Package 'ctypesio'

July 28, 2025

Type Package

Title Read and Write Standard 'C' Types from Files, Connections and Raw Vectors

Version 0.1.3

Maintainer Mike Cheng <mikefc@coolbutuseless.com>

Description Interacting with binary files can be difficult because R's types are a subset of what is generally supported by 'C'. This package provides a suite of functions for reading and writing binary data (with files, connections, and raw vectors) using 'C' type descriptions. These functions convert data between 'C' types and R types while checking for values outside the type limits, 'NA' values, etc.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Suggests knitr, rmarkdown, testthat (>= 3.0.0), jpeg

Config/testthat/edition 3

VignetteBuilder knitr

URL https://github.com/coolbutuseless/ctypesio

BugReports https://github.com/coolbutuseless/ctypesio/issues

Depends R (>= 4.1.0)

NeedsCompilation yes

Author Mike Cheng [aut, cre, cph],
Anne Fu [ctb] (Better UTF-8 support, ORCID:
<https://orcid.org/0000-0002-9025-6071>)

Repository CRAN

Date/Publication 2025-07-28 09:20:02 UTC

Contents

`aperm_array_to_vector` *Permute an R array to a linear vector of data*

Description

Permute an R array to a linear vector of data

Usage

```
aperm_array_to_vector(x, dst, flipy = FALSE)
```

Arguments

`x`	array
`dst`	Specification of destination dimensions in the order of presentation in the source data. Character vector which contains 3 strings: 'planes', 'rows', 'cols'. The order of these strings determines the order of output in the linear data. Currently, "planes" must always be the final element.
`flipy`	flip the array vertically. Default: FALSE

Value

vector

See Also

Other data permutation functions: aperm_vector_to_array(), flip_endian()

Examples

```
# create a small RGBA array in R with each
# plane of the array holding a different colour channel
  arr <- array(c(paste0('r', 1:6),
                 paste0('g', 1:6),
                 paste0('b', 1:6),
                 paste0('a', 1:6)), c(2, 3, 4))
arr

# A very common C ordering is packaged RGBA data in column major format
# i.e. Iterate over: planes, then columns, then rows
# i.e.
#    start at first element
#    (plane1, plane2, plane3, plane4)
#    go to next column
#    (plane1, plane2, plane3, plane4)
#    go to next column
#    ...
#    when last column is done
#    do to next row

# Convert to packed RGBA in column-major format
vec <- aperm_array_to_vector(arr, dst = c('planes', 'cols', 'rows'))
vec

# To convert column-major packed RGBA to an R array, use the same ordering
# for the dimensions, but also need to specify length along each dimension
aperm_vector_to_array(vec, src = c(planes = 4, cols = 3, rows = 2))
```

---

aperm_vector_to_array *Permute a linear vector of data into an R array*

Description

Permute a linear vector of data into an R array

Usage

```
aperm_vector_to_array(x, src, flipy = FALSE, simplify_matrix = TRUE)
```

Arguments

x	vector

src	Specification of source dimensions in the order of presentation in the source data. This must a named integer vector with the names "planes", "rows", "cols" (and their corresponding sizes) in the order in which they occur in the data. The first named element must always be "planes". Use `planes = 1` to indicate that this is matrix data.
flipy	flip the array vertically. Default: FALSE
simplify_matrix	If the resulting array only has a single plane, should this be simplified to a matrix? Default: TRUE

Value

array or matrix

See Also

Other data permutation functions: `aperm_array_to_vector()`, `flip_endian()`

Examples

```
# Convert a vector of packed RGB data to an array with 3 planes
x <- c(
  'r0', 'g0', 'b0',   'r1', 'g1', 'b1',   'r2', 'g2', 'b2',
  'r3', 'g3', 'b3',   'r4', 'g4', 'b4',   'r5', 'g5', 'b5'
)
aperm_vector_to_array(x, src = c(planes = 3, cols = 3, rows = 2))
```

---

flip_endian *Flip the endianness of elements in a vector*

Description

This will create a new vector with the values reversed within the given block size. This can be used for changing the endianness of a set of values

Usage

```
flip_endian(x, size)
```

Arguments

x	vector. Usually a raw vector, but can be any type
size	block size. Usually a power of 2.

