Pemrograman Rust

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

Version 0.0.1-rev-2020-08-28 06:20:55 +0700

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 (wip)

1.1. Tentang Bab Ini (wip)

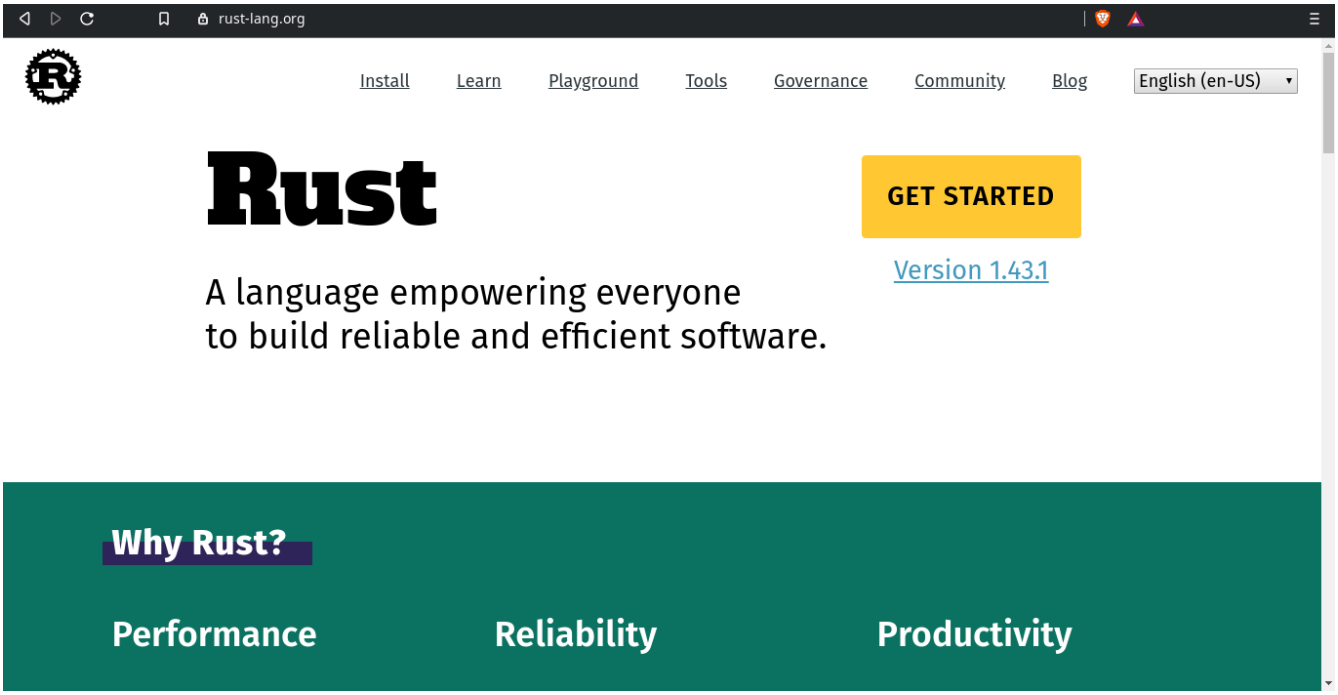
Bab ini membahas tentang gambaran umum dari bahasa pemrograman serta peranti pengembangan Rust. Dengan mempelajari bab ini, pembaca diharapkan bisa memahami gambaran umum dari Rust sehingga bisa memberikan semacam pemahaman terhadap ruang lingkup masalah-masalah pemrograman yang bisa diselesaikan menggunakan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

1.2. Apakah Rust Itu? (wip)

Secara umum, biasanya para pemrograman akan menyebut **Rust** untuk segala macam peranti pengembangan yang terkait dengan Rust. Pada dasarnya, saat membicarakan tentang peranti pengembangan, ada beberapa komponen sebagai berikut:

1. Spesifikasi bahasa pemrograman
2. Implementasi bahasa pemrograman dalam bentuk *compiler* / *interpreter*
3. *Package Manager* yang digunakan untuk mengelola pustaka / paket yang diperlukan saat dilakukan proses kompilasi maupun untuk instalasi berbagai peranti pendukung yang dibutuhkan pemrogram saat membangun aplikasi.
4. *Build Tool* yang digunakan untuk mengelola proses kompilasi serta hasil akhirnya.

