한Welcome to Comprehensive Rust

> 출처 : https://google.github.io/comprehensive-rust/welcome.html

안드로이드 팀에 의해 개발된 4일 러스트 코스이다. 이 코스는 러스트의 기본 문법에서 제네릭과 에러 처리와 같은 고급 주제 전체 스펙트럼을 다룬다. 마지막 날에는 Android 전용 콘텐츠도 포함되어 있다.

이 코스의 목표는 러스트를 가르치는 것이다. Rust에 대해 아무것도 모른다고 가정하고 다음을 희망한다.

- > Rust 구문 및 언어에 대한 포괄적인 이해를 제공한다.
- Rust에서 기존 프로그램을 수정하고 새 프로그램을 작성할 수 있다.
- 일반적인 Rust 관용구를 보여준다.

4일 차에는 다음과 같은 Android 관련 사항을 다룬다.

- 러스트로 안드로이드 구성 요소(Components)를 빌드한다.
- AIDL 서버와 클라이언트
- C, C++과 Java와 상호 운용성

이 코스는 러스트로 안드로이드 어플리케이션 개발을 다루지 않는 것을 주의하는것이 중요하다. 그리고 Android 관련 부분은 특히 운영 체제인 Android 자체에 대한 코드 작성에 관한 것입니다.

Non-Goal

러스트는 큰 언어이고 몇 일안에 모든 부분을 다룰 수는 없다. 이 코스의 몇몇 비목표는 다음과 같다.

- 비동기(async) 러스트를 사용하는 방법을 배우기 - 전통적인 동시성 기초요소를 다룰 때만 비동기 러스트를 언급할 것이다. 대신에 이 주제의 자세한 것은 러스트로 비동기 프로그래밍을 보세요.
- 매크로를 개발하는 방법을 배우기 - 대신에 Chapter 19.5 in the Rust Book 과 Rust by Example을 보세요.

Assumptions

이 코스는 프로그램 하는 방법을 이미 알고 있다고 가정한다. 러스트는 정적 타입 언어이고 러스트 접근 방식을 좀 더 잘 설명하거나 대조하기 위해 때때로 C/C++과 비교할 것이다.

Python or JavaScript와 같은 동적 타입 언어로 프로그램 하는 방법을 알고 있다면 잘 따라 갈 수 있을 것이다.

1. Running the Course

> This page is for the course instructor.

구글에서 내부적으로 이 코스를 어떻게 운영해 왔는지에 관한 약간의 배경 정보이다.

이 코스를 운영하기위해 필요한 것은

1. 과정 자료를 숙지하는 것이다. 핵심 사항을 강조하는 데 도움이 되도록 일부 페이지에 발표자 노트를 포함했다.(더 많은 발표자 노트를 제공하여 저희를 도와주시오!) 팝업에서 발표자 노트를 열어야 한다.("발표자 노트" 옆에 있는 작은 화살표가 있는 링크를 클릭한다.) 이 방법은 이 과정을 발표하는데 깔끔한 화면을 얻게 된다.
2. 날짜를 결정하라. 이 코스가 커진 때부터 2주에 걸쳐 4일을 예약하는 것이 좋다. 과정 참가자들은 우리가 제공하는 모든 정보를 처리하는 데 도움이 되기 때문에 과정에 공백이 있는 것이 도움이 된다고 말했습니다.
3. 대면 참가자를 위해 충분히 큰 방을 찾아라. 15~20 명이 한 반이였으면 좋겠다. 사람들이 편안하게 질문할 수 있을 만큼 적다. 또한 한 명의 강사가 질문에 답할 시간을 가질 수 있을 만큼 충분히 작다.
4. 코스 당일에는 준비를 위해 조금 일찍 방에 와라. 노트북에서 실행되는 `mdbook serve`를 사용하여 직접 발표하는 것을 권장한다. 이렇게 하면 페이지를 변경할 때 지연 없이 최적의 성능을 보장한다. 노트북을 사용하면 본인이나 코스 참가자가 오타를 발견할 때 오타를 수정할 수도 있다.
5. 사람들이 스스로 또는 소그룹으로 연습문제를 풀게하라. 사람들에게 문제가 있는지 또는 도울 수 있는 일이 있는지 물어보라. 많은 사람들이 같은 문제를 보게 될때에 학급 문제를 제기하고

솔루션을 제공한다. 예를 들어 사람들에게 표준 라이브러리에서 관련 정보를 찾을 수 있는 위치를 보준다.