Value

A vector of the same type as the initial vector with the values within each block reversed.

See Also

Other data permutation functions: aperm_array_to_vector(), aperm_vector_to_array()

Examples

```
vec <- c(1, 2, 3, 4)
flip_endian(vec, 1)   # should give: c(1, 2, 3, 4)
flip_endian(vec, 2)   # should give: c(2, 1, 4, 3)
flip_endian(vec, 4)   # should give: c(4, 3, 2, 1)
```

---

fprintf *Print formatted strings to a connection*

Description

fprintf_raw() writes the text without a nul-terminator. fprintf() writes a nul-terminator

Usage

```
fprintf(con, fmt, ..., sep = "\n", useBytes = FALSE)

fprintf_raw(con, fmt, ..., sep = "\n", useBytes = FALSE)
```

Arguments

con	Connection object or raw vector. When con is a raw vector, new data will be *appended* to the vector and returned. Connection objects can be created with file(), url(), rawConnection() or any of the other many connection creation functions.
fmt	a character vector of format strings. See sprintf()
...	values to be passed in to fmt. See sprintf()
sep	If there are multiple strings to be printed, this separated will be written after each one.
useBytes	See writeLines()

Value

If con is a connection then this connection is returned invisibly. If con is a raw vector then new data is appended to this vector

See Also

Other data output functions: write_f64(), write_hex(), write_raw(), write_uint8(), write_utf8()

Examples

```
con <- rawConnection(raw(), "wb")
fprintf(con, "%i,%6.2f", 1, 3.14159)
close(con)
```

`read_f64` *Read floating point values from a connection*

Description

Read floating point numbers into a standard R vector of doubles

Usage

```
read_f64(con, n = 1, endian = NULL)

read_f32(con, n = 1, endian = NULL)

read_f16(con, n = 1, endian = NULL)

read_bfloat(con, n = 1, endian = NULL)

read_dbl(con, n = 1, endian = NULL)

read_float(con, n = 1, endian = NULL)

read_half(con, n = 1, endian = NULL)
```

Arguments

`con`	Connection object or raw vector. Connection objects can be created with `file()`, `url()`, `rawConnection()` or any of the other many connection creation functions.
`n`	Number of elements to read. Default: 1
`endian`	Ordering of bytes within the file when reading multi-byte values. Possible values: 'big' or 'little'. Default: NULL indicates that endian option should be retrieved from the connection object if possible (where the user has used `set_endian()`) or otherwise will be set to `"little"`

Details

double precision 8 byte floating point numbers. `read_f64()` also available as `read_dbl()`

single precision 4 byte floating point numbers. `read_f32()` also available as `read_float()`

half precision 2 byte floating point numbers. `read_f16()` also available as `read_half()`. Consists of 1 sign bit, 5 bits for exponent and 10 bits for fraction.

bfloat 2 byte floating point numbers in the bfloat format `read_bfloat()`. Consits of 1 sign bit, 8 bits fo exponent and 7 bits for fraction.

Value

vector of double precision floating point numbers

See Also

Other data input functions: `read_hex()`, `read_raw()`, `read_str()`, `read_uint8()`, `scan_dbl()`

Examples

```
# Raw vector with 16 bytes (128 bits) of dummy data
data <- as.raw(1:16)
con <- rawConnection(data, 'rb')
read_f64(con, n = 1) # Read a 64-bit double-precision number
read_f16(con, n = 4) # Read 4 x 16-bit half-precision number
close(con)
```

---

read_hex *Read bytes as hexadecimal strings*

Description

Read bytes as hexadecimal strings

Usage

```
read_hex(con, n = 1, size = 1, endian = NULL)
```

Arguments

con	Connection object or raw vector. Connection objects can be created with `file()`, `url()`, `rawConnection()` or any of the other many connection creation functions.
n	Number of hexadecimal strings to read. Default: 1
size	size in bytes of each string. Default: 1
endian	Ordering of bytes within the file when reading multi-byte values. Possible values: 'big' or 'little'. Default: NULL indicates that endian option should be retrieved from the connection object if possible (where the user has used `set_endian()`) or otherwise will be set to `"little"`

