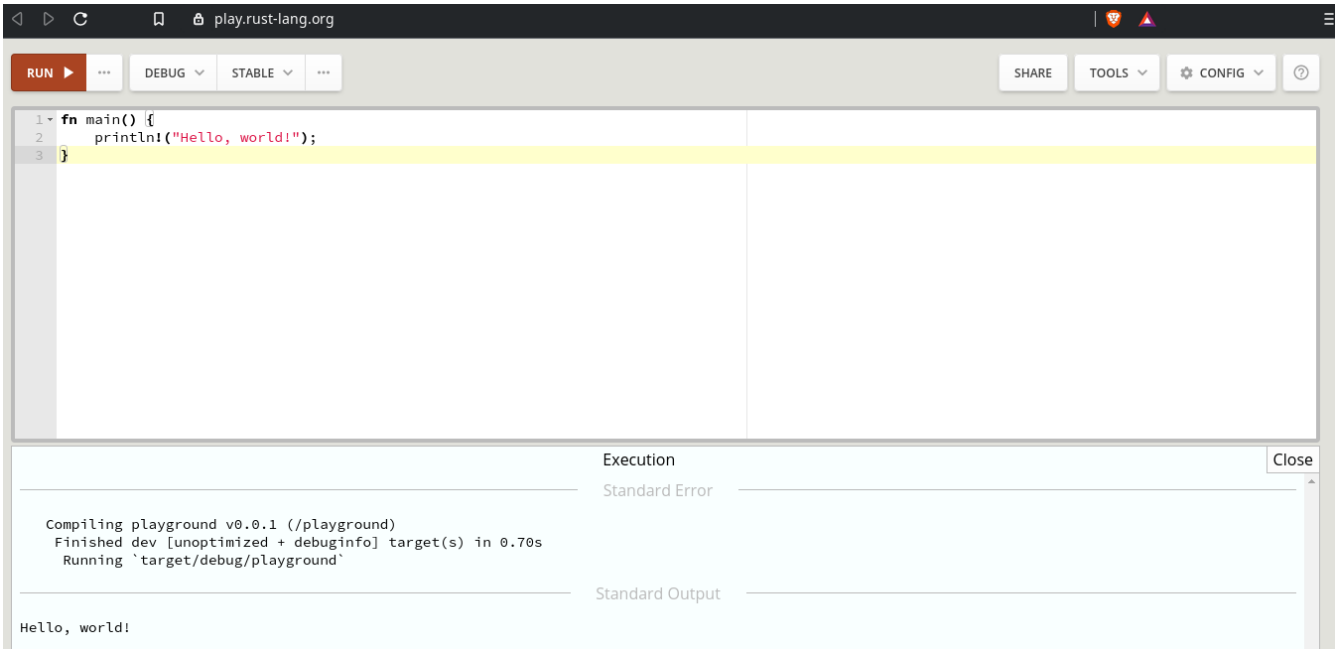
Untuk semua rilis tersebut, Rust menggunakan pedoman yang disebut dengan **Semantic Versioning**. Dengan menggunakan pedoman ini, setiap rilis Rust terdiri atas 3 bagian:

1. MAJOR: rilis dengan perubahan API (*Application Programming Intergace*) yang tidak kompatibel dengan versi MAJOR sebelumnya.
2. MINOR: rilis dengan penambahan fungsionalitas yang kompatibel dengan versi sebelumnya.
3. PATCH: rilis dengan perbaikan terhadap *bugs* yang kompatibel dengan versi sebelumnya.

Sebagai contoh, versi 1.44.0 dari versi Rust berisi Rust dengan semua API yang kompatibel dengan versi 1.x.x sebelumnya. Angka 44 berarti penambahan fungsionalitas yang bersifat kompatibel dengan versi penambahan fungsionalitas sebelumnya. Angka 0 berarti sama sekali belum ada perubahan perbaikan *bug* untuk versi 1.44 tersebut.

2.2. Rust Playground

Jika kebutuhan kita hanya untuk mencoba beberapa bagian kode sumber, maka kita cukup hanya menggunakan **Rust Playground** saja. Setelah mengakses URL tersebut, kita bisa menuliskan kode sumber dan menjalankan kode sumber tersebut tanpa perlu melakukan instalasi peranti pengembangan Rust.



