# 概述

Cargo设置和读取许多环境变量，这些环境变量在代码中可以检测或覆盖的。

# Environment Variables Read For Cargo

可以覆盖以下环境变量以更改 Cargo 在系统上的行为。

## CARGO\_HOME

Cargo维护着注册表索引和从git checkouts 的crate的本地缓存。默认情况下，它们存储在$HOME/.cargo下（Windows 上为 %USERPROFILE%\.cargo），但CARGO\_HOME变量会覆盖此目录的位置。一旦 crate 被缓存，clean 命令不会将其删除。

## CARGO\_TARGET\_DIR

放置所有生成的文件的位置，其相对于当前工作目录。

## RUSTC

指定Cargo 使用的编译器。默认为rustc。

## RUSTFLAGS

要传递给 Cargo 执行的所有编译器调用的标志列表，标志以空格分隔。与cargo rustc 相比，这对于将标志传递给所有编译器实例很有用。

## RUSTC\_WRAPPER

Cargo 不是简单地运行 rustc，而是执行这个指定的包装器，将 rustc作为其命令行第一个参数传递。 用于设置构建缓存工具，例如 sccache。

## RUSTC\_WORKSPACE\_WRAPPER

Cargo 不是简单地运行 rustc，而是只为工作区成员执行这个指定的包装器，将 rustc作为其命令行第一个参数传递。 它影响文件名哈希，以便包装器产生的文件被单独缓存。

## RUSTDOC

Cargo 将执行此指定的 rustdoc 实例，而不是运行 rustdoc。

## RUSTDOCFLAGS

要传递给 Cargo 执行的所有 rustdoc 调用标志列表，标志以空格分隔。与cargo rustdoc 相比，这对于将标志传递给所有 rustdoc 实例很有用。