Value

vector of hexadecimal character strings

See Also

Other data input functions: `read_f64()`, `read_raw()`, `read_str()`, `read_uint8()`, `scan_dbl()`

Examples

```
con <- rawConnection(as.raw(1:4))
read_hex(con, n = 4, size = 1)
close(con)

con <- rawConnection(as.raw(1:4))
read_hex(con, n = 1, size = 4)
close(con)

con <- rawConnection(as.raw(1:4))
read_hex(con, n = 2, size = 2, endian = "big")
close(con)
```

`read_raw` *Read raw bytes*

Description

Read raw bytes

Usage

```
read_raw(con, n = 1)
```

Arguments

`con`	Connection object or raw vector. Connection objects can be created with `file()`, `url()`, `rawConnection()` or any of the other many connection creation functions.
`n`	Number of elements to read. Default: 1

Value

raw vector

See Also

Other data input functions: `read_f64()`, `read_hex()`, `read_str()`, `read_uint8()`, `scan_dbl()`

Examples

```
con <- rawConnection(charToRaw("hello12.3"))
read_raw(con, 5)
close(con)
```

read_str *Read a character string from a connection*

Description

Read character string from a connection.

Usage

```
read_str(con)

read_str_raw(con, n)

read_utf8(con)

read_utf8_raw(con, n)
```

Arguments

con	Connection object or raw vector. Connection objects can be created with `file()`, `url()`, `rawConnection()` or any of the other many connection creation functions.
n	number of characters to read.

Details

Functions which have a suffix of `_raw` are for handling character strings without a nul-terminator.

Value

single character string

See Also

Other data input functions: `read_f64()`, `read_hex()`, `read_raw()`, `read_uint8()`, `scan_dbl()`

Examples

```
con <- rawConnection(c(charToRaw("hello12.3"), as.raw(0)))
read_str(con)
close(con)

con <- rawConnection(charToRaw("hello12.3"))
read_str_raw(con, 5)
close(con)

con <- rawConnection(c(charToRaw("hello12.3"), as.raw(0)))
read_utf8(con)
```

```
close(con)

con <- rawConnection(charToRaw("hello12.3"))
read_utf8_raw(con, 3)
close(con)
```

`read_uint8` *Read integer data from a connection*

Description

Read integer values into a standard R vector of integers or alternate containers for large types

Usage

```
read_uint8(con, n = 1, endian = NULL)

read_int8(con, n = 1, endian = NULL)

read_int16(con, n = 1, endian = NULL)

read_uint16(con, n = 1, endian = NULL)

read_int32(con, n = 1, endian = NULL)

read_uint32(con, n = 1, endian = NULL, promote = NULL)

read_int64(con, n = 1, endian = NULL, promote = NULL, bounds_check = NULL)

read_uint64(con, n = 1, endian = NULL, promote = NULL, bounds_check = NULL)
```

Arguments

`con`	Connection object or raw vector. Connection objects can be created with `file()`, `url()`, `rawConnection()` or any of the other many connection creation functions.
`n`	Number of elements to read. Default: 1
`endian`	Ordering of bytes within the file when reading multi-byte values. Possible values: 'big' or 'little'. Default: NULL indicates that endian option should be retrieved from the connection object if possible (where the user has used `set_endian()`) or otherwise will be set to `"little"`
`promote`	For 'uin32', 'int64' and 'uint64' types, the range of possible values exceeds R's standard integer type. For these integer types, values will be promoted to a different container type. Possible options 'dbl', 'raw', 'hex' and 'bit64'. Default: NULL indicates that this option should be retrieved from the connection object if possible (where the user has used `set_integer_promotion()`) or otherwise will default to `"dbl"`.