6. 4일차에 안드로이드 전용 부분을 건너뛰지 않는다면 AOSP checkout이 필요할 것이다. 동일한 머신에서 과정 저장소를 체크아웃하고 src/android/ 디렉토리를 AOSP 체크아웃의 루트로 이동한다. adb 동기화가 에뮬레이터 또는 실제 기기에서 작동하는지 확인하고 src/android/build_all.sh를 사용하여 모든 Android 예시를 미리 빌드하라. 실행되는 명령을 보려면 스크립트를 읽고 수동으로 실행할 때 작동하는지 확인해주세요.

저게 전부인데, 과정을 운영하는 행운을 응원한다! 우리가 그랬던 것처럼 여러분도 재미있기를 기대한다.

과정을 계속 개선할 수 있도록 나중에 피드백을 제공해주세요. 우리는 당신에게 무엇이 잘 되었고 무엇이 더 나아질 수 있는지 듣고 싶어요. 귀하의 학생들도 저희에게 피드백을 보내주시는 것을 매우 환영합니다.

1.1. Course Structure

> This page is for the course instructor.

이 코스는 빠르게 진행되고 많은 영역을 다룬다.

- Day 1: 기본 러스트, 소유권(Ownership)과 차용 검사기(Borrow checker)
- Day 2: 복합 데이터 타입, 패턴 매칭(pattern patching), 표준 라이브러리
- Day 3: Traits와 제네릭(generics), 에러 처리, 테스팅, unsafe 러스트.
- Day 4: 러스트의 동시성 및 다른 언어와의 상호 운영성

Format

이 과정은 매우 상호 작용적이며 우리는 질문이 Rust를 탐구하도록 하는 것을 권장한다.

1.2. Keyboard Shortcuts

There are several useful keyboard shortcuts in mdBook:

- Arrow-Left: Navigate to the previous page.
- Arrow-Right: Navigate to the next page.
- Ctrl + Enter: Execute the code sample that has focus.
- s: Activate the search bar.

2. Using Cargo

러스트에 관해 읽기 시작했을때에 곧 Cargo를 만날 것이다. Cargo는 러스트 생태계에서 사용되는 표준 툴로 러스트 어플리케이션을 빌드하고 실행한다. 여기서 우리는 Cargo가 무엇인지, 더 넓은 생태계에 어떻게 적용되는지, 그리고 어떻게 이 교육에 적용되는지에 대한 간략한 개요를 제공하고자 한다.

이건 그냥 https://google.github.io/comprehensive-rust/cargo.html or 구글링하면 많이 나오니 따라하면 될 듯. 더 이상의 설명은 생략한다.

2.1. Rust Ecosystem

Rust 생태계는 여러 도구로 구성되어 있으며 주요 도구는 다음과 같다:

- `rustc`: .rs 파일을 바이너리 및 기타 중간 형식으로 변환하는 Rust 컴파일러
- `cargo`: Rust 종속성 관리자 및 빌드 도구. Cargo는 https://crates.io에 호스팅된 종속성을 다운로드하는 방법을 알고 있으며 프로젝트를 빌드할 때 이를 rustc로 전달한다. Cargo에는 단위 테스트를 실행하는 데 사용되는 내장 테스트 러너도 함께 제공된다.
- `rustup`: 러스트 toolchain 인스톤러와 업데이터. 이 툴은 `rustc`와 `cargo`가 러스트의 새로운 버전이 릴리즈될때에 인스톨하고 업데이트에 사용된다. 추가적으로 rustup은 표준 라이브러리를 위한 문서도 다운로드할 수 있다. 한 번에 여러 버전의 Rust를 설치할 수 있으며 rustup을 사용하면 필요에 따라 버전 간에 전환할 수 있다.

2.2. Code Samples

이번 트레이닝 동안에 주로 브라우저를 통해 실행할 수 있는 예제를 통해 러스트 언어를 탐색한다.

