

# Package ‘luzlogr’

July 22, 2025

**Type** Package

**Title** Lightweight Logging for R Scripts

**Version** 0.2.1

**Description** Provides flexible but lightweight logging facilities for R scripts.  
Supports priority levels for logs and messages, flagging messages, capturing script output, switching logs, and logging to files or connections.

**Imports** assertthat (>= 0.2)

**Depends** R (>= 3.0)

**License** MIT + file LICENSE

**Encoding** UTF-8

**Suggests** testthat, knitr, rmarkdown

**VignetteBuilder** knitr

**RoxygenNote** 7.3.1

**NeedsCompilation** no

**Author** Ben Bond-Lamberty [cre, aut] (ORCID:  
<<https://orcid.org/0000-0001-9525-4633>>)

**Maintainer** Ben Bond-Lamberty <bondlamberty@pnnl.gov>

**Repository** CRAN

**Date/Publication** 2024-02-27 06:00:02 UTC

## Contents

closelog . . . . .	2
openlog . . . . .	3
printlog . . . . .	4
<b>Index</b>	<b>6</b>

closelog

*Close current logfile*

---

**Description**

Close current logfile

**Usage**

```
closelog(sessionInfo = TRUE)
```

**Arguments**

sessionInfo      Append [sessionInfo](#) output? (logical, optional)

**Details**

Close current logfile. The number of flagged messages is returned, invisibly. Note that if `options(luzlogr.close_on_error = TRUE)` is set, then if an error occurs, all log files will be automatically closed. This behavior is not currently enabled by default.

Logs are stored on a stack, and so when one is closed, logging output returns to the previous log (if any).

**Value**

Number of flagged messages (numeric).

**Note**

If the log was being written to a [connection](#), `closelog` will return the connection to its pre-logging state, whether open or closed.

**See Also**

[openlog](#) [printlog](#)

**Examples**

```
logfile1 <- openlog("A.log")
printlog("message to A", flag = TRUE)
logfile2 <- openlog("B.log")
printlog("message to B")
flagcountB <- closelog()
flagcountA <- closelog(sessionInfo = FALSE)
file.remove(logfile1, logfile2)
```

---

openlog	<i>Open a new logfile</i>
---------	---------------------------

---

## Description

Open a new logfile

## Usage

```
openlog(file, loglevel = -Inf, append = FALSE, sink = FALSE)
```

## Arguments

file	Name of logfile (character or writeable <a href="#">connection</a> )
loglevel	Minimum priority level (numeric, optional)
append	Append to logfile? (logical, optional)
sink	Send all console output to logfile? (logical, optional)

## Details

Open a new logfile. Messages will only appear in the logfile if their level exceeds the log's loglevel; this allows you to easily change the amount of detail being logged.

Re-opening a logfile will erase the previous output unless append is TRUE. Opening a new logfile when one is already open will temporarily switch logging to that new file.

If sink is TRUE, all screen output will be captured (via [sink](#)).

## Value

Invisible fully-qualified name of log file.

## See Also

[printlog](#) [closelog](#)

## Examples

```
logfile <- openlog("test.log")
printlog("message")
closelog()
readLines(logfile)
file.remove(logfile)
```

printlog

*Log a message*

---

**Description**

Log a message

**Usage**

```
printlog(..., level = 0, ts = TRUE, cr = TRUE, flag = FALSE, flush = FALSE)
```

```
flaglog(...)
```

**Arguments**

...	Expressions to be printed to the log
level	Priority level (numeric, optional)
ts	Print preceding timestamp? (logical, optional)
cr	Print trailing newline? (logical, optional)
flag	Flag this message (e.g. error or warning) (logical, optional)
flush	Immediately flush output to file (logical, optional)

**Details**

Logs a message, which consists of zero or more printable objects. Simple objects (numeric and character) are printed together on a single line, whereas complex objects (data frames, etc) start on a new line by themselves.

If the current log was opened with `sink = TRUE`, messages are printed to the screen, otherwise not. Messages can be flagged; `flaglog` assumes that the message is to be flagged, whereas `printlog` does not.

Messages will only appear in the logfile if their `level` exceeds the log's `loglevel`; this allows you to easily change the amount of detail being logged.

**Value**

Invisible success (TRUE) or failure (FALSE).

**Note**

A message's preceding timestamp and following carriage return can be suppressed using the `ts` and `cr` parameters.

**See Also**

[openlog](#) [closelog](#)

**Examples**

```
logfile <- openlog("test.log")
printlog("message")
printlog(1, "plus", 1, "equals", 1 + 1)
closelog()
readLines(logfile)
file.remove(logfile)

logfile <- openlog("test", loglevel = 1)
printlog("This message will not appear", level = 0)
printlog("This message will appear", level = 1)
closelog()
readLines(logfile)
file.remove(logfile)
```

# Index

`closeLog`, [2](#), [3](#), [4](#)

`connection`, [2](#), [3](#)

`flagLog (printLog)`, [4](#)

`openLog`, [2](#), [3](#), [4](#)

`printLog`, [2](#), [3](#), [4](#)

`sessionInfo`, [2](#)

`sink`, [3](#)