`dbl` Read integer values as double precision floating point. A 'double' will hold integer values (without loss) from -(2^53) up to (2^53). A further warning will be issued if an attempt is made to store an integer value that lies outside this range

`hex` Read integers as character vector of hexadecimal strings

`raw` Read integer value as a sequence of raw bytes

`bit64` Read integer value as a vector of type `bit64::integer64`. This is valid only when reading 'int64' and 'uint64' types

`bounds_check` Check values lie within bounds of the given type. Default: NULL indicates that this option should be retrieved from the connection object if possible (where the user has used `set_bounds_check()`) or otherwise will be set to `"error"`

Details

8-bit integers `read_int8()` and `read_uint8()`

16-bit integers `read_int16()` and `read_uint16()`

32-bit integers `read_int32()` and `read_uint32()`

64-bit integers `read_int64()` and `read_uint64()`

Value

Integer data. Usually in standard R integer vector but depending on the `promote` option may be returned in alternate formats

See Also

Other data input functions: `read_f64()`, `read_hex()`, `read_raw()`, `read_str()`, `scan_dbl()`

Examples

```
# Raw vector with 16 bytes (128 bits) of dummy data
data <- as.raw(c(1:7, 0, 1:8))
con <- rawConnection(data, 'rb')
read_int64(con, n = 1)
read_uint8(con, n = 4)
close(con)
```

---

scan_dbl *Read values encoded as characters strings*

Description

A lightweight wrapper around the standard `scan()` function.

Usage

```
scan_dbl(con, n = 1, quiet = TRUE, ...)

scan_int(con, n = 1, quiet = TRUE, ...)

scan_str(con, n = 1, quiet = TRUE, ...)
```

Arguments

con	Connection object or raw vector. Connection objects can be created with file(), url(), rawConnection() or any of the other many connection creation functions.
n	Number of elements to read. Default: 1
quiet	Default: TRUE
...	further arguments passed to scan()

Details

These functions are useful when the numeric values are encoded as strings written to the file, rather than as binary data. Values must be delimited by whitespace or other specified separator. See documentation for scan() for more information.

Value

Value of the given type

See Also

Other data input functions: read_f64(), read_hex(), read_raw(), read_str(), read_uint8()

Examples

```
con <- textConnection(r"(
  type
  20 30
  3.14159
)")

scan_str(con)
scan_int(con)
scan_int(con)
scan_dbl(con)
close(con)
```

set_bounds_check *For this connection, set the response when values do not fit into given type before writing.*

Description

For this connection, set the response when values do not fit into given type before writing.

Usage

```
set_bounds_check(con, bounds_check = "error")
```

Arguments

con	Connection object or raw vector. Connection objects can be created with `file()`, `url()`, `rawConnection()` or any of the other many connection creation functions.
bounds_check	Default bounds checking behaviour. One of: 'ignore', 'warn', 'error'. Default: 'error'. This default may be over-ridden by specifying the bounds_check argument when calling individual functions.
	`ignore` No explicit checks will be made for out-of-bound values. The underlying R functions (e.g. `readBin()`, `writeBin()`) may still do checking.
	`warn` Explicit checks will be made for out-of-bound values. If any are found, then a `warning()` will be issued.
	`error` Explicit checks will be made for out-of-bound values. If any are found, then a error will be raised.

Value

Modified connection object

See Also

Other connection configuration functions: `set_endian()`, `set_eof_check()`, `set_integer_promotion()`, `set_na_check()`

Examples

```
# Open a connection and configure it so out-of-bounds values
# will cause a warning only.
con <- rawConnection(as.raw(1:8), "rb")
con <- set_bounds_check(con, bounds_check = "warn")

# This line attempts to read a value from the connection which
# is too large to store in a double precision floating point without
# loss of integer precision.
# Usually this would cause an error to be raised, but the 'bounds_check'
```

```
# option has been set to give a warning only.
read_uint64(con, n = 1, promote = "dbl")

close(con)
```

`set_endian` *Tag a connection with the preferred endianness*

Description

Tag a connection with the preferred endianness

Usage

```
set_endian(con, endian = "little")
```

Arguments

`con`	Connection object or raw vector. Connection objects can be created with `file()`, `url()`, `rawConnection()` or any of the other many connection creation functions.
`endian`	Default endianness to assign to this connection. One of either "little" or "big". Default: "little". This default may be over-ridden by specifying the `endian` argument when calling individual functions.