Hal tersebut juga berlaku untuk **Rust**. Meskipun seringkali hanya disebut **Rust**, biasanya sudah mengacu ke setidaknya spesifikasi bahasa pemrograman Rust serta *compiler* dari Rust. Pada buku ini, penyebutan **Rust** akan mengacu pada spesifikasi bahasa pemrograman serta peranti standar untuk **compiler** maupun pustaka standar yang disertakan pada saat instalasi Rust. Informasi tentang Rust bisa diperoleh di Web Peranti Pengembangan Rust.



Gambar 1. Web Peranti Pengembangan Rust

1.3. Sejarah Singkat Rust (wip)

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*.

1.4. Paradigma Pemrograman Rust (wip)

Paradigma pemrograman merupakan cara pandang untuk menyelesaikan permasalahan pemrograman. Rust tidak mempunyai suatu paradigma pemrograman spesifik tertentu, tetapi lebih ke arah *multi-paradigm* atau mempunyai lebih dari satu paradigma. Secara umum, paradigma pemrograman Rust adalah:

1. Konkuren: mendukung pemrograman secara konkuren, artinya lebih dari satu unit komputasi bisa dieksekusi secara "bersamaan" (secara *overlap*, bergantian - tidak harus menunggu satu unit komputasi selesai baru kemudian satu unit komputasi berikutnya dieksekusi.
2. Fungsional: program dikonstruksi dengan menggunakan melakukan komposisi *function*.

3. Generik: tipe pada fungsi bisa dispesifikasikan belakangan, bukan pada saat pembuatan fungsi.
4. Imperatif: program terdiri atas berbagai *statement* yang mengubah *state* dari program (misal dengan memanipulasi variabel, dan lain-lain).
5. Terstruktur: program menekankan pada penggunakan berbagai struktur kendali serta subprogram, terutama digunakan untuk *readibility*.
6. PBO (Pemrograman Berorientasi Obyek): meski Rust tidak murni bahasa PBO, tetapi Rust mendukung berbagai fitur PBO yang memungkinkan pemrogram untuk mengkonstruksi program dengan pendekatan obyek yang mempunyai karakteristik serta perilaku tertentu dan adanya interaksi antar obyek tersebut dalam menyelesaikan masalah pemrograman.

1.5. Lisensi Rust (wip-nr)

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

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

1.6. Software yang Dibangun Menggunakan Rust (wip)

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.

1.7. Kelebihan dan Kekurangan Rust (wip)

1.8. Domain Masalah dari Rust (wip)

Rust bisa digunakan untuk menyelesaikan berbagai masalah pada berbagai domain. Secara umum, Rust bisa digunakan untuk pembuatan software di aras rendah (**system programming**) maupun di berbagai masalah pemrograman aras atas. Beberapa domain masalah yang bisa diselesaikan oleh

Rust antara lain:

1. **System Programming**
2. Akses ke peranti keras (**interfacing**)
3. CLI (**Command Line Interface**)
4. Backend
5. Aplikasi Web
6. Akses ke berbagai basis data
7. GUI (**Graphical User Interface**)
8. Cloud

Bab 2. Instalasi Rust (wip)

2.1. Tentang Bab Ini (wip)

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.

2.2. Rilis Rust (wip-nr)

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.

