

Package ‘R.utils’

January 23, 2012

Version 1.9.11

Date 2012-01-17

Title Various programming utilities

Author Henrik Bengtsson <henrikb@braju.com>

Maintainer Henrik Bengtsson <henrikb@braju.com>

Depends R (>= 2.5.0), R.oo (>= 1.8.3)

Suggests digest (>= 0.5.1)

Description This package provides utility classes and methods useful when programming in R and developing R packages.

License LGPL (>= 2.1)

URL <http://www.braju.com/R/>

LazyLoad TRUE

Repository CRAN

Date/Publication 2012-01-23 07:45:38

R topics documented:

R.utils-package	4
addFinalizerToLast	6
Arguments	7
arrayIndex	8
as.character.binmode	9
Assert	10
attachLocally.list	11
bunzip2	12
callHooks	13
callHooks.function	14

capitalize	15
colClasses	16
commandArgs	17
copyDirectory	19
countLines	20
createFileAtomically	21
createLink	23
createWindowsShortcut	24
dataFrame	25
detachPackage	26
devDone	27
devEval	27
devGetLabel	29
devIsOpen	30
devList	31
devNew	32
devOff	33
devOptions	33
devSet	35
devSetLabel	35
dimNA<-	36
displayCode	37
doCall	38
downloadFile.character	39
env	40
eps	41
evalCapture	42
evalWithTimeout	43
extract.array	45
fileAccess	47
filePath	49
FileProgressBar	51
finalizeSession	52
findSourceTraceback	53
getAbsolutePath	54
getParent	55
getRelativePath	55
GString	56
gzip	59
hasUrlProtocol	61
hpaste	61
inAnyInterval.numeric	63
insert	64
intervalsToSeq.matrix	66
intToBin	66
isAbsolutePath	67
isDirectory	68
isEof.connection	68

isFile	69
isOpen.character	70
isPackageInstalled	70
isPackageLoaded	71
isReplicated	72
isSingle	73
isUrl	74
isZero	75
Java	76
jpeg2	78
lastModified	79
LComments	79
listDirectory	80
loadObject	81
mapToIntervals.numeric	82
mergeIntervals.numeric	83
mkdirs	84
NullVerbose	85
onGarbageCollect	86
onSessionExit	87
Options	88
patchCode	90
png2	92
popBackupFile	92
popTemporaryFile	93
printf	94
ProgressBar	95
pushBackupFile	97
pushTemporaryFile	98
queryRCmdCheck	100
readBinFragments	101
readRdHelp	104
readTable	104
readTableIndex	106
readWindowsShortcut	107
relibrary	108
removeDirectory	109
resample	109
resetWarnings	110
saveObject	111
seqToHumanReadable	112
seqToIntervals	112
setOption	113
Settings	114
SmartComments	116
sourceDirectory	118
sourceTo	119
splitByPattern	121

stext	122
subplots	123
System	124
TextStatusBar	125
TimeoutException	127
toEPS	129
touchFile	129
toUrl	131
unwrap.array	131
VComments	132
Verbose	135
whichVector.logical	138
wrap.array	140
writeBinFragments	143
writeDataFrame.data.frame	144

Index	146
--------------	------------

R.utils-package	<i>Package R.utils</i>
-----------------	------------------------

Description

This package provides utility classes and methods useful when programming in R and developing R packages.

Warning: The Application Programming Interface (API) of the classes and methods in this package may change. Classes and methods are considered either to be stable, or to be in beta or alpha (pre-beta) stage. See list below for details.

The main reason for publishing this package on CRAN although it lacks a stable API, is that its methods and classes are used internally by other packages on CRAN that the author has published.

For package history, see `showHistory(R.utils)`.

Requirements

This package requires the **R.oo** package [1].

Installation and updates

To install this package do:

```
install.packages("R.utils")
```

To get started

- [Arguments](#)[alpha] Methods for common argument processing.
- [Assert](#)[alpha] Methods for assertion of values and states.
- [GString](#)[alpha] A character string class with methods for simple substitution.
- [Java](#)[beta] Reads and writes Java streams.
- [Options](#)[alpha] Tree-structured options queried in a file-system like manner.
- [Settings](#)[alpha] An Options class for reading and writing package settings.
- [ProgressBar](#)[beta] Text-based progress bar.
- [FileProgressBar](#)[beta] A ProgressBar that reports progress as file size.
- [System](#)[alpha] Methods for access to system.
- [Verbose](#)[alpha] A class for verbose and log output. Utilized by the VComments and LComments classes.
- [SmartComments](#), [VComments](#), [LComments](#)[alpha] Methods for preprocessing source code comments of certain formats into R code.

