

Korea University MatKor 스터디

# Introduction

Chris Ohk

[utilForever@gmail.com](mailto:utilForever@gmail.com)

# 발표자 소개

Korea University MatKor 스터디  
Introduction

- 옥찬호 (Chris Ohk)
  - (현) EJN Tech Lead
  - (전) Momenti Engine Engineer
  - (전) Nexon Korea Game Programmer
  - Microsoft Developer Technologies MVP
  - C++ Korea Founder & Administrator
  - Reinforcement Learning KR Administrator
  - IT 전문서 집필 및 번역 다수
    - 게임샐러드로 코드 한 줄 없이 게임 만들기 (2013)
    - 유니티 Shader와 Effect 제작 (2014)
    - 2D 게임 프로그래밍 (2014), 러스트 핵심 노트 (2017)
    - 모던 C++ 입문 (2017), C++ 최적화 (2019)

utilForever@gmail.com



utilForever



- 주 교재
  - Comprehensive Rust (Google, 2023)
  - Programming Rust (O'Reilly Media, 2021)
  - The Rust Programming Language, 2nd Edition (No Starch Press, 2023)
- 부 교재
  - Rust in Action (Manning, 2021)
  - Rust for Rustaceans (No Starch Press, 2021)

- Rust 기초 강의 리뉴얼
  - 주 교재를 **A Tour of Rust**에서 **Comprehensive Rust**로 변경
  - 주 교재에서 필요하지만 알려주지 않는 부분 추가 설명
  - 주 교재에서 빠진 부분 별도로 추가
    - 동시성 프로그래밍
    - 클로저, 매크로
    - FFI (Foreign Function Interface)
    - Rust + WebAssembly

- 진행 요일 및 시간
  - 강의 요일은 월요일 오후 8시 시작
  - 강의 시간은 보통 2시간 내외, 최대 2시간 30분
- 참고 사항
  - 진행자의 개인 사정에 따라 스터디 일정이 변경될 수 있음
  - 스터디를 시작했을 때 참석자가 저조할 경우 스터디 일정을 연기할 수 있음

- Week 1 (3/18)
  - Hello, World
  - Types and Values
  - Control Flow Basics
  - Tuples and Arrays
  - References
  - User-Defined Types
  - Assignment #1

- Week 2 (3/25)
  - Pattern Matching
  - Methods and Traits
  - Generics
  - Standard Library Types
  - Standard Library Traits
  - Assignment #2

- Week 3 (4/1)
  - Memory Management
  - Smart Pointers
  - Borrowing
  - Slices and Lifetimes
  - Assignment #3



- Week 4 (4/8)
  - Iterators
  - Modules
  - Testing
  - Error Handling
  - Unsafe Rust
  - Assignment #4

- Week 5 (4/29)
  - Threads
  - Channels
  - Send and Sync
  - Shared State
  - Async Basics
  - Control Flow
  - Pitfalls
  - Assignment #5

- Week 6 (5/13)
  - Closures
  - Macros
  - Assignment #6

- Week 7 (5/20)
  - FFI (Foreign Function Interface)
  - Rust + WebAssembly
  - Assignment #7

- Week 8 (5/27)
  - Make a Blog, Part 1
- Week 9 (6/3)
  - Make a Blog, Part 2
- Week 10 (6/10)
  - Make a Blog, Part 3