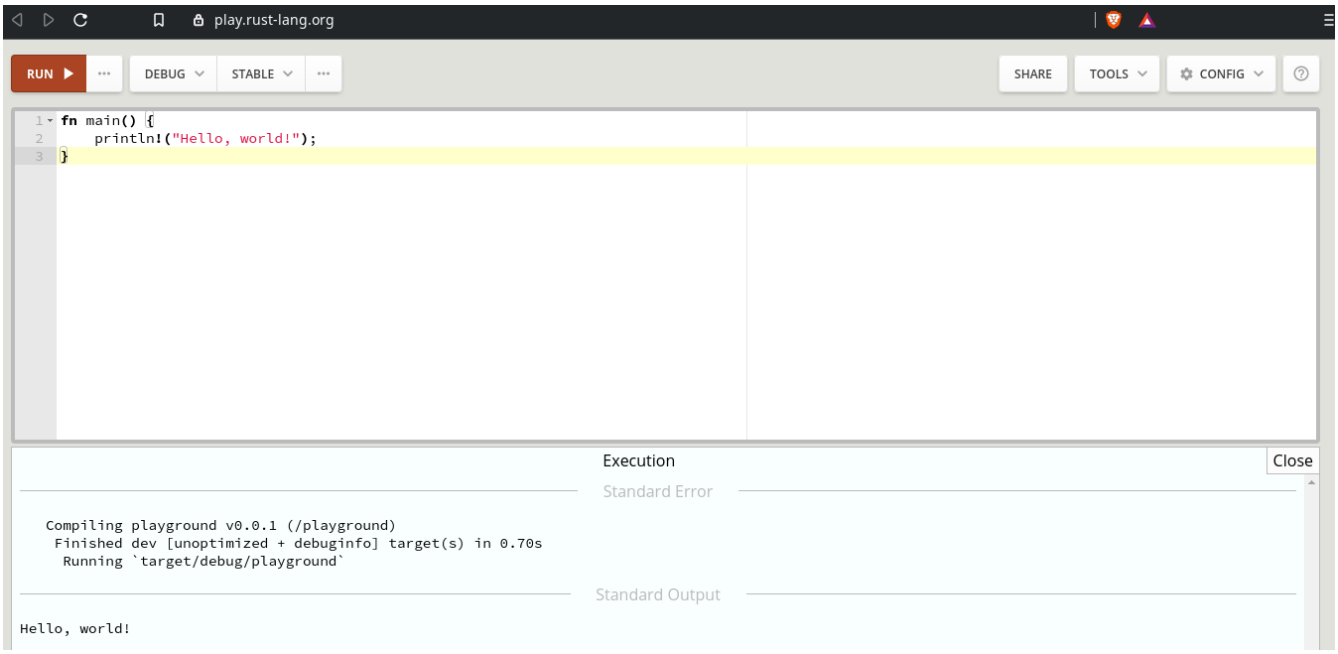
Sebagai contoh, versi 1.44.0 dari versi Rust berisi Rust dengan semua API yang kompatibel dengan versi 1.x.x sebelumnya. Angka 44 berarti penambahan fungsionalitas yang bersifat kompatibel dengan versi penambahan fungsionalitas sebelumnya. Angka 0 berarti sama sekali belum ada perubahan perbaikan *bug*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

2.3. Edisi Rust (wip)

2.4. Rust Playground (wip)

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.



2.5. Instalasi Rust (wip)

src/01-02/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.

2.6. Memahami Ekosistem Rust (wip)

2.7. Pengenalan Cargo (wip)

2.7.1. Inisialisasi Proyek (wip)

2.7.2. Membangun Proyek (wip)

2.7.3. Membersihkan Hasil Kompilasi (wip)

2.7.4. Antara Debug dan Release (wip)

2.7.5. Menjalankan Hasil (wip)

Bab 3. IDE untuk Rust (wip)

3.1. Tentang Bab Ini (wip-nr)

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:

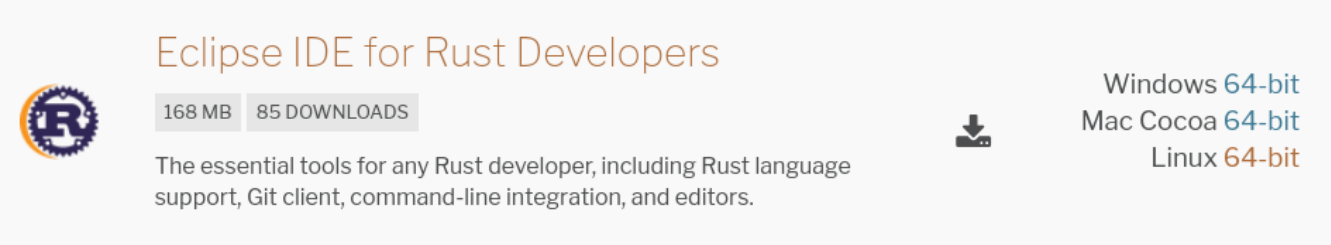
1. Vim
2. Visual Studio Code / VSCodium
3. Eclipse

3.2. Vim (wip)

3.3. Visual Studio Code / VSCodium (wip)

3.4. Eclipse (wip)

Eclipse Foundation juga mempunyai proyek yang didedikasikan untuk membuat IDE Rust dengan basis IDE Eclipse untuk membangun aplikasi menggunakan Rust. Untuk memperoleh software ini, silahkan akses ke https://www.eclipse.org/downloads/packages/ dan pilih [gbr-web-ecipse-rust]. Untuk menjalankan ini, anda seharusnya sudah mempunyai JDK (Java Development Kit) terinstall di komputer anda.



Gambar 2. Paket Eclipse for Rust Developers

Setelah itu, ekstrak hasil download:

```
$ tar -xvf eclipse-rust-2020-06-R-linux-gtk-x86_64.tar.gz
eclipse/
eclipse/p2/
eclipse/p2/org.eclipse.equinox.p2.engine/
eclipse/p2/org.eclipse.equinox.p2.engine/profileRegistry/
eclipse/p2/org.eclipse.equinox.p2.engine/profileRegistry/epp.package.rust.profile/
eclipse/p2/org.eclipse.equinox.p2.engine/profileRegistry/epp.package.rust.profile/1592
224392705.profile.gz
eclipse/p2/org.eclipse.equinox.p2.engine/profileRegistry/epp.package.rust.profile/.loc
k
eclipse/p2/org.eclipse.equinox.p2.engine/profileRegistry/epp.package.rust.profile/.dat
a/
...
...
...
eclipse/dropins/
eclipse/eclipse.ini
eclipse/configuration/
eclipse/configuration/org.eclipse.equinox.simpleconfigurator/
eclipse/configuration/org.eclipse.equinox.simpleconfigurator/bundles.info
eclipse/configuration/config.ini
eclipse/configuration/org.eclipse.update/
eclipse/configuration/org.eclipse.update/platform.xml
$
```

