Getting started with rust

Rust@Wellcome meetup Feb. 2
jeff-k

Installing rust

Rustup is the official toolchain manager. The recommended way to install it is through this script:

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

By default it will install the toolchain in your home directory and add the compiler to your $PATH:

```
$ which rustc

~/.cargo/bin/rustc
```

Rustup features

Rustup manages the installation of the rustc compiler as well as cargo. It also manages the toolchains:

```
$ rustup show

stable-x86_64-unknown-linux-gnu (default)

nightly-x86_64-unknown-linux-gnu


installed targets for active toolchain

--------------------------------------

wasm32-unknown-unknown

x86_64-unknown-linux-gnu
```

Cargo

Cargo is the official package manager for rust. Rust packages can contain multiple “crates”. and the official package registry is https://crates.io/.

It also handles dependencies and compilation for your projects.

To create a new package with Cargo:

```
$ cargo new hello_world
```

To make an existing directory into a package:

```
$ git clone git@github.com:example/test_package.git
```

```
$ cd test_package; cargo init
```

Cargo

This will create an empty template for a standalone binary project:

```
$ ls hello_world/*
hello_world/Cargo.toml

hello_world/src:
main.rs
```

Cargo

```
$ cat Cargo.toml


[package]
name = "hello_world"
version = "0.1.0"
edition = "2021"


[dependencies]
```

Compiling with Cargo

```
$ cargo build
   Compiling hello_world v0.1.0 (~/hello_world)
    Finished dev [unoptimized + debuginfo] target(s) in 0.93s
$ cargo build --release
   Compiling hello_world v0.1.0 (~/hello_world)
    Finished release [optimized] target(s) in 0.21s
$ ls target/
CACHEDIR.TAG  debug  release
```

Linting, testing, and documentation

- `cargo fmt`
- `cargo clippy`
- `cargo test`
  - Define test cases with the `#[test]` macro:

```
#[cfg(test)]

mod tests {

    #[test]

    fn always_passes() {

        assert_eq!(true, true);

    }

}
```

- `cargo doc`
  - Builds documentation from special comments in the code
    - `///`
    - `//!`
    - `/** … */`

Rust and Github

- Put `target/` in `.gitignore`, and maybe `Cargo.lock`
- Publishing to Crates.io only works with a github account
  - Generate API token on github and login with `cargo login`
  - Publish with `cargo publish`
- Use github CI to run tests, build releases, and report coverage with tarpaulin
  - Cf. rust-bio: (.github/workflows/rust.yml)

```
On:
…
 pull_request:
   branches: [ master ]
jobs:
 Formatting:
   runs-on: ubuntu-latest
   steps:
…
     - name: Check format
       run: cargo fmt -- --check
```