Value

Modified connection object

See Also

Other connection configuration functions: `set_bounds_check()`, `set_eof_check()`, `set_integer_promotion()`, `set_na_check()`

Examples

```
# Open a connection and configure it so all subsequent read/write operations
# use big-endian ordering.
con <- rawConnection(as.raw(c(0, 1, 0, 1)), "rb")
con <- set_endian(con, endian = "big")

# Future reads will be be big endian
read_uint16(con, n = 1)

# Unless over-ridden during the read
read_uint16(con, n = 1, endian = "little")

close(con)
```

`set_eof_check` *Set EOF (End-of-file) handling for this connection*

Description

When the end-of-file is reached and values are requested from the connection, how should a `read` call check and react?

Usage

```
set_eof_check(con, eof_check = "error")
```

Arguments

`con`	Connection object or raw vector. Connection objects can be created with `file()`, `url()`, `rawConnection()` or any of the other many connection creation functions.
`eof_check`	Default EOF checking behaviour. One of: 'ignore', 'warn', 'error' Default: 'error'.

`ignore` No explicit checks will be made for EOF. The underlying R functions (e.g. `readBin()`, `writeBin()`) may still do checking.

`warn` Explicit checks will be made for reading data at EOF. If this occurs, then a `warning()` will be issued.

`error` Explicit checks will be made for reading data at EOF. If any are found, then a error will be raised.

Details

Note: R's `readBin()` does not necessarily react when the end-of-file is reached, and in many situations all that will happen is that fewer data values will be returned than what was requested.

By setting this option on the connection, work is done to check the count of returned values after every call to try and detect when the end-of-file has been reached.

Value

Modified connection object

See Also

Other connection configuration functions: `set_bounds_check()`, `set_endian()`, `set_integer_promotion()`, `set_na_check()`

Examples

```
# Open a connection and configure it so reading past the end-of-file
# ignored, and operations simply return fewer values than requested
con <- rawConnection(as.raw(1:8), "rb")
con <- set_eof_check(con, eof_check = "ignore")

# There are only 8 bytes in the connection.
# Attempting to read 12 bytes will reach the end of the file.
# Because "eof_check" has been set to "ignore", there will just be
# silent truncation of the data
read_uint8(con, n = 12)

# The connection can be configured to raise an error or warning
# when EOF is reached
con <- set_eof_check(con, eof_check = "warn")
read_uint8(con, n = 12)

close(con)
```

---

set_integer_promotion *Tag a connection with the preferred integer promotion method for types larger that R's integer type i.e. uint32, uint64, int64*

Description

Tag a connection with the preferred integer promotion method for types larger that R's integer type i.e. uint32, uint64, int64

Usage

```
set_integer_promotion(con, uint32 = "dbl", int64 = "dbl", uint64 = "dbl")
```

Arguments

con	Connection object or raw vector. Connection objects can be created with `file()`, `url()`, `rawConnection()` or any of the other many connection creation functions.
uint32, int64, uint64	specifiy separate promotion methods for these types One of: 'dbl', 'hex', 'raw' and 'bit64' (for 64-bit types only) Default: 'dbl'. This default may be overridden by specifying the `promote` argument when calling individual functions. `dbl` Read in integers as doubles. Integer values above 2^53 will lose precision. `hex` Each integer is returned as a hexadecimal string `raw` A single raw vector containing all the integers in their original form `bit64` Return an `integer64` vector compatible with the `bit64` package. Note. `integer64` is a *signed* 64-bit integer

Value

Modified connection object

See Also

Other connection configuration functions: set_bounds_check(), set_endian(), set_eof_check(), set_na_check()

Examples

```
# Open a connection and configure it so all 'uint32' values are
# read as floating point and all all 'uint64' values are read as hexadecimal strings
con <- rawConnection(as.raw(c(1:7, 0, 1:7, 0, 1:7, 0, 1:7, 0)), "rb")
con <- set_integer_promotion(con, uint32 = "dbl", uint64 = "hex")

# Future reads of uint64 will return hex strings
read_uint64(con, n = 2)

# Unless over-ridden during the read
read_uint64(con, n = 1, promote = "dbl")

close(con)
```

---

set_na_check *Check for NAs in values before writing*

Description

For the majority of binary file formats, there is never the need to store or retrieve an NA value. The default behaviour of this package is to raise an error if any attempt is made to write an NA to file. Set this option to "warn" or "ignore" to modify this.