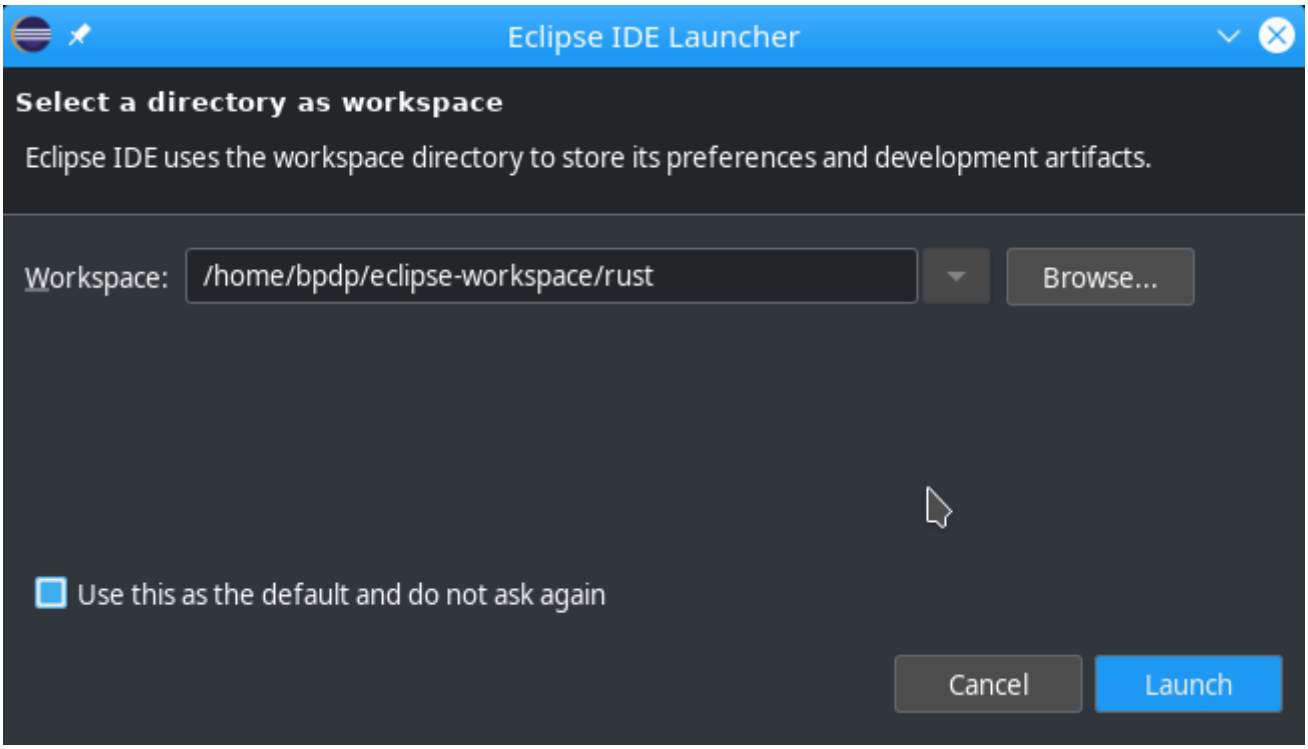
Hasil ekstrak adalah direktori **eclipse** dengan isi sebagai berikut:

```
$ ls -la
total 372
drwxr-xr-x  8 bpdp bpdp   4096 Jun 15 19:33 ./
drwxr-xr-x  3 bpdp bpdp   4096 Jun 18 21:57 ../
-rw-r--r--  1 bpdp bpdp 101678 Jun 15 19:33 artifacts.xml
drwxr-xr-x  4 bpdp bpdp   4096 Jun 15 19:33 configuration/
drwxr-xr-x  2 bpdp bpdp   4096 Jun 15 19:33 dropins/
-rwxr-xr-x  1 bpdp bpdp  61176 Jun  4 22:24 eclipse*
-rw-r--r--  1 bpdp bpdp    630 Jun 15 19:33 eclipse.ini
-rw-r--r--  1 bpdp bpdp     61 Jun  4 20:56 .eclipseproduct
drwxr-xr-x 49 bpdp bpdp   4096 Jun 15 19:33 features/
-rwxr-xr-x  1 bpdp bpdp 140566 Jun  4 22:24 icon.xpm*
drwxr-xr-x  4 bpdp bpdp   4096 Jun 15 19:33 p2/
drwxr-xr-x  9 bpdp bpdp  36864 Jun 15 19:33 plugins/
drwxr-xr-x  2 bpdp bpdp   4096 Jun 15 19:33 readme/
$
```

