

Package ‘RcppSpdlog’

October 12, 2025

Type Package

Title R and C++ Interfaces to 'spdlog' C++ Header Library for Logging

Version 0.0.23

Date 2025-10-11

License GPL (>= 2)

Description The mature and widely-used C++ logging library 'spdlog' by Gabi Melman provides many desirable features. This package bundles these header files for easy use by R packages from both their R and C or C++ code. Explicit use via 'LinkingTo:' is also supported. Also see the 'spd' package which enhanced this package with a consistent R and C++ interface.

URL <https://github.com/eddelbuettel/rcppspdlog>,
<https://eddelbuettel.github.io/rcppspdlog/>,
<https://dirk.eddelbuettel.com/code/rcpp.spdlog.html>

BugReports <https://github.com/eddelbuettel/rcppspdlog/issues>

LinkingTo Rcpp

Imports Rcpp

Suggests simplermardown

VignetteBuilder simplermardown

RoxygenNote 6.0.1

NeedsCompilation yes

Author Dirk Eddelbuettel [aut, cre] (ORCID:
<<https://orcid.org/0000-0001-6419-907X>>),
Gabi Melman [aut] (Author of spdlog),
Victor Zverovic [aut] (Author of fmt)

Maintainer Dirk Eddelbuettel <edd@debian.org>

Repository CRAN

Date/Publication 2025-10-12 05:10:02 UTC

Contents

RcppSpdlog-package	2
exampleRsink	3
formatter	4
get_stopwatch	4
log_setup	6
setLogLevel	7
Index	9

RcppSpdlog-package	<i>R and C++ Interfaces to 'spdlog' C++ Header Library for Logging</i>
--------------------	--

Description

The mature and widely-used C++ logging library 'spdlog' by Gabi Melman provides many desirable features. This package bundles these header files for easy use by R packages from both their R and C or C++ code. Explicit use via 'LinkingTo:' is also supported. Also see the 'spdl' package which enhanced this package with a consistent R and C++ interface.

Details

The DESCRIPTION file:

Package: RcppSpdlog
Type: Package
Title: R and C++ Interfaces to 'spdlog' C++ Header Library for Logging
Version: 0.0.23
Date: 2025-10-11
License: GPL (>= 2)
Authors@R: c(person("Dirk", "Eddelbuettel", role = c("aut", "cre"), email = "edd@debian.org", comment = c(ORCID =
Description: The mature and widely-used C++ logging library 'spdlog' by Gabi Melman provides many desirable featur
URL: https://github.com/eddelbuettel/rcppspdlog, https://eddelbuettel.github.io/rcppspdlog/, https://dirk.eddelbue
BugReports: https://github.com/eddelbuettel/rcppspdlog/issues
LinkingTo: Rcpp
Imports: Rcpp
Suggests: simplrmarkdown
VignetteBuilder: simplrmarkdown
RoxygenNote: 6.0.1
Author: Dirk Eddelbuettel [aut, cre] (ORCID: <https://orcid.org/0000-0001-6419-907X>), Gabi Melman [aut] (Aut
Maintainer: Dirk Eddelbuettel <edd@debian.org>

Index of help topics:

RcppSpdlog-package	R and C++ Interfaces to 'spdlog' C++ Header Library for Logging
exampleRsink	spdlog Example using a sink for R
formatter	Simple Pass-Through Formatter to 'fmt::format()'
get_stopwatch	R Accessor Functions for spdlog Stopwatch
log_setup	R Accessor Functions for spdlog Logger
setLogLevel	spdlog Logging Level Setter

This section should provide a more detailed overview of how to use the package, including the most important functions.

Author(s)

Dirk Eddelbuettel [aut, cre] (ORCID: <<https://orcid.org/0000-0001-6419-907X>>), Gabi Melman [aut] (Author of spdlog), Victor Zverovic [aut] (Author of fmt)

Maintainer: Dirk Eddelbuettel <edd@debian.org>

exampleRsink	<i>spdlog Example using a sink for R</i>
--------------	--

Description

A simple example invoking a derived R/Rcpp logger. Also demonstrates the stopwatch feature. For more features see the 'spdlog' documnetation.

Usage

```
exampleRsink()
```

Details

Note that this no longer triggers R warnings thanks to excellent help by Gabi Melman.

Value

None

Examples

```
exampleRsink()
```

 formatter

Simple Pass-Through Formatter to `fmt::format()`

Description

The C-level interface of R does not make it easy to pass ... arguments. This helper function assumes it has already been called with `format()` on each argument (as a wrapper can do) so it just spreads out the class to `fmt::format{}` which, being C++, uses variadic templates to receive the arguments. The main motivation for this function is to be able to format strings as used by the ‘`fmtlib::fmt`’ library included in ‘`spdlog`’ to write similar debug strings in both R and C++. This function permits R calls with multiple arguments of different types which (by being formatted on the R side) are handled as strings (whereas C++ logging has access to the templating logic).