In addition to the above, there is a large set of function for file handling such as support for reading/following Windows Shortcut links, but also other standalone utility functions. See package index for a list of these. These should also be considered to be in alpha or beta stage.

How to cite this package

Whenever using this package, please cite [1] as

```
@INPROCEEDINGS{BengtssonH_2003,
  author      = {Henrik Bengtsson},
  title       = {The {R.oo} package - Object-Oriented Programming
                with References Using Standard {R} Code},
  booktitle   = {Proceedings of the 3rd International Workshop on
                Distributed Statistical Computing (DSC 2003)},
  year        = {2003},
  editor      = {Kurt Hornik and Friedrich Leisch and Achim Zeileis},
  address     = {Vienna, Austria},
  month       = {March},
  issn        = {1609-395X},
  howpublished = {http://www.ci.tuwien.ac.at/Conferences/DSC-2003/},
}
```

Wishlist

Here is a list of features that would be useful, but which I have too little time to add myself. Contributions are appreciated.

- Write a TclTkProgressBar class.
- Improve/stabilize the GString class.

- Mature the SmartComments classes. Also add AComments and PComments for assertion and progress/status comments.

If you consider implement some of the above, make sure it is not already implemented by downloading the latest "devel" version!

License

The releases of this package is licensed under LGPL version 2.1 or newer.

The development code of the packages is under a private licence (where applicable) and patches sent to the author fall under the latter license, but will be, if incorporated, released under the "release" license above.

References

- 1 H. Bengtsson, *The R.oo package - Object-Oriented Programming with References Using Standard R Code*, In Kurt Hornik, Friedrich Leisch and Achim Zeileis, editors, Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003), March 20-22, Vienna, Austria. <http://www.ci.tuwien.ac.at/Conferences/DSC-2003/Proceedings/>

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

addFinalizerToLast *Modifies .Last() to call 'finalizeSession()*

Description

Modifies .Last() to call 'finalizeSession() *before* calling the default .Last() function.

Note that .Last() is *not* guaranteed to be called when the R session finished. For instance, the user may quit R by calling `quit(runLast=FALSE)` or run R in batch mode.

Note that this function is called when the R.utils package is loaded.

Usage

```
## Default S3 method:
addFinalizerToLast(...)
```

Arguments

... Not used.

Value

Returns (invisibly) `TRUE` if .Last() was modified, otherwise `FALSE`.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[onSessionExit\(\)](#).

Arguments

Static class to validate and process arguments

Description

Package: R.utils

Class Arguments

[Object](#)

~~|

~~+--Arguments

Directly known subclasses:

public static class **Arguments**

extends [Object](#)

Fields and Methods**Methods:**

getCharacter	-
getCharacters	Coerces to a character vector and validates.
getDirectory	-
getDouble	-
getDoubles	Coerces to a double vector and validates.
getEnvironment	Gets an existing environment.
getIndex	-
getIndices	Coerces to a integer vector and validates.
getInstanceOf	Gets an instance of the object that is of a particular class.
getInteger	-
getIntegers	Coerces to a integer vector and validates.
getLogical	-
getLogicals	Coerces to a logical vector and validates.
getNumeric	-
getNumerics	Coerces to a numeric vector and validates.

<code>getReadablePath</code>	-
<code>getReadablePathname</code>	Gets a readable pathname.
<code>getReadablePathnames</code>	Gets a readable pathname.
<code>getRegularExpression</code>	Gets a valid regular expression pattern.
<code>getVector</code>	Validates a vector.
<code>getVerbose</code>	Coerces to Verbose object.
<code>getWritablePath</code>	-
<code>getWritablePathname</code>	Gets a writable pathname.

Methods inherited from Object:

`$`, `$<-`, `[]`, `[[<-`, `as.character`, `attach`, `attachLocally`, `clearCache`, `clearLookupCache`, `clone`, `detach`, `equals`, `extend`, `finalize`, `gc`, `getEnvironment`, `getFieldModifier`, `getFieldModifiers`, `getFields`, `getInstantiationTime`, `getStaticInstance`, `hasField`, `hashCode`, `ll`, `load`, `objectSize`, `print`, `registerFinalizer`, `save`

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

arrayIndex

Converts vector indices to array indices

Description

Converts vector indices to array indices assuming last array dimension to "move fastest", e.g. matrices are stored column by column.