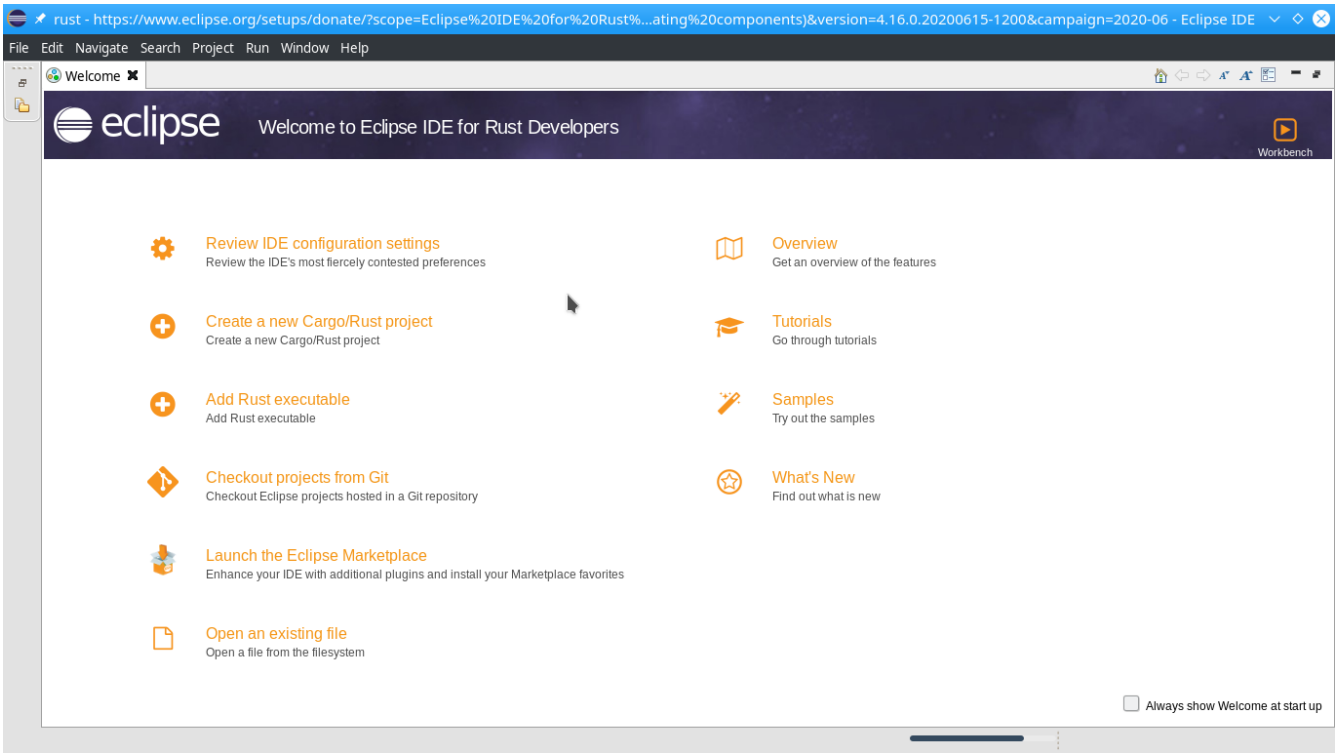
Untuk menjalankan **Eclipse for Rust Developers**, masuk ke direktori hasil ekstrak kemudian eksekusi file **eclipse** sebagai berikut:

```
$ cd eclipse
$ ./eclipse
```

Setelah **splash screen**, Eclipse akan menanyakan **workspace** tempat menyimpan berbagai proyek Rust seperti pada gambar Meneteapkan **workspace** untuk Eclipse for Rust Developers. Isikan seperti yang anda kehendaki (bebas), setelah itu, klik pada **Launch**, maka akan dimunculkan Tampilan pembuka Eclipse for Rust Developers.



Gambar 3. Meneteapkan **workspace** *untuk Eclipse for Rust Developers*



Gambar 4. Tampilan pembuka Eclipse for Rust Developers

Bab 4. Ekosistem Rust (wip)

4.1. Tentang Bab Ini (wip-nr)

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.

4.2. Paket (*Crates*) (wip)

4.3. *Crates* yang Bermanfaat Bagi Pemrogram Rust (wip)

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

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

Evcxr

```
cargo install evcxr_repl
```

4.3.2. IRust (https://github.com/sigmaSd/IRust) (wip)

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

IRust

```
cargo install irust
```

4.4. Berbagai Web untuk Melacak Status dalam Domain Tertentu (wip)

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

4.5. Mengikuti Perkembangan Rust (wip)

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

Bab 5. Struktur Proyek Rust (wip)

5.1. Tentang Bab Ini (wip)

Bab ini membahas tentang strukutur direktori dan file yang ada pada suatu proyek Rust. Direktori serta beberapa file default yang ada di dalamnya merupakan direktori serta file yang dibuat dengan menggunakan **cargo**. Bab ini menjelaskan tentang struktur proyek yang dibuat dengan menggunakan **cargo**.