2.3. Instalasi Rust

src/bab-02/hello-plain/hello.rs

```
fn main() {                    ①

    println!("Hello World!");   ②

}
```

① Bagian yang dieksekusi saat program hasil kompilasi dipanggil.

② Isi dari program.

rustup

```
$ curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
info: downloading installer

Welcome to Rust!

This will download and install the official compiler for the Rust
programming language, and its package manager, Cargo.

It will add the cargo, rustc, rustup and other commands to
Cargo's bin directory, located at:

  /home/zaky/.cargo/bin

This can be modified with the CARGO_HOME environment variable.

Rustup metadata and toolchains will be installed into the Rustup
home directory, located at:
```

```
  /home/zaky/.rustup

This can be modified with the RUSTUP_HOME environment variable.

This path will then be added to your PATH environment variable by
modifying the profile file located at:

  /home/zaky/.profile

You can uninstall at any time with rustup self uninstall and
these changes will be reverted.

Current installation options:


   default host triple: x86_64-unknown-linux-gnu
     default toolchain: stable
               profile: default
  modify PATH variable: yes

1) Proceed with installation (default)
2) Customize installation
3) Cancel installation
>1

info: profile set to 'default'
info: default host triple is x86_64-unknown-linux-gnu
info: syncing channel updates for 'stable-x86_64-unknown-linux-gnu'
info: latest update on 2020-06-04, rust version 1.44.0 (49cae5576 2020-06-01)
info: downloading component 'cargo'
  4.9 MiB /   4.9 MiB (100 %)   1.2 MiB/s in  4s ETA:  0s
info: downloading component 'clippy'
  1.9 MiB /   1.9 MiB (100 %)   1.1 MiB/s in  1s ETA:  0s
info: downloading component 'rust-docs'
 12.2 MiB /  12.2 MiB (100 %)   1.1 MiB/s in 11s ETA:  0s
info: downloading component 'rust-std'
 17.6 MiB /  17.6 MiB (100 %)   1.1 MiB/s in 15s ETA:  0s
info: downloading component 'rustc'
 60.2 MiB /  60.2 MiB (100 %)   1.1 MiB/s in 54s ETA:  0s
info: downloading component 'rustfmt'
  3.2 MiB /   3.2 MiB (100 %)   1.4 MiB/s in  2s ETA:  0s
info: installing component 'cargo'
info: installing component 'clippy'
info: installing component 'rust-docs'
 12.2 MiB /  12.2 MiB (100 %)   7.4 MiB/s in  1s ETA:  0s
info: installing component 'rust-std'
info: installing component 'rustc'
 60.2 MiB /  60.2 MiB (100 %)  13.5 MiB/s in  6s ETA:  0s
info: installing component 'rustfmt'
info: default toolchain set to 'stable'
```

```
  stable installed - rustc 1.44.0 (49cae5576 2020-06-01)


Rust is installed now. Great!

To get started you need Cargo's bin directory ($HOME/.cargo/bin) in your PATH
environment variable. Next time you log in this will be done
automatically.

To configure your current shell run source $HOME/.cargo/env
$
```

Setelah proses instalasi tersebut, ada file berisi variabel lingkungan (*environment variables*) yang harus diaktifkan, yaitu **$HOME/.cargp/env**. Berikut adalah kondisi sebelum diaktifkan dan setelah diaktifkan menggunakan perintah **source**:

cargo/env

```
$ rustc
-bash: rustc: perintah tidak ditemukan
$ cargo
-bash: cargo: perintah tidak ditemukan
$ source .cargo/env
$ rustc --version
rustc 1.44.0 (49cae5576 2020-06-01)
$ cargo --version
cargo 1.44.0 (05d080faa 2020-05-06)
$
```

Secara default, isi dari file **.cargo/env** sudah diletakkan pada file **.profile** sehingga akan aktif setiap login.