Usage

```
## Default S3 method:
arrayIndex(i, dim, ...)
```

Arguments

<code>i</code>	A vector of vector indices to be converted to array indices.
<code>dim</code>	A non-empty numeric vector specifying the dimension of the array.
<code>...</code>	Not used.

Value

Returns an **integer matrix** of `length(i)` rows and `length(dim)` columns.

References

[1] H. Bengtsson, *Bayesian Networks - a self-contained introduction with implementation remarks*, Master's Thesis in Computer Science, Mathematical Statistics, Lund Institute of Technology, 1999.

See Also

From R v2.11.0 there is `arrayInd()`, which does the same thing as this method. `which()` with argument `arr.ind=TRUE`.

Examples

```
# Single index
print(arrayIndex(21, dim=c(4,3,3)))

# Multiple indices
print(arrayIndex(20:23, dim=c(4,3,3)))

# Whole array
x <- array(1:30, dim=c(5,6))
print(arrayIndex(1:length(x), dim=dim(x)))

# Find (row,column) of maximum value
m <- diag(4-abs(-4:4))
print(arrayIndex(which.max(m), dim=dim(m)))
```

as.character.binmode *Converts a binary/octal/hexadecimal number into a string*

Description

Converts a binary/octal/hexadecimal number into a string.

Usage

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

Arguments

x	Object to be converted.
...	Not used.

Value

Returns a `character`.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

`as.character.octmode()`, cf. `octmode.intToBin()` (incl. `intToOct()` and `intToHex()`).

 Assert

The Assert class

Description

Package: R.utils

Class Assert

[Object](#)

~~|

~~+--Assert

Directly known subclasses:

public static class **Assert**
 extends [Object](#)

Usage

Assert(...)

Arguments

... Not used.

Fields and Methods

Methods:

check	Static method asserting that a generic condition is true.
inherits	Static method asserting that an object inherits from of a certain class.
isMatrix	Static method asserting thatan object is a matrix.
isScalar	Static method asserting thatan object is a single value.
isVector	Static method asserting thatan object is a vector.

Methods inherited from Object:

\$, \$<-, [, [[<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

attachLocally.list *Assigns an objects elements locally*

Description

Assigns an objects elements locally.

Usage

```
## S3 method for class 'list'  
attachLocally(object, fields=NULL, excludeFields=NULL, overwrite=TRUE, envir=parent.frame(), ...)
```

Arguments

object	An object with named elements such as an environment , a list , or a data.frame .
fields	A character vector specifying elements to be copied. If <code>NULL</code> , all elements are considered.
excludeFields	A character vector specifying elements not to be copied. This has higher priority than <code>fields</code> .
overwrite	If <code>FALSE</code> , fields that already exists will not be copied.
envir	The environment where elements are copied to.
...	Not used.

Value

Returns (invisibly) a [character vector](#) of the fields copied.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[attachLocally\(\)](#) of class `Object`. [attach\(\)](#).

Examples

```
foo <- function(object) {
  cat("Local objects in foo():\n")
  print(ls())

  attachLocally(object)

  cat("\nLocal objects in foo():\n")
  print(ls())

  for (name in ls()) {
    cat("\nObject '", name, "':\n", sep="")
    print(get(name, inherits=FALSE))
  }
}

a <- "A string"
l <- list(a=1:10, msg="Hello world", df=data.frame(a=NA, b=2))
foo(l)
print(a)
```

bunzip2*Bunzip a file*

Description

Bunzip a file.

Usage

```
## Default S3 method:
bunzip2(filename, destname=gsub("[.]bz2$", "", filename), overwrite=FALSE, remove=TRUE, BFR.SIZE=1e+0
```

Arguments

filename	Pathname of (bzip2'ed) input file to be bunzip2'ed.
destname	Pathname of output file.
overwrite	If the output file already exists, then if <code>overwrite</code> is <code>TRUE</code> the file is silently overwriting, otherwise an exception is thrown.
remove	If <code>TRUE</code> , the input file is removed afterward, otherwise not.
BFR.SIZE	The number of bytes read in each chunk.
...	Not used.

Details

Internally `bzfile()` (see [connections](#)) is used to read chunks of the bzip2'ed file, which are then written to the output file.

Value

Returns the number of (input/compressed) bytes read.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

callHooks	<i>Call hook functions by hook name</i>
-----------	---

Description

Call hook functions by hook name.