# Rust란?

---

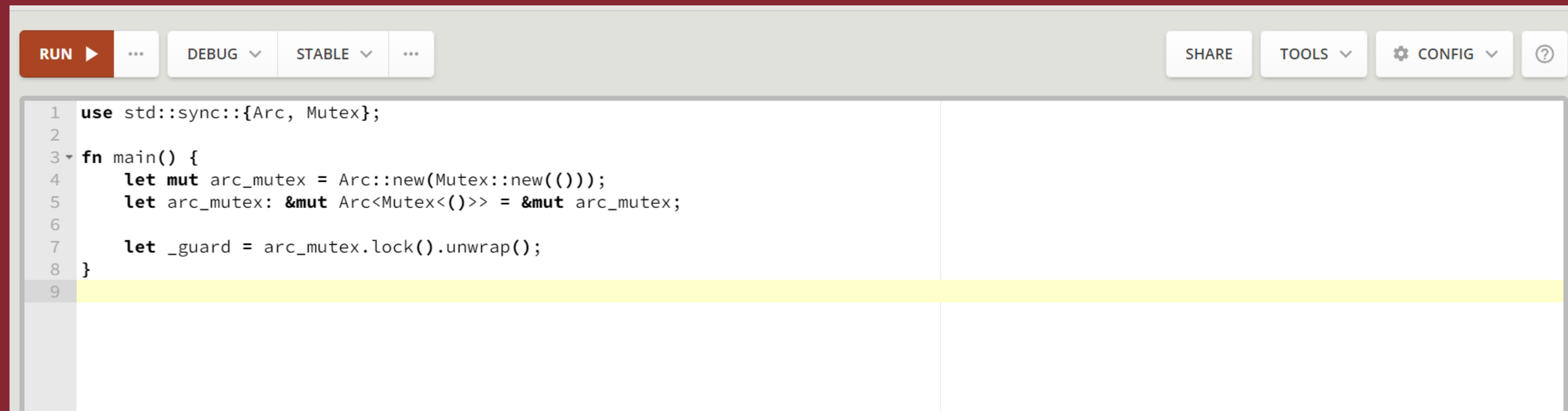
- <https://www.rust-lang.org/>
- 모질라 재단에서 2010년 7월 7일 처음 발표
- 현재는 러스트 재단으로 독립해서 개발되고 있다.
- Rust 언어의 특징
  - 안전한 메모리 관리
  - 철저한 예외나 에러 관리
  - 특이한 enum 시스템
  - 트레이트
  - 하이지닉 매크로
  - 비동기 프로그래밍
  - 제네릭



