Package: cargo (via r-universe)

September 13, 2024

Title Functions to Define R Functions from Inlined Rust Code

Version 0.6.0

Description Dynamically define R functions with inlined 'Rust' code.
Help is provided to run 'Cargo'
<`https://doc.rust-lang.org/cargo/`> in a manner consistent with
CRAN policies in R packages using 'Rust'. The package is not
official, affiliated with, nor endorsed by the Rust project.

URL `https://github.com/dbdahl/cargo-framework` (repository)

BugReports `https://github.com/dbdahl/cargo-framework/issues`

License MIT + file LICENSE | Apache License 2.0

Depends R (>= 4.2.0)

Suggests roxygen2 (>= 7.2.3)

Encoding UTF-8

RoxygenNote 7.3.1

Roxygen list(markdown = TRUE)

Repository https://dbdahl.r-universe.dev

RemoteUrl https://github.com/dbdahl/cargo-framework

RemoteRef HEAD

RemoteSha d2252d046a8b8ee753b1fa1d2f8ae25d0cc3615a

Contents

`install` *Install Rust Toolchain*

Description

This function downloads the '`rustup`' installer, run it, and adds targets to compile for all the CRAN build machines.

Usage

```
install(force = FALSE)
```

Arguments

`force`	If `TRUE`, installation proceeds without asking for user confirmation.

Value

Invisibly, `TRUE` if successful and `FALSE` otherwise.

`run` *Run Cargo*

Description

This function runs Cargo (Rust's package manager) with the `...` arguments passed as command line arguments.

Usage

```
run(
  ...,
  minimum_version = ".",
  search_methods = c("cache", "convention", "path"),
  leave_no_trace = FALSE,
  environment_variables = list(),
  rustflags = NULL,
  verbose = TRUE,
  run_twice = FALSE,
  stdout = "",
  stderr = ""
)
```

Arguments

`...`	Character vector of command line arguments passed to the `cargo` command.
`minimum_version`	A character string representing the minimum version of Rust that is needed. Or a path to the root of a package (i.e., the directory containing the DESCRIPTION file), in which case the value is found from the field: `SystemRequirements: Cargo (>= XXXX)`. For the `search_methods` being `"cache"`, the shell command `rustup` is used to upgrade the Cargo installation if needed.
`search_methods`	A character vector potentially containing values `"path"`, `"convention"`, and `"cache"`. This indicates the methods to use (and their order) when searching for a suitable Cargo installation. `"path"` indicates to try to use `base::Sys.which()`. `"convention"` indicates to try to use the directories `.cargo` in the user's home directory. `"cache"` indicates to try to use the directory from the cargo package's own installation as given by the `tools::R_user_dir('cargo', 'cache')`.
`leave_no_trace`	If `TRUE`, the `CARGO_HOME` environment variable is set to a temporary directory that is subsequently deleted.
`environment_variables`	A named character vector providing environment variables which should be temporarily set while running Cargo. Note that the `CARGO_HOME` and `RUSTUP_HOME` environment variables are automatically set when using the `"cache"` search method. Also, the `CARGO_HOME` environment variable is also set when `leave_no_trace == TRUE`.
`rustflags`	A character vector from which the `CARGO_ENCODED_RUSTFLAGS` environment variables is constructed and then temporarily set. Or, if `NULL`, this environment variable is left unchanged.
`verbose`	If `TRUE`, details of the search for Cargo are shown. If `FALSE`, no details are shown. If it is a connection, then details are shown and also written to the connection.
`run_twice`	Should the cargo command be run twice? The environment variable `R_CARGO_RUN_COUNTER` is set to either `1` or `2` during each run.
`stdout`	See argument of the same name in `base::system2()`.
`stderr`	See argument of the same name in `base::system2()`.

Value

The same value and behavior as the `base::system2()` function, except a non-zero exit code will be given in Cargo is not found.

Examples

```
if (run("--version") != 0) {
  message("Cargo is not installed. Please run cargo::install() in an interactive session.")
}
```

rust_fn *Define an R Function Implemented in Rust*

Description

This function takes Rust code as a string from the last unnamed argument, takes variable names for all other unnamed arguments, compiles the Rust function, and wraps it as an R function.

Usage

```
rust_fn(
  ...,
  dependencies = character(0),
  minimum_version = "1.31.0",
  verbose = FALSE,
  cached = TRUE,
  longjmp = TRUE,
  invisible = FALSE,
  force = FALSE,
  which = NULL
)
```

Arguments

`...`	Rust code is taken as a string from the last unnamed argument, and variable names come for all other unnamed arguments. See example.
`dependencies`	A character vector of crate dependencies, e.g., `c('rand = "0.8.5"','rand_pcg = "0.3.1"')`.
`minimum_version`	A character string representing the minimum version of Rust that is needed. Or a path to the root of a package (i.e., the directory containing the DESCRIPTION file), in which case the value is found from the field: `SystemRequirements: Cargo (>= XXXX)`. For the `search_methods` being `"cache"`, the shell command `rustup` is used to upgrade the Cargo installation if needed.
`verbose`	If `TRUE`, Cargo prints compilation details. If `FALSE`, Cargo is run in quiet mode, except for the first time this function is run. If `"never"`, Cargo is always run in quiet mode. In any case, errors in code are always shown.
`cached`	Should Cargo use previously downloaded and compiled artifacts?
`longjmp`	Should the compiled function use the faster (but experimental) longjmp functionality when Rust code panics?
`invisible`	Should the compiled function return values invisibly?
`force`	If `TRUE`, write to cache directory on first usage without asking for user confirmation.

`which`	NULL or a vector of length one with name either `'tag'` or `'branch'` indicating the specific version of the roxido framework to use. If NULL, `'c(tag = "latest")'` is used. This only has an effect when downloading artifacts (e.g., when `cached = FALSE`).

Value

An R function implemented with the supplied Rust code.

Index