Usage

```
## Default S3 method:
callHooks(hookName, ..., removeCalledHooks=FALSE)
```

Arguments

```
hookName      A character string of the hook name.
...           Argument passed to each hook function.
removeCalledHooks
              If TRUE, called hook functions are removed, otherwise not.
```

Value

Returns (invisibly) whatever `callHooks.list()` returns.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

Internally, after retrieving hook functions, `callHooks.list()` is called.

Examples

```
# -----
# Example 1
# -----
# First, clean up if called more than once
setHook("myFunction.onEnter", NULL, action="replace")
setHook("myFunction.onExit", NULL, action="replace")

runConference <- function(...) {
```

```

    callHooks("myFunction.onEnter")
    cat("Speaker A: Hello there...\n")
    callHooks("myFunction.onExit")
  }

  setHook("myFunction.onEnter", function(...) {
    cat("Chair: Welcome to our conference.\n")
  })

  setHook("myFunction.onEnter", function(...) {
    cat("Chair: Please welcome Speaker A!\n")
  })

  setHook("myFunction.onExit", function(...) {
    cat("Chair: Please thanks Speaker A!\n")
  })

  runConference()

# -----
# Example 2
# -----
setHook("randomNumber", NULL, action="replace")
setHook("randomNumber", rnorm)      # By function
setHook("randomNumber", "rexp")    # By name
setHook("randomNumber", "runif")   # Non-existing name
setHook("randomNumber", .GlobalEnv) # Not a function

res <- callHooks("randomNumber", n=1)
str(res)
cat("Number of hooks: ", length(res), "\n");
isErroneous <- unlist(lapply(res, FUN=function(x) !is.null(x$exception)));
cat("Erroneous hooks: ", sum(isErroneous), "\n");

```

callHooks.function *Call hook functions*

Description

Call hook functions.

Usage

```
## S3 method for class 'function'
callHooks(hooks, ...)
```

Arguments

hooks A [function](#) or a [list](#) of hook [functions](#) or names of such.
... Argument passed to each hook function.

Value

Returns (invisibly) a [list](#) that is named with hook names, if possible. Each element in the list is in turn a [list](#) with three element: `fcn` is the hook function called, `result` is its return value, and `exception` is the exception caught or `NULL`.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

See [callHooks\(\)](#) to call hook function by name.

capitalize

Capitalizes/decapitalizes each character string in a vector

Description

Capitalizes/decapitalized (making the first letter upper/lower case) of each character string in a vector.

Usage

```
## Default S3 method:  
capitalize(str, ...)  
## Default S3 method:  
decapitalize(str, ...)
```

Arguments

str A [vector](#) of [character](#) strings to be capitalized.
... Not used.

Value

Returns a [vector](#) of [character](#) strings of the same length as the input vector.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[toCamelCase](#).

Examples

```
words <- strsplit("Hello wOrld", " ")[[1]];
cat(paste(toupper(words), collapse=" "), "\n")      # "HELLO WORLD"
cat(paste(tolower(words), collapse=" "), "\n")     # "hello world"
cat(paste(capitalize(words), collapse=" "), "\n")  # "Hello WOrld"
cat(paste(decapitalize(words), collapse=" "), "\n") # "hello wOrld"

# Sanity checks
stopifnot(paste(toupper(words), collapse=" ") == "HELLO WORLD")
stopifnot(paste(tolower(words), collapse=" ") == "hello world")
stopifnot(paste(capitalize(words), collapse=" ") == "Hello WOrld")
stopifnot(paste(decapitalize(words), collapse=" ") == "hello wOrld")
```

colClasses

Creates a vector of column classes used for tabular reading

Description

Creates a vector of column classes used for tabular reading based on a compact format string.

Usage

```
## Default S3 method:
colClasses(fmt, ...)
```

Arguments

`fmt` A [character](#) string specifying the column-class format. This string is first translated by [sprintf\(\)](#).

`...` Optional arguments for the [sprintf\(\)](#) translation.

Value

Returns a [vector](#) of [character](#) strings.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[read.table](#).

