Package 'cleancall'

July 22, 2025

Title C Resource Cleanup via Exit Handlers

Version 0.1.3

Description Wrapper of .Call() that runs exit handlers to clean up C resources. Helps managing C (non-R) resources while using the R API.

License MIT + file LICENSE

URL https://github.com/r-lib/cleancall#readme, https://r-lib.github.io/cleancall/

BugReports https://github.com/r-lib/cleancall/issues

Depends R (>= 3.4)

Suggests covr, testthat (>= 3.0.0)

Config/Needs/website tidyverse/tidytemplate

Encoding UTF-8

RoxygenNote 7.0.2

Config/testthat/edition 3

NeedsCompilation yes

Author Lionel Henry [aut], Gábor Csárdi [aut, cre] (ORCID: <https://orcid.org/0000-0001-7098-9676>), RStudio [cph, fnd]

Maintainer Gábor Csárdi <csardi.gabor@gmail.com>

Repository CRAN

Date/Publication 2022-10-26 13:05:07 UTC

Contents

cleancall-package *cleancall: C Resource Cleanup via Exit Handlers*

Description

Wrapper of .Call() that runs exit handlers to clean up C resources. Helps managing C (non-R) resources while using the R API.

Author(s)

Maintainer: Gábor Csárdi <csardi.gabor@gmail.com> (ORCID)

Authors:

- Lionel Henry <lionel@rstudio.com>

Other contributors:

- RStudio [copyright holder]

See Also

Useful links:

- https://github.com/r-lib/cleancall#readme
- Report bugs at https://github.com/r-lib/cleancall/issues

call_with_cleanup *Call a native routine within an exit context*

Description

C functions called this way can call the `r_call_on_exit()` and/or `r_call_on_early_exit()` functions to establish exit handlers.

Usage

```
call_with_cleanup(ptr, ...)
```

Arguments

`ptr`	A native pointer object.
`...`	Arguments for the native routine.
	Handlers installed via `r_call_on_exit()` are always executed on exit. Handlers installed via `r_call_on_early_exit()` are only executed on early exit, i.e. *not* on normal termination.

C API

- `void r_call_on_exit(void (*fn)(void* data), void *data)`

  Push an exit handler to the stack. This exit handler is always executed, i.e. both on normal and early exits.

  Exit handlers are executed right after the function called from `call_with_cleanup()` exits. (Or the function used in `r_with_cleanup_context()`, if the cleanup context was established from C.)

  Exit handlers are executed in reverse order (last in is first out, LIFO). Exit handlers pushed with `r_call_on_exit()` and `r_call_on_early_exit()` share the same stack.

  Best practice is to use this function immediately after acquiring a resource, with the appropriate cleanup function for that resource.

- `void r_call_on_early_exit(void (*fn)(void* data), void *data)`

  Push an exit handler to the stack. This exit handler is only executed on early exists, *not* on normal termination.

  Exit handlers are executed right after the function called from `call_with_cleanup()` exits. (Or the function used in `r_with_cleanup_context()`, if the cleanup context was established from C.)

  Exit handlers are executed in reverse order (last in is first out, LIFO). Exit handlers pushed with `r_call_on_exit()` and `r_call_on_early_exit()` share the same stack.

  Best practice is to use this function immediately after acquiring a resource, with the appropriate cleanup function for that resource.

- `SEXP r_with_cleanup_context(SEXP (*fn)(void* data), void* data)`

  Establish a cleanup stack and call `fn` with `data`. This function can be used to establish a cleanup stack from C code.

See Also

The package README file.

Index