Package 'enc'

June 24, 2019

Title Portable Tools for 'UTF-8' Character Data

Date 2019-06-24

Version 0.2.1

Description Implements an S3 class for storing 'UTF-8' strings, based on regular character vectors.
Also contains routines to portably read and write 'UTF-8' encoded text files,
to convert all strings in an object to 'UTF-8',
and to create character vectors with various encodings.

Depends R (>= 3.1)

Imports methods

Suggests digest, pillar, readr, rlang, testthat, withr

License GPL-3

Encoding UTF-8

LazyData true

BugReports https://github.com/krlmlr/enc/issues

URL https://github.com/krlmlr/enc

RoxygenNote 6.1.1

NeedsCompilation yes

Author Kirill Müller [aut, cre]

Maintainer Kirill Müller <krlmlr+r@mailbox.org>

Repository CRAN

Date/Publication 2019-06-24 09:10:11 UTC

R topics documented:

`enc-package` *enc: Portable Tools for 'UTF-8' Character Data*

Description

Implements an S3 class for storing 'UTF-8' strings, based on regular character vectors. Also contains routines to portably read and write 'UTF-8' encoded text files, to convert all strings in an object to 'UTF-8', and to create character vectors with various encodings.

Author(s)

Maintainer: Kirill Müller <krlmlr+r@mailbox.org>

See Also

Useful links:

- https://github.com/krlmlr/enc
- Report bugs at https://github.com/krlmlr/enc/issues

`encoding` *Encoding information*

Description

The `encoding` function returns `"ASCII"` if the entire value consists of ASCII symbols only, and works identically to `Encoding()` otherwise. The `all_utf8` function is an efficient variant of `all(encoding(x) %in% c("ASCII", "UTF-8"))`.

Usage

```
encoding(x)

all_utf8(x)
```

Arguments

`x`	A character vector.

Examples

```
encoding("a")
encoding("\u00fc")
all_utf8(enc2utf8(c("a", "\u00fc")))

# Platform-dependent:
all_utf8(enc2native(c("a", "\u00fc")))
```

read_lines_enc *Reads from a text file*

Description

This function is a drop-in replacement for `readLines()` from disk files. It always returns text in the UTF-8 encoding and never warns on missing EOL on the last line.

`try_read_lines_enc()` is a variant that returns an empty character vector on error, with a warning.

Usage

```
read_lines_enc(path, file_encoding = "UTF-8", n = -1L, ok = TRUE,
  skipNul = FALSE)

try_read_lines_enc(path, file_encoding = "UTF-8", n = -1L, ok = TRUE,
  skipNul = FALSE)
```

Arguments

`path`	Path to the file
`file_encoding`	The encoding to assume for the input file.
`n`	integer. The (maximal) number of lines to read. Negative values indicate that one should read up to the end of input on the connection.
`ok`	logical. Is it OK to reach the end of the connection before `n > 0` lines are read? If not, an error will be generated.
`skipNul`	logical: should nuls be skipped?

See Also

`readr::read_lines()` for a faster alternative.

Other file functions: `transform_lines_enc`, `write_lines_enc`

to_encoding *Deep conversion to an encoding*

Description

Converts all characters directly or indirectly contained in an object to a specific encoding. This works even if the encoding is different in the elements of a character vector.

Usage

```
to_encoding(x, ...)

to_utf8(x, ...)

to_native(x, ...)

to_latin1(x, ...)

to_alien(x, ...)

## S3 method for class 'character'
to_encoding(x, ..., converter)
```

Arguments

x	A character vector.
...	passed on to methods
converter	A function that accepts a character value as first argument and returns a (possibly classed) character with the desired encoding

Details

`to_utf8` converts to UTF-8, using the `utf8()` class where possible. Implemented as `to_encoding(x, as_utf8)`

`to_native` converts to the native encoding. Implemented as `to_encoding(x, enc2native)` on Windows and as `to_encoding(x, as_utf8)` on Linux and OS X

`to_latin1` converts to the latin-1 encoding

`to_alien` converts to the "other" encoding, i.e., UTF-8 on Windows and latin-1 on Linux and OS X.

See Also

- `rlang::as_utf8_character()` and `iconv()` for different ways to convert character vectors to Unicode

Examples

```
to_utf8(letters)
to_utf8(iris)
class(levels(to_utf8(iris)$Species))
```

transform_lines_enc *Transform a text file*

Description

Reads a file from disk, applies a function on the contents, and optionally writes the file back if different. The line ending separator of the input file is used if it can be read and contains at least one, otherwise `native_eol()` is used.

Usage

```
transform_lines_enc(path, fun, file_encoding = "UTF-8", ok = TRUE,
  skipNul = FALSE, write_back = TRUE, verbose = interactive())
```

Arguments

`path`	A vector of file paths.
`fun`	A function that returns a character vector.
`file_encoding`	The encoding to assume for the input file.
`ok`	logical. Is it OK to reach the end of the connection before `n > 0` lines are read? If not, an error will be generated.
`skipNul`	logical: should nuls be skipped?
`write_back`	Should the results of the transformation be written back to the file?
`verbose`	Should the function show a message with a list of changed files?

Value

A named logical vector of the same length as `path` that indicates if a file has changed (`TRUE` or `FALSE`), or if an error occurred (`NA`)

See Also

Other file functions: `read_lines_enc`, `write_lines_enc`

utf8 *A simple class for storing UTF-8 strings*

Description

The values are stored as a `character()` vector. On construction, the `enc2utf8()` function is called on the input. Subsetting and concatenation operations on an object of this class return an object of this class again. Calls to `Encoding<-()` are not intercepted.

Usage

```
utf8(x = character())

is_utf8(x)

as_utf8(x, ...)

## Default S3 method:
as_utf8(x, ...)

## S3 method for class 'NULL'
as_utf8(x, ...)

## S3 method for class 'character'
as_utf8(x, ...)

## S3 method for class 'utf8'
as_utf8(x, ...)

## S3 method for class 'utf8'
as.character(x, ...)

## S3 method for class 'utf8'
as.data.frame(x, row.names = NULL, optional = FALSE,
  ..., nm = paste(deparse(substitute(x), width.cutoff = 500L), collapse =
  " "))

## S3 method for class 'utf8'
format(x, ...)

## S3 method for class 'utf8'
print(x, ...)
```

Arguments

x	A vector
...	Arguments passed on to further methods.
row.names	NULL or a character vector giving the row names for the data frame. Missing values are not allowed.
optional	logical. If TRUE, setting row names and converting column names (to syntactic names: see make.names) is optional. Note that all of R's **base** package as.data.frame() methods use optional only for column names treatment, basically with the meaning of data.frame(*, check.names = !optional). See also the make.names argument of the matrix method.
nm	Name of column in new data frame

Examples

```
utf8(letters)
utf8("ä")
utf8(iconv("ä", to = "latin1"))
```

`write_lines_enc` *Writes to a text file*

Description

This function is a drop-in replacement for `writeLines()` from disk files. It always expects text in the UTF-8 encoding, and by default writes in the UTF-8 encoding with Unix line separators.

Usage

```
write_lines_enc(text, path, file_encoding = "UTF-8", sep = "\n")
```

Arguments

`text`	A character vector
`path`	Path to the file
`file_encoding`	The encoding for the output file.
`sep`	character string. A string to be written to the connection after each line of text.

See Also

`readr::write_lines()` for a faster alternative.

Other file functions: `read_lines_enc`, `transform_lines_enc`

Index