Examples

```

# All predefined types
print(colClasses("-?cdfilnrzDP"))
## [1] "NULL"      "NA"        "character" "double"
## [5] "factor"    "integer"   "logical"   "numeric"
## [9] "raw"       "complex"   "Date"      "POSIXct"

# A string in column 1, integers in column 4 and 5, rest skipped
print(colClasses("c--ii---"))
## [1] "character" "NULL"      "NULL"      "integer"
## [5] "integer"   "NULL"      "NULL"      "NULL"
## [9] "NULL"

# Repeats and custom column classes
c1 <- colClasses("3c{MyClass}3{foo}")
print(c1)
## [1] "character" "character" "character" "MyClass"
## [5] "foo"       "foo"       "foo"

# Passing repeats and class names using sprintf() syntax
c2 <- colClasses("%dc{%s}%d{foo}", 3, "MyClass", 3)
stopifnot(identical(c1, c2))

# Repeats of a vector of column classes
c3 <- colClasses("3{MyClass,c}")
print(c3)
## [1] "MyClass"   "character" "MyClass"   "character"
## [4] "MyClass"   "character"

# Large number repeats
c4 <- colClasses("321{MyClass,c,i,d}")
c5 <- rep(c("MyClass", "character", "integer", "double"), times=321)
stopifnot(identical(c4, c5))

```

commandArgs

Extract Command Line Arguments

Description

Provides access to a copy of the command line arguments supplied when this R session was invoked. This function is backward compatible with `commandArgs()` of the **base** package, but adds more features.

Usage

```
commandArgs(asValues=FALSE, excludeReserved=FALSE, excludeEnvVars=FALSE, os=NULL, ...)
```

Arguments

asValues	If TRUE , a named list is returned, where command line arguments of type <code>--foo</code> will be returned as TRUE with name <code>foo</code> , and arguments of type <code>-foo=value</code> will be returned as character string value with name <code>foo</code> . In addition, if <code>-foo</code> value is given, this is interpreted as <code>-foo=value</code> , as long as <code>value</code> does not start with a double dash (<code>--</code>).
excludeReserved	If TRUE , arguments reserved by R are excluded, otherwise not. Which the reserved arguments are depends on operating system. For details, see Appendix B on "Invoking R" in <i>An Introduction to R</i> .
excludeEnvVars	If TRUE , arguments that assigns environment variable are excluded, otherwise not. As described in R <code>--help</code> , these are arguments of format <code><key>=<value></code> .
os	A vector of character strings specifying which set of reserved arguments to be used. Possible values are "unix", "mac", "windows", "ANY" or "current". If "current", the current platform is used. If "ANY" or NULL , all three OSs are assumed for total cross-platform compatibility.
...	Passed to <code>commandArgs()</code> of the base package.

Details

This function should be fully backward compatible with the same function in the base package.

Value

If `asValue` is **FALSE**, a **character vector** is returned, which contains the name of the executable and the non-parsed user-supplied arguments.

If `asValue` is **TRUE**, a named **list** containing is returned, which contains the the executable and the parsed user-supplied arguments.

The first element is the name of the executable by which **R** was invoked. As far as I am aware, the exact form of this element is platform dependent. It may be the fully qualified name, or simply the last component (or `basename`) of the application. The attribute `isReserved` is a **logical vector** specifying if the corresponding command line argument is a reserved **R** argument or not.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

`commandArgs()`, `Platform()`

Examples

```
#####
# Non-parsed command-line arguments
#####
# Display how this instance of R was invoked.
```

```

cmd <- paste(commandArgs(), collapse=" ")
cat("How R was invoked:\n");
cat(cmd, "\n")

# Get all arguments
args <- commandArgs()
print(args)

# Get only "private" arguments and not the name of the R executable.
args <- commandArgs(excludeReserved=TRUE)[-1]
print(args)

# Assert backward compatibility
args0 <- base::commandArgs()
args <- commandArgs()
stopifnot(all.equal(args, args0, check.attributes=FALSE))

#####
# Parsed command-line arguments
#####
# Get all arguments as a named list, e.g. if R is started as:
#
#   R DATAPATH=../data --args --root="do da" --foo bar --details --a=2
#
# then 'args' below will equal
#
#   list(R=NA, DATAPATH="../data" args=TRUE, root="do da",
#        foo="bar", details=TRUE, a="2")
args <- commandArgs(asValue=TRUE)
str(args)

# Turn arguments into R variables
args <- commandArgs(asValue=TRUE, excludeReserved=TRUE)[-1]
keys <- attachLocally(args)
cat("Command-line arguments attached to global environment:\n");
print(keys);
str(mget(keys, envir=globalenv()))

```

copyDirectory

Copies a directory

Description

Copies a directory.

Usage

```

## Default S3 method:
copyDirectory(from, to=".", ..., private=TRUE, recursive=TRUE)

```

Arguments

from	The pathname of the source directory to be copied.
to	The pathname of the destination directory.
...	Additional arguments passed to <code>file.copy()</code> , e.g. <code>overwrite</code> .
private	If <code>TRUE</code> , files (and directories) starting with a period is also copied, otherwise not.
recursive	If <code>TRUE</code> , subdirectories are copied too, otherwise not.

