Yocto Project®: Meta-Rust --> OE-Core

Randy MacLeod, Wind River Systems

Yocto Project Virtual Summit Europe, October 29-30, 2020
YP Rust: Overview

• What is Rust?
• Why does YP need to support Rust?
• What is the meta-rust layer?
• Merging meta-rust to oe-core.
• What’s next?
What is Rust?

Rust is a memory and concurrency safe, performance-oriented systems language.
What is the Rust?

Rust was started as a personal project by Graydon Hoare in 2006. The Mozilla Foundation adopted it in 2010, in part as a response to the difficulties of maintaining the FireFox web browser, a very large multi-threaded C++ project. The project has been widely adopted, hit 1.0 in 2015, and has been the ‘most beloved language’ of a majority of developers for several years in StackOverflow’s annual survey.

Rust promises performance, control, memory safety, and fearless concurrency!
YP Rust: Cool Features

- **Performance**: Similar to C/C++
- **Control**: Low-level language suitable for OS/system development. There is no run-time garbage collector.
- **Memory Safety**: >99.44% of errors caught at build time!
- **Fearless concurrency**: Due to memory safety and type-checking, many concurrency errors are compile-time errors.
- **Productivity**: Cargo build tool, clear docs, helpful errors messages.
Hello Rust! Use https://rustup.rs/ to get started.

- **hellorust.rs**

```rust
fn main() {
    // Print text to the console (println! is a macro so it ends with !).
    println!("Hello Rust!");
}
```

- **rustc -o hellorust hellorust.rs**

```
$ ldd hellorust: libgcc_s.so, libpthread.so, libdl.so, libc.so
$ size hellorust: 280K (stripped)
Can be <6K, see:
https://lifthrasiir.github.io/rustlog/why-is-a-rust-executable-large.html
```
Hello Cargo! Rust package tool: build, deps, fetch, etc

- cargo new hello-cargo; cd hello-cargo; cat src/main.rs

```rust
fn main() {
    println!("Hello, world!");
}
```

- cargo build [-release] → build binary/library and deps
- cargo run → run the binary
- cargo test → run all tests
- cargo search tipc → tipc ="0.1.1" # Bindings for Lx TIPC...
Hello Cargo!! Rust package tool: build, deps, fetch, etc

$ cargo install ripgrep # faster implementation of grep → R.I.P. grep
  Updating crates.io index
  Downloaded ripgrep v12.1.1
  Downloaded 1 crate (256.5 KB) in 1.86s
  Installing ripgrep v12.1.1
  Downloaded grep-regex v0.1.8
  ...
  Downloaded 14 crates (2.4 MB) in 16.68s (largest was `encoding_rs` at 1.4 MB)
  Compiling memchr v2.3.3
  ...
  Compiling grep v0.2.7
  Finished release [optimized + debuginfo] target(s) in 2m 57s
  Installing /home/rmacleod/.cargo/bin/rg
  Installed package `ripgrep v12.1.1` (executable `rg`)
$ rg --help
Why Does Yocto need Rust support?

More and more software is being implemented in Rust so it’s time to merge Rust to OE-Core.
YP Rust → oe-core: Pros/Cons?

- P: oe-core package update held back (gnome: librsvg, etc).
- P: Top language for a few years on StackOverflow.
- P: Safe, performant language: good match for embedded.
- C/P: Another toolchain to support.
- C: Longer builds: getting better but this is a cost of safety
- C: Static linking default (so monolithic apps like busybox).
Hello Cargo!! Rust package tool: build, deps, fetch, etc

$ sudo apt install gcc-9-aarch64-linux-gnu
$ rustup target add aarch64-unknown-linux-gnu
$ cat ~/.cargo/config
[target.aarch64-unknown-linux-gnu]
l linker = "aarch64-linux-gnu-gcc-9"
$ cargo build --target=aarch64-unknown-linux-gnu
  Compiling hello-cargo v0.1.0 (hello-cargo)
  Finished dev [unoptimized + debuginfo] target(s) in 1.44s
$ file target/aarch64-unknown-linux-gnu/debug/hello-cargo
  ELF 64-bit LSB shared object, ARM aarch64, version 1 (SYSV), ...
# copy to Rpi4/64bit
rpi4 $ /tmp/hello-cargo
Hello, world!