## CARGO\_INCREMENTAL

如果设置为 1，则 Cargo 将强制为当前编译启用增量编译，设置为 0 时将强制禁用它。如果此环境变量不存在，则将使用cargo的默认值。

## CARGO\_CACHE\_RUSTC\_INFO

如果设置为 0，则 Cargo 不会尝试缓存编译器版本信息。

## CARGO\_NAME

用于cargo new的author name。

## CARGO\_EMAIL

用于cargo new的author email。

## HTTPS\_PROXY

要使用的 HTTP 代理。https\_proxy 或 http\_proxy的环境变量同样有效。

## HTTP\_TIMEOUT

HTTP 超时（以秒为单位）。

## TERM

如果这设置为dumb，它会禁用进度条。

## BROWSER

指定执行cargo doc's' --open命令时打开的浏览器。

## RUSTFMT

执行cargo fmt命令时将执行这个指定的 rustfmt 实例，而不是原来的rustfmt。

# Environment Variables Read For Configuration

Cargo 读取以下环境变量覆盖对应配置值：

* CARGO\_ALIAS\_<name>：Command aliases, see [alias](https://doc.rust-lang.org/cargo/reference/config.html#alias).
* CARGO\_BUILD\_JOBS：Number of parallel jobs, see [build.jobs](https://doc.rust-lang.org/cargo/reference/config.html#buildjobs).
* CARGO\_BUILD\_RUSTC：The rustc executable, see [build.rustc](https://doc.rust-lang.org/cargo/reference/config.html#buildrustc).
* CARGO\_BUILD\_RUSTC\_WRAPPER：The rustc wrapper, see [build.rustc-wrapper](https://doc.rust-lang.org/cargo/reference/config.html#buildrustc-wrapper).
* CARGO\_BUILD\_RUSTC\_WORKSPACE\_WRAPPER：The rustc wrapper for workspace members, see [build.rustc-workspace-wrapper](https://doc.rust-lang.org/cargo/reference/config.html#buildrustc-workspace-wrapper).
* CARGO\_BUILD\_RUSTDOC：The rustdoc executable, see [build.rustdoc](https://doc.rust-lang.org/cargo/reference/config.html#buildrustdoc).
* CARGO\_BUILD\_TARGET：The default target platform, see [build.target](https://doc.rust-lang.org/cargo/reference/config.html#buildtarget).
* CARGO\_BUILD\_TARGET\_DIR：The default output directory, see [build.target-dir](https://doc.rust-lang.org/cargo/reference/config.html#buildtarget-dir).
* CARGO\_BUILD\_RUSTFLAGS：Extra rustc flags, see [build.rustflags](https://doc.rust-lang.org/cargo/reference/config.html#buildrustflags).
* CARGO\_BUILD\_RUSTDOCFLAGS：Extra rustdoc flags, see [build.rustdocflags](https://doc.rust-lang.org/cargo/reference/config.html#buildrustdocflags).
* CARGO\_BUILD\_INCREMENTAL：Incremental compilation, see [build.incremental](https://doc.rust-lang.org/cargo/reference/config.html#buildincremental).
* CARGO\_BUILD\_DEP\_INFO\_BASEDIR：Dep-info relative directory, see [build.dep-info-basedir](https://doc.rust-lang.org/cargo/reference/config.html#builddep-info-basedir).
* CARGO\_BUILD\_PIPELINING：Whether or not to use rustc pipelining, see [build.pipelining](https://doc.rust-lang.org/cargo/reference/config.html#buildpipelining).
* CARGO\_CARGO\_NEW\_NAME：The author name to use with [cargo new](https://doc.rust-lang.org/cargo/commands/cargo-new.html), see [cargo-new.name](https://doc.rust-lang.org/cargo/reference/config.html#cargo-newname).
* CARGO\_CARGO\_NEW\_EMAIL：The author email to use with [cargo new](https://doc.rust-lang.org/cargo/commands/cargo-new.html), see [cargo-new.email](https://doc.rust-lang.org/cargo/reference/config.html#cargo-newemail).
* CARGO\_CARGO\_NEW\_VCS：The default source control system with [cargo new](https://doc.rust-lang.org/cargo/commands/cargo-new.html), see [cargo-new.vcs](https://doc.rust-lang.org/cargo/reference/config.html#cargo-newvcs).
* CARGO\_HTTP\_DEBUG：Enables HTTP debugging, see [http.debug](https://doc.rust-lang.org/cargo/reference/config.html#httpdebug).
* CARGO\_HTTP\_PROXY：Enables HTTP proxy, see [http.proxy](https://doc.rust-lang.org/cargo/reference/config.html#httpproxy).
* CARGO\_HTTP\_TIMEOUT：The HTTP timeout, see [http.timeout](https://doc.rust-lang.org/cargo/reference/config.html#httptimeout).
* CARGO\_HTTP\_CAINFO：The TLS certificate Certificate Authority file, see [http.cainfo](https://doc.rust-lang.org/cargo/reference/config.html#httpcainfo).
* CARGO\_HTTP\_CHECK\_REVOKE：Disables TLS certificate revocation checks, see [http.check-revoke](https://doc.rust-lang.org/cargo/reference/config.html#httpcheck-revoke).
* CARGO\_HTTP\_SSL\_VERSION：The TLS version to use, see [http.ssl-version](https://doc.rust-lang.org/cargo/reference/config.html#httpssl-version).
* CARGO\_HTTP\_LOW\_SPEED\_LIMIT：The HTTP low-speed limit, see [http.low-speed-limit](https://doc.rust-lang.org/cargo/reference/config.html#httplow-speed-limit).
* CARGO\_HTTP\_MULTIPLEXING：Whether HTTP/2 multiplexing is used, see [http.multiplexing](https://doc.rust-lang.org/cargo/reference/config.html#httpmultiplexing).