Value

Returns (invisibly) a `character vector` of pathnames copied.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

countLines	<i>Counts the number of lines in a text file</i>
------------	--

Description

Counts the number of lines in a text file by counting the number of occurrences of platform-independent newlines (CR, LF, and CR+LF [1]), including a last line with neither. An empty file has zero lines.

Usage

```
## Default S3 method:
countLines(file, chunkSize=5e+07, ...)
```

Arguments

file	A <code>connection</code> or a pathname.
chunkSize	The number of bytes read in each chunk.
...	Not used.

Value

Returns an non-negative `integer`.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

References

[1] Page *Newline*, Wikipedia, July 2008. <http://en.wikipedia.org/wiki/Newline>

Examples

```
pathname <- system.file("NEWS", package="R.utils");
n <- countLines(pathname);
n2 <- length(readLines(pathname));
stopifnot(n == n2);
```

createFileAtomically *Creates a file atomically*

Description

Creates a file atomically by first creating and writing to a temporary file which is then renamed.

Usage

```
## Default S3 method:
createFileAtomically(filename, path=NULL, FUN, ..., skip=FALSE, overwrite=FALSE, backup=TRUE, verbose=FALSE)
```

Arguments

filename	The filename of the file to create.
path	The path to the file.
FUN	A function that creates and writes to the pathname that is passed as the first argument. This pathname is guaranteed to be a non-existing temporary pathname.
...	Additional argumentes passed to pushTemporaryFile() and popTemporaryFile() .
skip	If TRUE and a file with the same pathname already exists, nothing is done/written.
overwrite	If TRUE and a file with the same pathname already exists, the existing file is overwritten. This is also done atomically such that if the new file was not successfully created, the already original file is restored. If restoration also failed, the original file remains as the pathname with suffix ".bak" appended.
backup	If TRUE and a file with the same pathname already exists, then it is backed up while creating the new file. If the new file was not successfully created, the original file is restored from the backup copy.
verbose	A logical or Verbose .

Value

Returns (invisibly) the pathname.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

Internally, [pushTemporaryFile\(\)](#) and [popTemporaryFile\(\)](#) are used for working toward a temporary file, and [pushBackupFile\(\)](#) and [popBackupFile\(\)](#) are used for backing up and restoring already existing file.

Examples

```
# -----
# Create a file atomically
# -----
n <- 10;
createFileAtomically("foobar.txt", FUN=function(pathname) {
  cat(file=pathname, "This file was created atomically.\n");
  cat(file=pathname, "Timestamp: ", as.character(Sys.time()), "\n", sep="");
  for (kk in 1:n) {
    cat(file=pathname, kk, "\n", append=TRUE);
    Sys.sleep(0.1);
  }
  cat(file=pathname, "END OF FILE\n", append=TRUE);
}, overwrite=TRUE)

bfr <- readLines("foobar.txt");
cat(bfr, sep="\n");

# -----
# Overwrite the file atomically (emulate write failure)
# -----
tryCatch({
  createFileAtomically("foobar.txt", FUN=function(pathname) {
    cat(file=pathname, "Trying to create a new file.\n");
    cat(file=pathname, "Writing a bit, but then an error...\n", append=TRUE);
    # Emulate write error
    stop("An error occurred while writing to the new file.");
    cat(file=pathname, "END OF FILE\n", append=TRUE);
  }, overwrite=TRUE)
}, error = function(ex) {
  print(ex$message);
})

# The original file was never overwritten
bfr2 <- readLines("foobar.txt");
cat(bfr2, sep="\n");
stopifnot(identical(bfr2, bfr));

# The partially temporary file remains
stopifnot(isFile("foobar.txt.tmp"));
bfr3 <- readLines("foobar.txt.tmp");
cat(bfr3, sep="\n");

file.remove("foobar.txt.tmp");
```

createLink	<i>Creates a link to a file or a directory</i>
------------	--

Description

Creates a link to a file or a directory. First it tries to create a (Unix-like) symbolic link and if that was not successful it tries to create a Windows Shortcut link. If neither works, an exception is thrown.

Usage

```
## Default S3 method:  
createLink(link=".", target, overwrite=FALSE, methods=getOption("createLink/args/methods", c("unix-sy
```

Arguments

link	The path or pathname of the link to be created. If "." (or <code>NULL</code>), it is inferred from the target argument.
target	The target file or directory to which the shortcut should point to.
overwrite	If <code>TRUE</code> , an existing link file is overwritten, otherwise not.
methods	A character vector specifying what methods (and in what order) should be tried for creating links.
...	Not used.

