

Overview

Rust is a general-purpose programming language focused on performance, memory safety without a garbage collector, and concurrency, using an ownership system to prevent memory errors.

Strengths:

- Compiled code about same performance as C / C++, and excellent memory and energy efficiency.
- Can avoid 70% of all safety issues present in C / C++, and most memory issues.
- Strong type system prevents data races, brings 'fearless concurrency' (amongst others).
- Seamless C interop, and dozens of supported platforms (based on LLVM).

Installation:

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

Toolbox

**Rustup** is used to install and manage Rust toolchains. Toolchains are complete installations of Rust compiler and tools.

rustup show	Show installed toolchains
rustup update	Update all toolchains
rustup default TC	Set the default toolchain to TC
rustup component list	List available components
rustup component add NAME	Add component NAME
rustup target list	List available compilation targets
rustup target add NAME	Add a compilation target NAME
rustup docs	Open local Rust documentation in standard web browser
rustup uninstall TC	Uninstall toolchain TC

**Components:** parts of toolchains, partly *mandatory* partly *optional*.










rustc	man	The Rust compiler and Rustdoc
cargo	man	Package manager and build tool
rustfmt	opt	Code formatter
rust-std	man	Rust standard library, each target separate
rust-docs	opt	Local copy of the Rust documentation
rust-analyzer	opt	Language server for editors and IDEs
clippy	opt	Lint tool providing extra checks for common mistakes and stylistic choices
miri	opt	Experimental Rust interpreter
rust-src	opt	Local copy of standard library source
rust-mingw	opt	Linker and platform libraries for building on the <i>x86_64-pc-windows-gnu</i> platform
llvm-tools	opt	Collection of LLVM tools. Unstable
rustc-dev	opt	Compiler as a library, only needed for development of tools that link to the compiler

**Cargo** is used to build and run Rust projects.

cargo init	Create a new binary project
cargo init -lib	Create a new library project
cargo check	Check code for errors
cargo clippy	Lint code
cargo doc	Generate project documentation
cargo run	Run the project
cargo run -bin NAME	Run a specific project binary
cargo build	Build everything in debug mode
cargo build -bin NAME	Build binary NAME in debug mode
cargo build -release	Build everything in release mode
cargo build -target NAME	Build for a specific target NAME
cargo -explain CODE	Details on compiler error CODE
cargo test	Run all tests
cargo test TEST_NAME	Run a specific test TEST_NAME
cargo test -doc	Run doctests only
cargo test -examples	Run tests for example code only
cargo bench	Run benchmarks

Primitive Data Types

Numeric Types and Boolean Types

bool		Boolean true or false
u8,i8		0..255, -128..127
u16,i16		0..65535, -32768..32767
u32,i32		0..4294967295, -2147483648..2147483647
u64,i64		0..18446744073709551615, -9223372036854775808..9223372036854775807
u128,i128		0..340282366920938463463374607431768211455 -170141183460469231731687303715884105728..170141183460469231731687303715884105727
usize, isize		Same as ptr on platform.
f16		16-bit floating point
Bad  {keycap-hash}		