

Package ‘brio’

January 20, 2021

Title Basic R Input Output

Version 1.1.1

Description Functions to handle basic input output, these functions always read and write UTF-8 (8-bit Unicode Transformation Format) files and provide more explicit control over line endings.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Suggests testthat (>= 2.1.0), covr

URL <https://github.com/r-lib/brio>

BugReports <https://github.com/r-lib/brio/issues>

NeedsCompilation yes

Author Jim Hester [aut, cre] (<<https://orcid.org/0000-0002-2739-7082>>),
RStudio [cph, fnd]

Maintainer Jim Hester <jim.hester@rstudio.com>

Repository CRAN

Date/Publication 2021-01-20 18:40:02 UTC

R topics documented:

file_line_endings	2
readLines	2
read_file	3
read_lines	4
writeLines	5
write_file	6
write_lines	6

Index	8
--------------	----------

file_line_endings *Retrieve the type of line endings used by a file*

Description

Retrieve the type of line endings used by a file

Usage

```
file_line_endings(path)
```

Arguments

path A character string of the path to the file to read.

Value

The line endings used, one of

- `'\n'` - if the file uses Unix line endings
- `'\r\n'` - if the file uses Windows line endings
- `NA` - if it cannot be determined

Examples

```
tf1 <- tempfile()
tf2 <- tempfile()
write_lines("foo", tf1, eol = "\n")
write_lines("bar", tf2, eol = "\r\n")

file_line_endings(tf1)
file_line_endings(tf2)

unlink(c(tf1, tf2))
```

readLines *Read text lines from a file*

Description

This is a drop in replacement for `base::readLines()` with restricted functionality. Compared to `base::readLines()` it:

- Only works with file paths, not connections.
- Assumes the files are always UTF-8 encoded.
- Does not warn or skip embedded nulls, they will likely crash R.
- Does not warn if the file is missing the end of line character.
- The arguments `ok`, `warn`, `encoding` and `skipNul` are ignored, with a warning.

Usage

```
readLines(con, n = -1, ok, warn, encoding, skipNul)
```

Arguments

con	A character string of the path to a file. Throws an error if a connection object is passed.
n	integer. The number of lines to read. A negative number means read all the lines in the file.
ok	Ignored, with a warning.
warn	Ignored, with a warning.
encoding	Ignored, with a warning.
skipNul	Ignored, with a warning.

Value

A UTF-8 encoded character vector of the lines in the file.

See Also

[writeLines\(\)](#)

Examples

```
authors_file <- file.path(R.home("doc"), "AUTHORS")
data <- readLines(authors_file)

# Trying to use connections throws an error
con <- file(authors_file)
try(readLines(con))
close(con)

# Trying to use unsupported args throws a warning
data <- readLines(authors_file, encoding = "UTF-16")
```

read_file

Read an entire file

Description

read_file() reads an entire file into a single character vector. read_file_raw() reads an entire file into a raw vector.

Usage

```
read_file(path)
```

```
read_file_raw(path)
```

Arguments

path A character string of the path to the file to read.

Details

read_file() assumes the file has a UTF-8 encoding.

Value

- `read_file()`: A length 1 character vector.
- `read_file_raw()`: A raw vector.

Examples

```
authors_file <- file.path(R.home("doc"), "AUTHORS")
data <- read_file(authors_file)
data_raw <- read_file_raw(authors_file)
identical(data, rawToChar(data_raw))
```

read_lines	<i>Read text lines from a file</i>
------------	------------------------------------

Description

The file is assumed to be UTF-8 and the resulting text has its encoding set as such.

Usage

```
read_lines(path, n = -1)
```

Arguments

path A character string of the path to the file to read.
n integer. The number of lines to read. A negative number means read all the lines in the file.

Details

Both `'\r\n'` and `'\n'` are treated as a newline.

Value

A UTF-8 encoded character vector of the lines in the file.

Examples

```
authors_file <- file.path(R.home("doc"), "AUTHORS")
data <- read_lines(authors_file)
```

writeLines	<i>Write lines to a file</i>
------------	------------------------------

Description

This is a drop in replacement for `base::writeLines()` with restricted functionality. Compared to `base::writeLines()` it:

- Only works with file paths, not connections.
- Uses `enc2utf8()` to convert `text()` to UTF-8 before writing.
- Uses `sep` unconditionally as the line ending, regardless of platform.
- The `useBytes` argument is ignored, with a warning.

Usage

```
writeLines(text, con, sep = "\n", useBytes)
```

Arguments

<code>text</code>	A character vector to write
<code>con</code>	A character string of the path to a file. Throws an error if a connection object is passed.
<code>sep</code>	The end of line characters to use between lines.
<code>useBytes</code>	Ignored, with a warning.

Value

The UTF-8 encoded input text (invisibly).

See Also

[readLines\(\)](#)

Examples

```
tf <- tempfile()

writeLines(rownames(mtcars), tf)

# Trying to use connections throws an error
con <- file(tf)
try(writeLines(con))
close(con)

# Trying to use unsupported args throws a warning
writeLines(rownames(mtcars), tf, useBytes = TRUE)

unlink(tf)
```

write_file	<i>Write data to a file</i>
------------	-----------------------------

Description

This function differs from `write_lines()` in that it writes the data in `text` directly, without any checking or adding any newlines.

Usage

```
write_file(text, path)
```

Arguments

<code>text</code>	A character vector of length 1 with data to write.
<code>path</code>	A character string giving the file path to write to.

Value

The UTF-8 encoded input text (invisibly).

Examples

```
tf <- tempfile()
write_file("some data\n", tf)
unlink(tf)
```

write_lines	<i>Write lines to a file</i>
-------------	------------------------------

Description

The text is converted to UTF-8 encoding before writing.

Usage

```
write_lines(text, path, eol = "\n")
```

Arguments

<code>text</code>	A character vector to write
<code>path</code>	A character string giving the file path to write to.
<code>eol</code>	The end of line characters to use between lines.

Details

The files are opened in binary mode, so they always use exactly the string given in `eol` as the line separator.

To write a file with windows line endings use `write_lines(eol = "\r\n")`

Value

The UTF-8 encoded input text (invisibly).

Examples

```
tf <- tempfile()

write_lines(rownames(mtcars), tf)

# Write with Windows style line endings
write_lines(rownames(mtcars), tf, eol = "\r\n")

unlink(tf)
```

Index

`base::readLines()`, 2
`base::writeLines()`, 5

`enc2utf8()`, 5

`file_line_endings`, 2

`read_file`, 3
`read_file()`, 4
`read_file_raw(read_file)`, 3
`read_file_raw()`, 4
`read_lines`, 4
`readLines`, 2
`readLines()`, 5

`write_file`, 6
`write_lines`, 6
`write_lines()`, 6
`writeLines`, 5
`writeLines()`, 3