# Rust Playground

Korea University MatKor 스터디  
Introduction

- <https://play.rust-lang.org/>



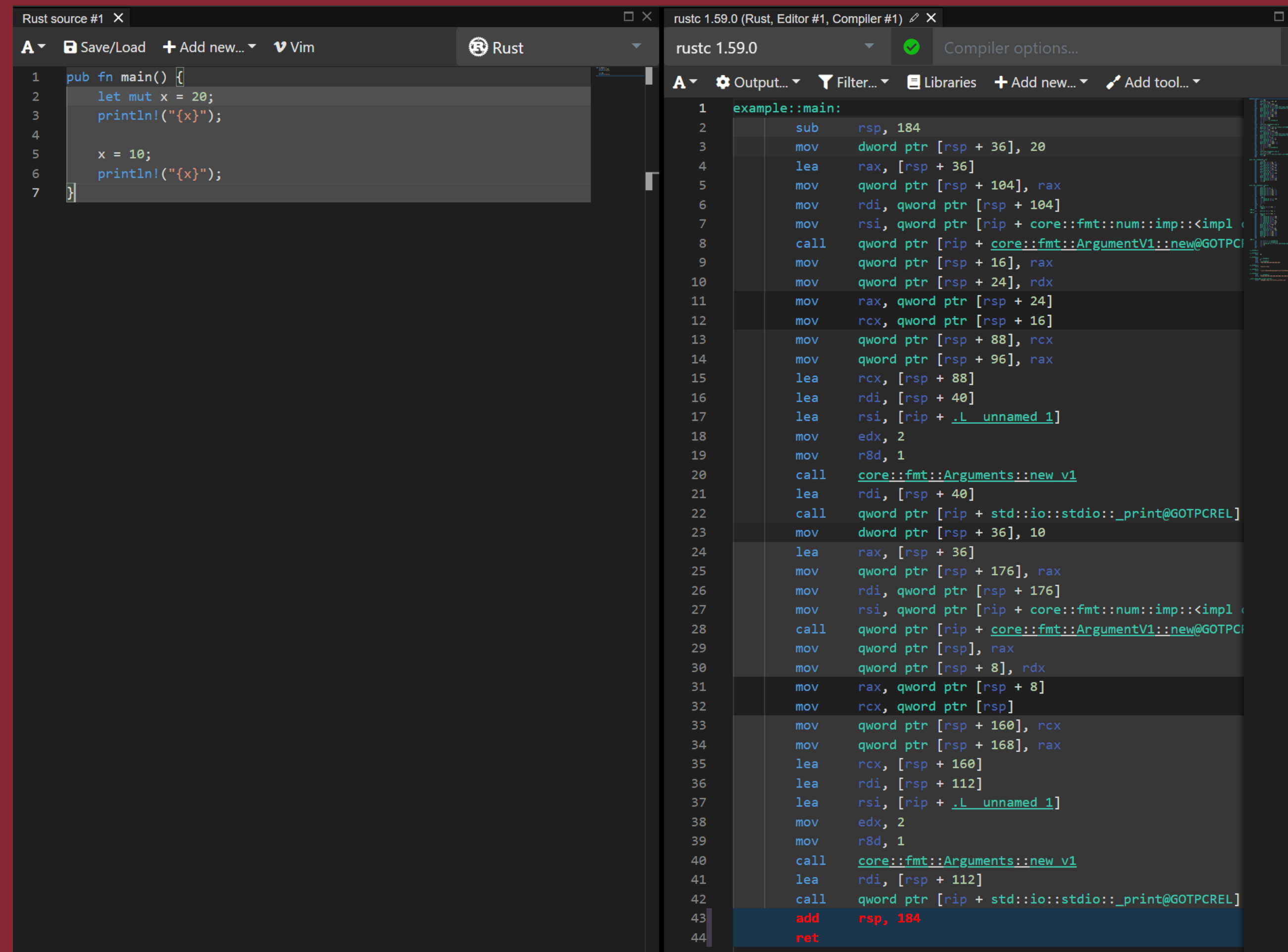
The screenshot shows the Rust Playground interface. At the top, there is a toolbar with buttons for 'RUN' (with a play icon), 'DEBUG' (with a dropdown arrow), 'STABLE' (with a dropdown arrow), and a help icon. To the right of these are buttons for 'SHARE', 'TOOLS' (with a dropdown arrow), 'CONFIG' (with a dropdown arrow), and a question mark icon. Below the toolbar is a text editor with a light gray background. The code in the editor is as follows:

```
1 use std::sync::{Arc, Mutex};
2
3 fn main() {
4     let mut arc_mutex = Arc::new(Mutex::new(()));
5     let arc_mutex: &mut Arc<Mutex<()>> = &mut arc_mutex;
6
7     let _guard = arc_mutex.lock().unwrap();
8 }
9
```

The line numbers 1 through 9 are visible on the left side of the editor. The code is syntactically correct Rust code that demonstrates the use of an Arc-wrapped Mutex.