`Cargo`설치는 여전히 격려된다. 이것은 연습을 하기에 더 쉽게 한다. 마지막 날에는 종속성을 다루는 방법과 이를 위해 `Cargo`가 필요한 방법을 보여주는 더 큰 연습을 할 것이다.

이번 코스에 있는 코드 블럭은 완전히 상호 작용한다.

```
fn main() {
  println!("Edit me!");
}
```

이건 https://google.github.io/comprehensive-rust/cargo/code-samples.html 에서 해보세요.

2.3. Running Cargo Locally

만약 갖고 있는 시스템에서 코드를 경험하고 싶다면 첫번째 러스트를 설치할 필요가 있다. Rust Book의 지침에 따라 이를 수행하라. 이것은 `rustc`와 `c`로 일하는걸 제공한다.

```
geayoungbaek@geayoungui-MacBookPro 003-Comprehensive-Rust % rustc --version
rustc 1.66.1 (90743e729 2023-01-10)
geayoungbaek@geayoungui-MacBookPro 003-Comprehensive-Rust % cargo --version
cargo 1.66.1 (ad779e08b 2023-01-10)
```

이 자리에서, 그런 다음 다음 단계에 따라 이 교육의 예제 중 하나에서 Rust 바이너리를 빌드한다.

1. 복사하려는 예제에서 "클립보드에 복사" 버튼을 클릭.
2. 코드에 새로운 `exercise/` 디렉토리를 만들기 위해서 `cargo new exercise` 를 사용

   ```
   MacBookPro 003-Comprehensive-Rust % cargo new 2_3_exercise --name
   exercise
        Created binary (application) `exercise` package
   ```

3. `exercise/`로 들어가서 바이너리를 빌드하고 실행하기 위해서 `cargo run`을 사용

   ```
   MacBookPro 2_3_exercise % cargo run
      Compiling exercise v0.1.0 (/Users/geayoungbaek/Project/Rust/003-
   Comprehensive-Rust/2_3_exercise)
       Finished dev [unoptimized + debuginfo] target(s) in 16.44s
        Running `target/debug/exercise`
   Hello, world!
   ```

4. 코드에서 `src/main.rs` 상용구(boiler-plate코드를 교체. 예를 들어 아래 `src/main.rs` 처럼 만드는 이전 페이지에서 예지를 사용

   ```rust
   fn main() {
       println!("Edit me");
   }
   ```

5. 업데이트된 바이너리를 빌드하고 실행하기위해 `cargo run`을 사용

```
MacBookPro 2_3_exercise % cargo run
   Compiling exercise v0.1.0 (/Users/geayoungbaek/Project/Rust/003-
Comprehensive-Rust/2_3_exercise)
    Finished dev [unoptimized + debuginfo] target(s) in 1.25s
     Running `target/debug/exercise`
Edit me
```

6. 에러를 위해 프로젝트를 빠르게 확인하기 위해 `cargo check`를 사용, 실행 없이 컴파일을 위해 `cargo build`를 사용. 일반 디버그 빌드를 위해 `target/debug`에서 결과물을 발견 할 것이다. target/release/에 최적화된 릴리즈 빌드를 만들기 위해서 `cargo build --relase`를 사용

7. `Cargo.toml`을 수정하는 것으로 프로젝트에 의존성들을 추가할 수 있다. `cargo` 명령을 실행 할때에 누락된(missing) 의존성들을 자동적으로 다운로드하고 컴파일 할 것이다.

3. Welcome

Welcome to Day 1

종합 러스트의 첫번째 날이다. 오늘 많은 영역을 다룰 것이다.

- 기본 러스트 구문 : 변수, 스칼라 그리고 복합 타입, enum, struct, 참조, 함수 그리고 메서드
- 메모리 관리 : 스택 vs 힙, 수동 메모리 관리, 범위 기반 메모리 관리와 가비지 컬렉션
- 소유권(Ownership) : 이동 시맨틱, 복사와 복제, 차용(borrowing) 과 수명(lifetime)

3.1. What is Rust?

러스트는 2015년에 1.0 릴리즈된 새로운 프로그래밍 언어이다.

- 러스트는 C++로 유사한 역할을 하는 정적으로 컴파일된 언어이다.
-