Usage

```
formatter(s, v)
```

Arguments

<code>s</code>	A character variable with a format string for ‘ <code>fmtlib::fmt</code> ’
<code>v</code>	A character vector with the logging string arguments.

Value

A single (formatted) string

See Also

<https://github.com/fmtlib/fmt>

 get_stopwatch

R Accessor Functions for `spdlog` Stopwatch

Description

A set of functions provides access to the `spdlog` stopwatch facility. As `stopwatch` object is a simple container around a C++ `std::chrono` object which (essentially) reports elapsed-time since creation. The object is exported to R via an external pointer permitting use from both R and C++.

Usage

```
get_stopwatch()

elapsed_stopwatch(sw)

format_stopwatch(sw)

## S3 method for class 'stopwatch'
print(x, ...)

## S3 method for class 'stopwatch'
format(x, ...)
```

Arguments

<code>sw</code>	An S3 object of type stopwatch.
<code>x</code>	An S3 object of type stopwatch.
<code>...</code>	Dotted argument required by generic, unused here.

Details

Several functions are provided:

`get_stopwatch` Returns a stopwatch object (as an S3 object).
`elapsed_stopwatch` Returns elapsed time for stopwatch in seconds.
`format_stopwatch` Returns elapsed time for stopwatch as character variable.

The stopwatch object has `print` and `format` methods.

Value

The desired object is returned: respectively, a stopwatch object as an external pointer in an S3 class, the elapsed time in seconds as a double, or formatted as a character variable.

Examples

```
w <- get_stopwatch()
Sys.sleep(0.2)
elapsed_stopwatch(w)
format_stopwatch(w)
```

log_setup	<i>R Accessor Functions for spdlog Logger</i>
-----------	---

Description

Several R-level functions can access the spdlog logging facilities. As spdlog is a C++-level logging library, these are R function permit concurrent logging from both R and C++.

Usage

```
log_setup(name = "default", level = "warn")

log_init(level = "warn")

log_filesetup(filename, name = "default", level = "warn")

log_drop(name)

log_set_pattern(s)

log_set_level(s)

log_trace(s)

log_debug(s)

log_info(s)

log_warn(s)

log_error(s)

log_critical(s)
```

Arguments

name	A character variable with the logging instance name, default value is ‘default’.
level	A character variable with the default logging level, default value is ‘warn’.
filename	A character variable with the logging filename if a file-based logger is instantiated.
s	A character variable with the logging pattern, level or message.

Details

Several functions are provided:

log_setup Initializes a logger (which becomes the default logger).

`log_filesetup` Initializes a file-based logger (which becomes the default).

`log_drop` Removes logger (which in general should not be needed).

`log_set_pattern` Changes the default logging message pattern.

`log_set_level` Sets the logging level threshold.

`log_trace` Logs a trace-level message.

`log_debug` Logs a debug-level message.

`log_info` Logs a info-level message.

`log_warn` Logs a warn-level message.

`log_error` Logs a error-level message.

`log_critical` Logs a critical-level message.

Supported logging levels are, in order of increasing threshold values, 'trace', 'debug', 'info', 'warn', 'error', and 'critical'. A message issued below the current threshold is not displayed whereas a message at or above the current threshold is displayed. The default level is 'warn'.

Value

Nothing is returned from these functions as they are invoked for their side-effects.

See Also

The logging pattern format is described in at the repo in the page <https://github.com/gabime/spdlog/wiki/3.-Custom-formatting>.

Examples

```
log_setup("demo") # at default level 'warn'
log_info("this message is NOT seen")
log_set_level("debug")
log_info("this message is seen")
log_warn("as is this message")
```

setLogLevel

spdlog Logging Lever Setter

Description

A helper function to turn a logging level given as string into the current logging level

Usage

```
setLogLevel(name)
```

Arguments

name	A string with the logging level. Value understood are, in decreasing verbosity 'trace', 'debug', 'info', 'warning', 'error', 'critical', and 'off'. Unrecognised names are equivalent to 'off'.
------	---

Value

Nothing is returned.

Index

* **package**

RcppSpdlog-package, [2](#)

elapsed_stopwatch (get_stopwatch), [4](#)

exampleRsink, [3](#)

format.stopwatch (get_stopwatch), [4](#)

format_stopwatch (get_stopwatch), [4](#)

formatter, [4](#)

get_stopwatch, [4](#)

log_critical (log_setup), [6](#)

log_debug (log_setup), [6](#)

log_drop (log_setup), [6](#)

log_error (log_setup), [6](#)

log_filesetup (log_setup), [6](#)

log_info (log_setup), [6](#)

log_init (log_setup), [6](#)

log_set_level (log_setup), [6](#)

log_set_pattern (log_setup), [6](#)

log_setup, [6](#)

log_trace (log_setup), [6](#)

log_warn (log_setup), [6](#)

print.stopwatch (get_stopwatch), [4](#)

RcppSpdlog (RcppSpdlog-package), [2](#)

RcppSpdlog-package, [2](#)

setLogLevel, [7](#)