2.4. Memahami Ekosistem Rust

2.5. Pengenalan Cargo

2.5.1. Inisialisasi Proyek

2.5.2. Membangun Proyek

2.5.3. Membersihkan Hasil Kompilasi

2.5.4. Antara Debug dan Release

2.5.5. Menjalankan Hasil

Bab 3. IDE untuk Rust

IDE (*Integrated Development Environment*) adalah software yang digunakan sebagai peranti pengembangan terintegrasi. IDE sangat penting dalam membangun aplikasi. Produktivitas pemrogram biasanya sangat tergantung dari kepiawaiannya menggunakan IDE. Bab ini membahas tentang IDE yang bisa digunakan untk membangun aplikasi menggunakan Rust. Ada dua software dasar yang akan dijelaskan dan digunakan dalam bab ini, yaitu:

1. Vim
2. Visual Studio Code / VSCodium

3.1. Vim

3.2. Visual Studio Code / VSCodium

Bab 4. Ekosistem Rust

Mempelajari suatu bahasa pemrograman, khususnya dalam rangka untuk membangun aplikasi, tidak hanya selesai dengan mempelajari unsur bahasanya saja. Di luar itu, biasanya masih banyak unsur pendukung yang membentuk suatu ekosistem. Dengan memahami ekosistem Rust, seorang pemrogram yang menggunakan Rust akan relatif lebih mudah untuk menggunakan berbagai macam sumber daya untuk menyelesaikan masalah pemrograman yang dihadapi.

4.1. Paket (*Crates*)

4.2. *Crates* yang Bermanfaat Bagi Pemrogram Rust

4.2.1. Evcxr (https://github.com/google/evcxr/tree/master/evcxr_repl)

Crate ini digunakan sebagai REPL (*Read-Eval-Print-Loop*) untuk Rust. Untuk instalasi:

Evcxr

```
cargo install evcxr_repl
```

4.2.2. IRust (https://github.com/sigmaSd/IRust)

Crate ini juga digunakan sebagai REPL untuk Rust. Untuk instalasi:

IRust

```
cargo install irust
```

4.3. Berbagai Web untuk Melacak Status dalam Domain Tertentu

Komunitas membangun berbagai situs Web untuk melacak kesiapan Rust dalam domain tertentu. Situs Web tersebut biasanya dikenal dengan nama **Are we X yet?** dengan **X** adalah domain tertentu tersebut. Daftar dari **Are we X yet?** bisa dilihat di https://wiki.mozilla.org/Areweyet

4.4. Mengikuti Perkembangan Rust

Ada berbagai sumber daya di Internet yang bisa digunakan untuk mengikuti perkembangan dari Rust serta ekosistemnya.

1. **This Week in Rust**, bisa diakses di https://this-week-in-rust.org/
2. **Twitter**, berbagai *account* yang bisa diikuti:
    - https://twitter.com/rustlang: Twitter resmi dari Rust
    - *Hashtag* #rustlang di Twitter

Bagian II: Sintaksis dan Semantik dari Rust

Bagian ini menjelaskan tentang sintaksis dari bahasa pemrograman Rust. Beberapa bagian sudah membahas tentang pustaka standar sesuai pada materi pembahasan. Jika merupakan pustaka standar, bagian tersebut akan diberi catatan.

Bab 5. Variabel dan Komentar

Var dan komentar

Bab 6. Tipe Data

Tipe data

Bab 7. Fungsi

Fungsi

Bab 8. Pengandali Aliran Program

Pengendali aliran program

Bab 9. Struktur Data

Struktur data

Bab 10. Ownership

Ownership

Bab 11. Enum

Enum

Bab 12. Pembandingan Pola

Pembandingan pola

Bagian III: Pustaka Standar Rust

Bagian ini menjelaskan berbagai pustaka standar dari Rust. Pustaka standar merupakan API yang menjadi bagian dari Rust dan bisa diakses setelah kita melakukan instalasi Rust tanpa perlu melakukan instalasi tambahan lain. Bagian ini tidak menjelaskan secara rinci semua pustaka standar, tetapi hanya beberapa saja. Setelah itu, pembaca bisa melihat pada dokumentasi lengkap dari pustaka standar dari Rust untuk mengetahui lebih lanjut.

Bab 13. Mengelola String

Bab 14. Pemrograman Konkuren

Bab 15. Net

Bab 16. Path

Bab 17. Sistem File

Bab 18. Operasi I/O

Bab 19. Mengelola Proses

Bagian IV: Topik Khusus

Bagian ini menjelaskan berbagai topik pemrograman khusus yang bisa dilakukan dengan menggunakan Rust.

Bab 20. Akses Basis Data

Bab 21. Pemrograman Web

Bab 22. Antarmuka Teks

Bab 23. Antarmuka Grafis