* CARGO\_HTTP\_USER\_AGENT：The HTTP user-agent header, see [http.user-agent](https://doc.rust-lang.org/cargo/reference/config.html#httpuser-agent).
* CARGO\_INSTALL\_ROOT：The default directory for [cargo install](https://doc.rust-lang.org/cargo/commands/cargo-install.html), see [install.root](https://doc.rust-lang.org/cargo/reference/config.html#installroot).
* CARGO\_NET\_RETRY：Number of times to retry network errors, see [net.retry](https://doc.rust-lang.org/cargo/reference/config.html#netretry).
* CARGO\_NET\_GIT\_FETCH\_WITH\_CLI：Enables the use of the git executable to fetch, see [net.git-fetch-with-cli](https://doc.rust-lang.org/cargo/reference/config.html#netgit-fetch-with-cli).
* CARGO\_NET\_OFFLINE：Offline mode, see [net.offline](https://doc.rust-lang.org/cargo/reference/config.html#netoffline).
* CARGO\_PROFILE\_<name>\_BUILD\_OVERRIDE\_<key>：Override build script profile, see [profile.<name>.build-override](https://doc.rust-lang.org/cargo/reference/config.html#profilenamebuild-override).
* CARGO\_PROFILE\_<name>\_CODEGEN\_UNITS：Set code generation units, see [profile.<name>.codegen-units](https://doc.rust-lang.org/cargo/reference/config.html#profilenamecodegen-units).
* CARGO\_PROFILE\_<name>\_DEBUG：What kind of debug info to include, see [profile.<name>.debug](https://doc.rust-lang.org/cargo/reference/config.html#profilenamedebug).
* CARGO\_PROFILE\_<name>\_DEBUG\_ASSERTIONS：Enable/disable debug assertions, see [profile.<name>.debug-assertions](https://doc.rust-lang.org/cargo/reference/config.html#profilenamedebug-assertions).
* CARGO\_PROFILE\_<name>\_INCREMENTAL：Enable/disable incremental compilation, see [profile.<name>.incremental](https://doc.rust-lang.org/cargo/reference/config.html#profilenameincremental).
* CARGO\_PROFILE\_<name>\_LTO：Link-time optimization, see [profile.<name>.lto](https://doc.rust-lang.org/cargo/reference/config.html#profilenamelto).
* CARGO\_PROFILE\_<name>\_OVERFLOW\_CHECKS：Enable/disable overflow checks, see [profile.<name>.overflow-checks](https://doc.rust-lang.org/cargo/reference/config.html#profilenameoverflow-checks).
* CARGO\_PROFILE\_<name>\_OPT\_LEVEL：Set the optimization level, see [profile.<name>.opt-level](https://doc.rust-lang.org/cargo/reference/config.html#profilenameopt-level).
* CARGO\_PROFILE\_<name>\_PANIC：The panic strategy to use, see [profile.<name>.panic](https://doc.rust-lang.org/cargo/reference/config.html#profilenamepanic).
* CARGO\_PROFILE\_<name>\_RPATH：The rpath linking option, see [profile.<name>.rpath](https://doc.rust-lang.org/cargo/reference/config.html#profilenamerpath).
* CARGO\_REGISTRIES\_<name>\_INDEX：URL of a registry index, see [registries.<name>.index](https://doc.rust-lang.org/cargo/reference/config.html#registriesnameindex).
* CARGO\_REGISTRIES\_<name>\_TOKEN：Authentication token of a registry, see [registries.<name>.token](https://doc.rust-lang.org/cargo/reference/config.html#registriesnametoken).
* CARGO\_REGISTRY\_DEFAULT：Default registry for the --registry flag, see [registry.default](https://doc.rust-lang.org/cargo/reference/config.html#registrydefault).
* CARGO\_REGISTRY\_TOKEN：Authentication token for [crates.io](https://crates.io/), see [registry.token](https://doc.rust-lang.org/cargo/reference/config.html#registrytoken).
* CARGO\_TARGET\_<triple>\_LINKER：The linker to use, see [target.<triple>.linker](https://doc.rust-lang.org/cargo/reference/config.html#targettriplelinker).
* CARGO\_TARGET\_<triple>\_RUNNER：The executable runner, see [target.<triple>.runner](https://doc.rust-lang.org/cargo/reference/config.html#targettriplerunner).
* CARGO\_TARGET\_<triple>\_RUSTFLAGS：Extra rustc flags for a target, see [target.<triple>.rustflags](https://doc.rust-lang.org/cargo/reference/config.html#targettriplerustflags).
* CARGO\_TERM\_VERBOSE：The default terminal verbosity, see [term.verbose](https://doc.rust-lang.org/cargo/reference/config.html#termverbose).
* CARGO\_TERM\_COLOR：The default color mode, see [term.color](https://doc.rust-lang.org/cargo/reference/config.html#termcolor).
* CARGO\_TERM\_PROGRESS\_WHEN：The default progress bar showing mode, see [term.progress.when](https://doc.rust-lang.org/cargo/reference/config.html#termprogresswhen).
* CARGO\_TERM\_PROGRESS\_WIDTH：The default progress bar width, see [term.progress.width](https://doc.rust-lang.org/cargo/reference/config.html#termprogresswidth).