# Rust 디스어셈블리

- <https://rust.godbolt.org/>



The screenshot displays the Rust Godbolt compiler interface. The left pane shows the Rust source code for a `main` function, and the right pane shows the corresponding x86-64 assembly code generated by `rustc 1.59.0`.

```
1 pub fn main() {  
2     let mut x = 20;  
3     println!("{}", x);  
4  
5     x = 10;  
6     println!("{}", x);  
7 }
```

```
1 example::main:  
2     sub     rsp, 184  
3     mov     dword ptr [rsp + 36], 20  
4     lea     rax, [rsp + 36]  
5     mov     qword ptr [rsp + 104], rax  
6     mov     rdi, qword ptr [rsp + 104]  
7     mov     rsi, qword ptr [rip + core::fmt::num::imp::<impl ...  
8     call    qword ptr [rip + core::fmt::ArgumentV1::new@GOTPC...  
9     mov     qword ptr [rsp + 16], rax  
10    mov     qword ptr [rsp + 24], rdx  
11    mov     rax, qword ptr [rsp + 24]  
12    mov     rcx, qword ptr [rsp + 16]  
13    mov     qword ptr [rsp + 88], rcx  
14    mov     qword ptr [rsp + 96], rax  
15    lea     rcx, [rsp + 88]  
16    lea     rdi, [rsp + 40]  
17    lea     rsi, [rip + .L_unnamed_1]  
18    mov     edx, 2  
19    mov     r8d, 1  
20    call    core::fmt::Arguments::new_v1  
21    lea     rdi, [rsp + 40]  
22    call    qword ptr [rip + std::io::stdio::_print@GOTPCREL]  
23    mov     dword ptr [rsp + 36], 10  
24    lea     rax, [rsp + 36]  
25    mov     qword ptr [rsp + 176], rax  
26    mov     rdi, qword ptr [rsp + 176]  
27    mov     rsi, qword ptr [rip + core::fmt::num::imp::<impl ...  
28    call    qword ptr [rip + core::fmt::ArgumentV1::new@GOTPC...  
29    mov     qword ptr [rsp], rax  
30    mov     qword ptr [rsp + 8], rdx  
31    mov     rax, qword ptr [rsp + 8]  
32    mov     rcx, qword ptr [rsp]  
33    mov     qword ptr [rsp + 160], rcx  
34    mov     qword ptr [rsp + 168], rax  
35    lea     rcx, [rsp + 160]  
36    lea     rdi, [rsp + 112]  
37    lea     rsi, [rip + .L_unnamed_1]  
38    mov     edx, 2  
39    mov     r8d, 1  
40    call    core::fmt::Arguments::new_v1  
41    lea     rdi, [rsp + 112]  
42    call    qword ptr [rip + std::io::stdio::_print@GOTPCREL]  
43    add     rsp, 184  
44    ret
```



- Windows
  - 32bit : <https://static.rust-lang.org/rustup/dist/i686-pc-windows-msvc/rustup-init.exe>
  - 64bit : [https://static.rust-lang.org/rustup/dist/x86\\_64-pc-windows-msvc/rustup-init.exe](https://static.rust-lang.org/rustup/dist/x86_64-pc-windows-msvc/rustup-init.exe)
- Windows Subsystem for Linux
  - `curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh`
- Linux and MacOS
  - `curl https://sh.rustup.rs -sSf | sh -s -- --help`

- 여러 프로그램에서 Rust를 사용할 수 있다. 두 프로그램을 많이 사용한다.
  - Visual Studio Code
  - JetBrains RustRover
- Visual Studio Code와 같이 사용하면 좋은 Extensions
  - rust-analyzer
  - crates
  - Even Better TOML

# Cargo 프로젝트 만들기

---

- 바이너리 파일을 생성하는 프로젝트
  - `cargo new [프로젝트명]`
  - Cargo.toml과 main.rs가 생성됨
- 라이브러리 파일을 생성하는 프로젝트
  - `cargo new [프로젝트명] --lib`
  - Cargo.toml과 lib.rs가 생성됨

- 프로젝트 빌드 방법
  - `cargo build (--release)`
  - 기본은 디버그 모드로 빌드된다.
  - 릴리즈 모드로 빌드하고 싶다면 `--release`를 추가하면 된다.
- 프로젝트 실행 방법
  - `cargo run (--release)`
  - 기본은 디버그 모드로 빌드 후 실행된다.
  - 릴리즈 모드로 빌드 후 실행하고 싶다면 `--release`를 추가하면 된다.

- rustfmt
  - Rust 팀에서 개발, 관리하고 있는 공식 포맷터 (Formatter)
  - 공식 스타일 가이드라인을 참고해서 자동으로 코드 스타일을 수정할 수 있다.
  - `cargo fmt`
- clippy
  - Rust 팀에서 개발, 관리하고 있는 코드 린터 (Linter)
  - 현재 코드의 문제점을 파악하고, 자동으로 수정할 수 있다.
  - `cargo clippy`

- audit
  - Rust로 만들어진 소프트웨어의 보안 취약점들을 확인하는 도구
  - `cargo audit`
- test
  - Rust 코드의 단위 또는 통합 테스트를 수행하는 도구
  - `cargo test`
- tarpaulin
  - 코드 커버리지를 쉽게 측정할 수 있는 도구
  - `cargo tarpaulin --ignore-tests`

Thank you!