5.2. Cargo dan Pembuatan Direktori serta File-File Proyek (wip)

Secara umum, **cargo** bisa digunakan untuk membuat 2 tipe proyek Rust:

1. **Binary executable**: jika hasil dari proyek digunakan dengan cara dieksekusi / dijalankan secara langsung.
2. **Library**: jikha hasil dari proyek dimaksudkan sebagai pustaka yang bisa digunakan oleh pemrogram Rust.

Untuk memulai membuat proyek menggunakan Rust, kita bisa menggunakan cargo dengan perintah-perintah berikut ini:

```
$ cargo new bindefault
     Created binary (application) `bindefault` package
$ cargo new --lib libdefault
     Created library `libdefault` package
$
```

Hasil dari perintah-perintah di atas adalah 2 direktori:

1. **bindefault**: untuk proyek yang menghasilkan **binary executable**.
2. **libdefault**: untuk proyek yang menghasilkan pustaka yang bisa digunakan para proyek pengembangan software aplikasi berbasis Rust lainnya.

Perbedaan dari kedua tipe proyek tersebut bisa kita lihat dari struktur serta file-file yang ada pada direktori tersebut:

```
$
.
├──── bindefault
│     ├──── Cargo.toml
│     └──── src
│         └──── main.rs
└──── libdefault
      ├──── Cargo.toml
      └──── src
          └──── lib.rs

4 directories, 4 files

$
```

Berikut adalah penjelasan tentang struktur direktori beserta berbagai file yang ada pada proyek tersebut:

1. **Cargo.toml**: File ini berisi metadata, yaitu data tentang proyek Rust yang aktif.
2. **src**: Direktori ini digunakan untuk menyimpan berbagai **source code** Rust. Pemrogram Rust akan membuat **source code** di direktori ini.

5.3. Proses Build (wip)

Setelah memperoleh struktur direktori default seperti di atas, pemrograman akan meneruskan proses coding dengan menambahkan **source code**, pustaka, dan lain-lain. Setelah itu, akan dilakukan proses **build** untuk mengkompilasi **source code**. Proses build ini akan menghasilkan banyak direktori dan file sesuai dengan versi **build**. Ada 2 versi **build** yang bisa dihasilkan:

1. Versi **debug**: merupakan versi default, dilakukan jika kita tidak menggunakan parameter apapun saat melakukan proses **build**. Hasil dari proses ini merupakan hasil yang masih menyertakan komponen untuk **debugging** pada hasil proses **build**.
2. Versi **release**: merupakan versi **build** yang dilakukan jika menyertakan **--release** sebagai parameter **cargo build**. Hasil dari proses ini merupakan hasil yang siap untuk dirilis dengan optimasi siap untuk dijalankan **end user**. Versi ini tidak menyertakan komponen untuk **debugging**.

```
$ cargo build
    Updating crates.io index
   Compiling libc v0.2.71
   Compiling bitflags v1.2.1
   Compiling gimli v0.21.0
   Compiling cfg-if v0.1.10
   Compiling unicode-width v0.1.7
   Compiling rustc-demangle v0.1.16
   Compiling object v0.19.0
   Compiling ansi_term v0.11.0
   Compiling strsim v0.8.0
   Compiling vec_map v0.8.2
   Compiling smallvec v0.4.5
   Compiling textwrap v0.11.0
   Compiling atty v0.2.14
   Compiling addr2line v0.12.1
   Compiling clap v2.33.1
   Compiling backtrace v0.3.48
   Compiling error-chain v0.10.0
   Compiling ferris-says v0.1.2
   Compiling binproses v0.1.0 (/home/bpdp/kerjaan/git-
repos/oldstager/current/github/rustid/buku-pemrograman-rust/src/01-
05/proses/binproses)
    Finished dev [unoptimized + debuginfo] target(s) in 18.56s
$
```