Usage

```
set_na_check(con, na_check)
```

Arguments

con	Connection object or raw vector. Connection objects can be created with file(), url(), rawConnection() or any of the other many connection creation functions.
na_check	Default NA checking behaviour. One of: 'ignore', 'warn', 'error' Default: 'error'. This default may be over-ridden by specifying the na_check argument when calling individual functions.
	ignore No explicit checks will be made for NA values The underlying R functions (e.g. readBin(), writeBin()) may still do checking.

warn Explicit checks will be made for NA values before writing. If any NAs are present, then a `warning()` will be issued.

error Explicit checks will be made for NA values before writing. If any NAs are present, then an error will be raised.

Value

Modified connection object

See Also

Other connection configuration functions: `set_bounds_check()`, `set_endian()`, `set_eof_check()`, `set_integer_promotion()`

Examples

```
# Open a connection and configure it so any attempt to write an NA
# value will cause a warning only (the default behaviour is to raise an error)
con <- rawConnection(raw(), "wb")
con <- set_na_check(con, na_check = "warn")

# This write should work without issues
write_dbl(con, c(1, 2, 3, 4))

# This write will cause a warning
write_dbl(con, c(1, 2, 3, NA))

close(con)
```

---

write_f64 *Convert values to the given type and write to a connection*

Description

Convert values to the given type and write to a connection

Usage

```
write_f64(con, x, endian = NULL, bounds_check = NULL, na_check = NULL)

write_dbl(con, x, endian = NULL, bounds_check = NULL, na_check = NULL)

write_f32(con, x, endian = NULL, bounds_check = NULL, na_check = NULL)

write_single(con, x, endian = NULL, bounds_check = NULL, na_check = NULL)

write_f16(con, x, endian = NULL, bounds_check = NULL, na_check = NULL)

write_half(con, x, endian = NULL, bounds_check = NULL, na_check = NULL)
```

Arguments

con	Connection object or raw vector. When con is a raw vector, new data will be *appended* to the vector and returned. Connection objects can be created with file(), url(), rawConnection() or any of the other many connection creation functions.
x	vector to write
endian	Ordering of bytes within the file when reading multi-byte values. Possible values: 'big' or 'little'. Default: NULL indicates that endian option should be retrieved from the connection object if possible (where the user has used set_endian()) or otherwise will be set to "little"
bounds_check	Check values lie within bounds of the given type. Default: NULL indicates that this option should be retrieved from the connection object if possible (where the user has used set_bounds_check()) or otherwise will be set to "error"
na_check	Check for NAs in the data to be written. Default: NULL indicates that this option should be retrieved from the connection object if possible (where the user has used set_na_check()) or otherwise will be set to "error"

Value

If con is a connection then this connection is returned invisibly. If con is a raw vector then new data is appended to this vector

See Also

Other data output functions: fprintf(), write_hex(), write_raw(), write_uint8(), write_utf8()

Examples

```
con <- file(tempfile(), "wb")
write_f64(con, c(1, 2, 3, 4))
close(con)
```

---

write_hex *Write hexadecimal string as raw bytes*

Description

Write hexadecimal string as raw bytes

Usage

```
write_hex(con, x, endian = NULL)
```

Arguments

con	Connection object or raw vector. When con is a raw vector, new data will be *appended* to the vector and returned. Connection objects can be created with file(), url(), rawConnection() or any of the other many connection creation functions.
x	vector to write
endian	Ordering of bytes within the file when reading multi-byte values. Possible values: 'big' or 'little'. Default: NULL indicates that endian option should be retrieved from the connection object if possible (where the user has used set_endian()) or otherwise will be set to "little"

Value

If con is a connection then this connection is returned invisibly. If con is a raw vector then new data is appended to this vector