Value

Returns (invisibly) the path or pathname to the destination.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[createWindowsShortcut\(\)](#) and [file.symlink\(\)](#)

createWindowsShortcut *Creates a Microsoft Windows Shortcut (.lnk file)*

Description

Creates a Microsoft Windows Shortcut (.lnk file).

Usage

```
## Default S3 method:  
createWindowsShortcut(pathname, target, overwrite=FALSE, ...)
```

Arguments

pathname	The pathname (with file extension *.lnk) of the link file to be created.
target	The target file or directory to which the shortcut should point to.
overwrite	If TRUE , an existing link file is overwritten, otherwise not.
...	Not used.

Value

Returns (invisibly) the pathname.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

References

[1] Create a windows shortcut (.LNK file), SS64.com, <http://ss64.com/nt/shortcut.html>

See Also

[readWindowsShortcut\(\)](#)

Examples

```
# Create Windows Shortcut links to a directory and a file  
targets <- list(  
  system.file(package="R.utils"),  
  system.file("DESCRIPTION", package="R.utils")  
)  
  
for (kk in seq(along=targets)) {  
  cat("Link #", kk, "\n", sep="");  
  
  target <- targets[[kk]];
```

```

cat("Target: ", target, "\n", sep="");

# Name of *.lnk file
pathname <- sprintf("%s.LNK", tempfile())

tryCatch({
  # Will only work on Windows systems with support for VB scripting
  createWindowsShortcut(pathname, target=target)
}, error = function(ex) {})

# Was it created?
if (isFile(pathname)) {
  cat("Created link file: ", pathname, "\n", sep="");

  # Validate that it points to the correct target
  dest <- filePath(pathname, expandLinks="any")
  cat("Available target: ", dest, "\n", sep="");

  file.remove(pathname)
  stopifnot(tolower(dest) == tolower(target))
}
}

```

dataFrame

Allocates a data frame with given column classes

Description

Allocates a data frame with given column classes.

Usage

```
## Default S3 method:
dataFrame(colClasses, nrow=1, ...)
```

Arguments

colClasses	A character vector of column classes, cf. read.table .
nrow	An integer specifying the number of rows of the allocated data frame.
...	Not used.

Value

Returns an $N \times K$ [data.frame](#) where N equals `nrow` and K equals `length(colClasses)`.

See Also

[data.frame](#).

Examples

```
df <- dataFrame(colClasses=c(a="integer", b="double"), nrow=10)
df[,1] <- sample(1:nrow(df))
df[,2] <- rnorm(nrow(df))
print(df)
```

detachPackage	<i>Detaches a packages by name</i>
---------------	------------------------------------

Description

Detaches a packages by name, if loaded.

Usage

```
## Default S3 method:
detachPackage(pkgname, ...)
```

Arguments

pkgname	A character string of the package name to be detached.
...	Not used.

Value

Returns (invisibly) [TRUE](#) if package was detached, otherwise [FALSE](#).

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[detach\(\)](#).

devDone	<i>Closes an open device unless it is a on-screen (interactive) device</i>
---------	--

Description

Closes an open device unless it is a on-screen (interactive) device.

Usage

```
devDone(which=dev.cur(), ...)
```

Arguments

which	An index (numeric) or a label (character).
...	Not used.

Value

Returns nothing.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[devOff\(\)](#). [dev.interactive](#).

devEval	<i>Opens a new device, evaluate (graphing) code, and closes device</i>
---------	--

Description

Opens a new device, evaluate (graphing) code, and closes device.

Usage

```
devEval(type=getOption("device"), expr, envir=parent.frame(), name="Rplot", tags=NULL, ..., ext=subst
```

Arguments

type	Specifies the type of device to be used by <code>devNew</code> .
expr	The <code>expression</code> of graphing commands to be evaluated.
envir	The <code>environment</code> where expr should be evaluated.
name, tags	The fullname name of the image is specified as the name with optional comma-separated tags appended.
ext	The filename extension of the image file generated, if any. By default, it is inferred from argument type.
...	Additional arguments passed to <code>devNew</code> .
filename	The filename of the image saved, if any. See also below.
path	The directory where then image should be saved, if any.
field	An optional <code>character</code> string specifying a specific field of the named result <code>list</code> to be returned.
force	If <code>TRUE</code> , and the image file already exists, then it is overwritten, otherwise not.