But get the linker from Yocto SDK or ALL from SDK.
What is the meta-rust layer?

The meta-rust layer support Rust and Cargo in a Yocto-compatible manner.
Meta-rust: What is it, What is it? (Talking Heads song!)

- Meta-rust is a YP layer provides rustc/cargo X-tools.
- Started in 2014 by Cody P Schafer. Doug Goldstein, Derek Straka, Steven Walter, and more: 35 Authors,
- ~40 commits/year recently.
- Rust is written in Rust so need a bootstrap TC (arm64?).
- Typical stage 1, stage 2 toolchain build.
- Sample hello-rust app to test single binary.
Meta-rust: What is it?

• Build core-image-minimal, then rust-hello-world
Meta-rust: What is ‘cargo bitbake’?

See: https://github.com/meta-rust/cargo-bitbake

$ cargo install cargo-bitbake
$ cd .../ripgrep.git
$ cargo bitbake

Wrote: ripgrep_12.1.1.bb
$ cat ripgrep_12.1.1.bb
inherit cargo
SRC_URI += "git://github.com/BurntSushi/ripgrep;protocol=https;nobranch=1"
SRCREV = "7cb211378a2ac6d421c5f6f3f71411937af23136"
...
SRC_URI += " \
    crate://crates.io/aho-corasick/0.7.10 \n    crate://crates.io/atty/0.2.14 \n...

Just add to your layer and:
$ bitbake ripgrep
Merging meta-rust to oe-core.

Merging meta-rust to oe-core will provide YP support & greater visibility to Rust for YP users.
Move commits from meta-rust to oe-core: **Done.**
Add libgit2, libssh2 from meta-oe for Rust: **Done.**
Uprev librsvg: C API + Rust core library: Done?

- **Currently rustc doesn’t build the ‘Rust core’ files:**

```
$ tmp-glibc/sysroots-components/x86_64/rust-native/usr/bin/rustc -print cfg
error: Error loading target specification:
   Could not find specification for target "x86_64-linux".
```

- **CARGO_DISABLE_BITBAKE_VEND Kelvin** = "1"
- Uprev packages once librsvg uprev works: gstreamer, ...
Meta-rust → oe-core: Slide 2/2

• QemuRiscv64 support: Perhaps llvm-11.0 fixes this.
• Support Arm64 as build: May just need testing.
• Make desktop ‘rustup’ work with YP toolchain or just test Cargo/rustc from YP SDK?
• Parallelize rust build. It really is too slow.
What’s next for Rust in YP?

1. Get it merged, 2. ???, 3. Profit! ;-)

Yocto Project | The Linux Foundation
What’s next/missing for Rust in YP?

• Continue to track upstream releases: 6 weeks.
• Unify rust-llvm, llvm: There can only be one!
• Rustc/cargo self-test support for -native?
• Enable build of rustc/cargo binaries for target.
• Support crate dynamic linking?
• Kernel driver support.
• All the things that I don’t know about. Ideas?
Thanks for your time
More Ideas!

- Make meta-tc-bootstrap layer for GCC, Rust, Clang?
- Gnu Mes project to bootstrap
  - Mes: 5000 LOC in C → Scheme → MesCC C compiler
  - MesCC → TinyCC → \{glibc-2.2.5 binutils-2.20.1 gcc-2.95.3\}
  - Build later versions of GCC
- Multi-stage rustc bootstrap
  - MRustC: alternative Rust compiler written in C++
  - Use mrustc → 1.29.2...1.47.0:
    See: https://github.com/dtolnay/bootstrap