Versi rilis:

```
$ cargo build --release
   Compiling libc v0.2.71
   Compiling gimli v0.21.0
   Compiling bitflags v1.2.1
   Compiling object v0.19.0
   Compiling rustc-demangle v0.1.16
   Compiling unicode-width v0.1.7
   Compiling cfg-if v0.1.10
   Compiling strsim v0.8.0
   Compiling ansi_term v0.11.0
   Compiling vec_map v0.8.2
   Compiling smallvec v0.4.5
   Compiling textwrap v0.11.0
   Compiling atty v0.2.14
   Compiling clap v2.33.1
   Compiling addr2line v0.12.1
   Compiling backtrace v0.3.48
   Compiling error-chain v0.10.0
   Compiling ferris-says v0.1.2
   Compiling binproses v0.1.0 (/home/bpdp/kerjaan/git-
repos/oldstager/current/github/rustid/buku-pemrograman-rust/src/01-
05/proses/binproses)
    Finished release [optimized] target(s) in 12.21s
$
```

Setelah proses ini, isi dari direktori proyek akan berubah. Jumlah file akan berbeda-beda untuk tiap proyek karena masalah dependensi, tetapi pada umumnya, struktur sama:

```
target/
├─── debug
│    ├─── binproses
│    ├─── binproses.d
│    ├─── build
│    ├─── deps
│    ├─── examples
│    └─── incremental
└─── release
     ├─── binproses
     ├─── binproses.d
     ├─── build
     ├─── deps
     ├─── examples
     └─── incremental

10 directories, 4 files
```

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 6. Variabel dan Komentar (wip)

6.1. Tentang Bab Ini (wip)

Bab 7. Tipe Data (wip)

7.1. Tentang Bab Ini (wip)

Tipe data

Bab 8. Fungsi (wip)

8.1. Tentang Bab Ini (wip)

Fungsi

Bab 9. Pengandali Aliran Program (wip)

9.1. Tentang Bab Ini (wip)

Pengendali aliran program

Bab 10. Struktur Data (wip)

10.1. Tentang Bab Ini (wip)

Struktur data

Bab 11. Ownership (wip)

11.1. Tentang Bab Ini (wip)

Ownership

Bab 12. Enum (wip)

12.1. Tentang Bab Ini (wip)

Enum

Bab 13. Pattern Matching (wip)

13.1. Tentang Bab Ini (wip)

Bab 14. *Reusability* (wip)

14.1. Tentang Bab Ini (wip)

Bab 15. Mengelola *Error* (wip)

15.1. Tentang Bab Ini (wip)

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 16. Mengelola String (wip)

16.1. Tentang Bab Ini (wip)

Bab 17. Pemrograman Konkuren (wip)

17.1. Tentang Bab Ini

Bab 18. Net (wip)

18.1. Tentang Bab Ini (wip)

Bab 19. Path (wip)

19.1. Tentang Bab Ini (wip)

Bab 20. Sistem File (wip)

20.1. Tentang Bab Ini (wip)

Bab 21. Operasi I/O (wip)

21.1. Tentang Bab Ini (wip)

Bab 22. Mengelola Proses (wip)

22.1. Tentang Bab Ini (wip)

Bagian IV: Topik Khusus

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

Bab 23. Akses Basis Data (wip)

23.1. Tentang Bab Ini (wip)

Bab 24. Antarmuka Teks (wip)

24.1. Tentang Bab Ini (wip)

Bab 25. Antarmuka Grafis (wip)

25.1. Tentang Bab Ini (wip)

Bab 26. Pemrograman Web (wip)

26.1. Tentang Bab Ini (wip)

Bab 27. Serialisasi Data (wip)

27.1. Tentang Bab Ini (wip)