See Also

Other data output functions: fprintf(), write_f64(), write_raw(), write_uint8(), write_utf8()

Examples

```
con <- file(tempfile(), "wb")
write_hex(con, c("ff80", "0102"))
close(con)
```

---

write_raw *Write raw bytes*

Description

Write raw bytes

Usage

```
write_raw(con, x, bounds_check = NULL)
```

Arguments

con	Connection object or raw vector. When con is a raw vector, new data will be *appended* to the vector and returned. Connection objects can be created with file(), url(), rawConnection() or any of the other many connection creation functions.
x	vector to write
bounds_check	Check values lie within bounds of the given type. Default: NULL indicates that this option should be retrieved from the connection object if possible (where the user has used set_bounds_check()) or otherwise will be set to "error"

Value

If con is a connection then this connection is returned invisibly. If con is a raw vector then new data is appended to this vector

See Also

Other data output functions: `fprintf()`, `write_f64()`, `write_hex()`, `write_uint8()`, `write_utf8()`

Examples

```
con <- file(tempfile(), "wb")
write_raw(con, as.raw(1:4))
write_raw(con, 1:4)
close(con)
```

---

write_uint8 *Convert values to the given type and write to a connection*

Description

Convert values to the given type and write to a connection

Usage

```
write_uint8(con, x, endian = NULL, bounds_check = NULL, na_check = NULL)

write_int8(con, x, endian = NULL, bounds_check = NULL, na_check = NULL)

write_uint16(con, x, endian = NULL, bounds_check = NULL, na_check = NULL)

write_int16(con, x, endian = NULL, bounds_check = NULL, na_check = NULL)

write_uint32(con, x, endian = NULL, bounds_check = NULL, na_check = NULL)

write_int32(con, x, endian = NULL, bounds_check = NULL, na_check = NULL)

write_uint64(con, x, endian = NULL, bounds_check = NULL, na_check = NULL)

write_int64(con, x, endian = NULL, bounds_check = NULL, na_check = NULL)
```

Arguments

con	Connection object or raw vector. When con is a raw vector, new data will be *appended* to the vector and returned. Connection objects can be created with `file()`, `url()`, `rawConnection()` or any of the other many connection creation functions.
x	vector to write

endian	Ordering of bytes within the file when reading multi-byte values. Possible values: 'big' or 'little'. Default: NULL indicates that endian option should be retrieved from the connection object if possible (where the user has used set_endian()) or otherwise will be set to "little"
bounds_check	Check values lie within bounds of the given type. Default: NULL indicates that this option should be retrieved from the connection object if possible (where the user has used set_bounds_check()) or otherwise will be set to "error"
na_check	Check for NAs in the data to be written. Default: NULL indicates that this option should be retrieved from the connection object if possible (where the user has used set_na_check()) or otherwise will be set to "error"

Value

If con is a connection then this connection is returned invisibly. If con is a raw vector then new data is appended to this vector

See Also

Other data output functions: fprintf(), write_f64(), write_hex(), write_raw(), write_utf8()

Examples

```
con <- file(tempfile(), "wb")
write_uint8(con, 1:4)
close(con)
```

---

write_utf8 *Write UTF8 string*

Description

write_utf8_raw() writes the string without a nul-terminator. write_utf8() includes a nul-terminator

Usage

```
write_utf8(con, x, from = "")

write_utf8_raw(con, x, from = "")
```

Arguments

con	Connection object or raw vector. When con is a raw vector, new data will be *appended* to the vector and returned. Connection objects can be created with file(), url(), rawConnection() or any of the other many connection creation functions.
x	single character string

`from` current encoding. Argument is passed to `iconv()`. Default: "" (empty string) works well for most inputs. See documentaiton for `iconv()` for more information.

Value

If `con` is a connection then this connection is returned invisibly. If `con` is a raw vector then new data is appended to this vector

See Also

Other data output functions: `fprintf()`, `write_f64()`, `write_hex()`, `write_raw()`, `write_uint8()`

Examples

```
con <- file(tempfile(), "wb")
write_utf8(con, "hello")
close(con)
```

Index