# Environment Variables Set For Crate

Cargo 在编译时将给crate暴露一些环境变量。可以使用以下方式在 Rust 程序中获取这些变量的值：

|  |
| --- |
| let version = env!("CARGO\_PKG\_VERSION"); |

以下是Cargo编译时会设置的环境变量：

* CARGO：执行构建的cargo二进制文件的路径。
* CARGO\_MANIFEST\_DIR：包含包的cargo.toml的目录。
* CARGO\_PKG\_VERSION：包的完整版本号。
* CARGO\_PKG\_VERSION\_MAJOR：包的主版本号。
* CARGO\_PKG\_VERSION\_MINOR：包的次版本号。
* CARGO\_PKG\_VERSION\_PATCH：包的补丁版本号。
* CARGO\_PKG\_VERSION\_PRE：包的预发布版本。
* CARGO\_PKG\_AUTHORS：cargo.toml中设置的作者列表，以冒号分隔。
* CARGO\_PKG\_NAME：包名。
* CARGO\_PKG\_DESCRIPTION：cargo.toml中设置的包描述。
* CARGO\_PKG\_HOMEPAGE：cargo.toml中设置的包主页。
* CARGO\_PKG\_REPOSITORY：cargo.toml中设置的包仓库。
* CARGO\_PKG\_LICENSE：cargo.toml中设置的包许可。
* CARGO\_PKG\_LICENSE\_FILE：cargo.toml中设置的包许可文件。
* CARGO\_CRATE\_NAME：当前正在编译的 crate 的名称。
* CARGO\_BIN\_NAME：当前正在编译的二进制文件的名称。名称不包含文件扩展名，如".exe"。
* OUT\_DIR：如果包有构建脚本，则将其设置为构建脚本的输出文件夹。
* CARGO\_BIN\_EXE\_<name>：二进制目标可执行文件的绝对路径。这仅在构建集成测试或基准测试时设置。
* CARGO\_PRIMARY\_PACKAGE：如果正在构建的包是主包，则此环境变量有值。构建依赖项时不会设置此环境变量。
* PATH：动态链接库路径，仅windows有效。
* LD\_LIBRARY\_PATH：动态链接库路径，仅unix有效。
* DYLD\_FALLBACK\_LIBRARY\_PATH：动态链接库路径，仅macos有效。

# Environment Variables Set For Build Script

Cargo 在构建脚本运行时会设置一些环境变量。可以使用以下方式在构建脚本运行时获取这些变量的值：

|  |
| --- |
| use std::env;  let out\_dir = env::var("OUT\_DIR").unwrap(); |

以下是Cargo构建脚本运行时会设置的环境变量：

* CARGO：执行构建的cargo二进制文件的路径。
* CARGO\_MANIFEST\_DIR：包含正在构建的包的cargo.toml的目录（包含构建脚本的包）。
* CARGO\_MANIFEST\_LINKS：cargo.toml链接值。
* CARGO\_MAKEFLAGS：包含 Cargo 的作业服务器实现并行化子流程所需的参数。
* CARGO\_FEATURE\_<name>：对于正在构建的包的激活特性，其中 <name> 是大写的功能名称，并将 - 转换为 \_。
* CARGO\_CFG\_<cfg>：对于正在构建的包的配置选项，此环境变量将包含配置的值，其中 <cfg> 是大写的配置名称，并将 - 转换为 \_。
* OUT\_DIR：应放置所有输出的文件夹。
* TARGET：正在编译的目标三元组。
* HOST：Rust 编译器的主机三元组。
* NUM\_JOBS：指定的顶层并行度。
* OPT\_LEVEL, DEBUG：当前正在构建的配置文件的相应变量的值。
* PROFILE：当前使用的profile，可以时debug或release。
* RUSTC, RUSTDOC：Cargo 决定使用的编译器和文档生成器，传递给构建脚本，以便它也可以使用它。
* RUSTC\_LINKER：Cargo 已解析用于当前目标的链接器二进制文件的路径。

# Environment Variables Sets For 3rd Party Subcommands

Cargo 将以下环境变量暴露给第三方子命令：

* CARGO：执行构建的cargo二进制文件的路径。