Value

Returns a named `list` with items specifying for instance the pathname, the fullname etc of the generated image. If argument `field` is given, then the value of the corresponding element is returned. *Note that the return value may be changed in future releases.*

Generated image file

If created, the generated image file is saved in the directory specified by argument `path` with a filename consisting of the name followed by optional comma-separated tags and a filename extension given by argument `ext`.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

To change default device parameters such as the width or the height, `devOptions()`. `devNew()`.

Examples

```
res <- devEval("png", name="MyPlot", tags=c("10", "rnd"), aspectRatio=0.7, {
  plot(1:10)
})
print(res$pathname)
# [1] "figures/MyPlot,10,rnd.png"

res <- devEval("pdf", name="MyPlot", tags=c("10", "rnd"), aspectRatio=0.7, {
  plot(1:10)
})
print(res$pathname)
```

```
# [1] "figures/MyPlot,10,rnd.pdf"

res <- devEval("eps", name="MyPlot", tags=c("10", "rnd"), aspectRatio=0.7, {
  plot(1:10)
})
print(res$pathname)
# [1] "figures/MyPlot,10,rnd.eps"
```

devGetLabel *Gets the label of a device*

Description

Gets the label of a device.

Usage

```
devGetLabel(which=dev.cur(), ...)
```

Arguments

which	An index (numeric) or a label (character).
...	Not used.

Value

Returns a [character](#) string.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[devSetLabel\(\)](#) and [devList\(\)](#).

devIsOpen	<i>Checks if a device is open or not</i>
-----------	--

Description

Checks if a device is open or not.

Usage

```
devIsOpen(which=dev.cur(), ...)
```

Arguments

which	An index (numeric) or a label (character).
...	Not used.

Value

Returns **TRUE** if the device is open, otherwise **FALSE**.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

Examples

```
# -----  
# Use devices for conditional processing of code.  
# Close devices to rerun code.  
# -----  
cat("Currently opened device:\n")  
print(devList())  
  
# Alt A: Use device index counter (starting with the 16:th)  
fig <- 15  
if (!devIsOpen(fig <- fig + 1)) {  
  devSet(fig)  
  cat("Figure", fig, "\n")  
  plot(1:10)  
}  
cat("Currently opened device:\n")  
print(devList())  
  
if (!devIsOpen(fig <- fig + 1)) {  
  devSet(fig)  
  cat("Figure", fig, "\n")  
  plot(1:10)  
}  
cat("Currently opened device:\n")
```

```
print(devList())

# Alt B: Use device labels
if (!devIsOpen(label <- "part 1")) {
  devSet(label)
  cat("Part 1\n")
  plot(1:10)
}
cat("Currently opened device:\n")
print(devList())

if (!devIsOpen(label <- "part 2")) {
  devSet(label)
  cat("Part 2\n")
  plot(1:10)
}
cat("Currently opened device:\n")
print(devList())
```

devList

Lists the indices of the open devices named by their labels

Description

Lists the indices of the open devices named by their labels.

Usage

```
devList(...)
```

Arguments

... Not used.

Value

Returns a named [integer vector](#).

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[dev.list\(\)](#).

devNew	<i>Opens a new device</i>
--------	---------------------------

Description

Opens a new device.

Usage

```
devNew(type=getOption("device"), ..., aspectRatio=1, par=NULL, label=NULL)
```

Arguments

type	A character string specifying the type of device to be opened. This string should match the name of an existing device function .
...	Additional arguments passed to the device function , e.g. width and height.
aspectRatio	A numeric ratio specifying the aspect ratio of the image. See below.
par	An optional named list of graphical settings applied, that is, passed to par , immediately after opening the device.
label	An optional character string specifying the label of the opened device.

Value

Returns what the device [function](#) returns.

Aspect ratio

The aspect ratio of an image is the height relative to the width. If argument height is not given (or `NULL`), it is calculated as `aspectRatio*width` as long as they are given. Likewise, if argument width is not given (or `NULL`), it is calculated as `width/aspectRatio` as long as they are given. If neither width nor height is given, then width defaults to `devOptions(type)$width`. If both width and height are given, then `aspectRatio` is ignored.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[devDone\(\)](#) and [devOff\(\)](#). For simplified generation of image files, see [devEval\(\)](#).

devOff	<i>Closes a device</i>
--------	------------------------

Description

Closes a device.

Usage

```
devOff(which=dev.cur(), ...)
```

Arguments

which	An index (numeric) or a label (character).
...	Not used.

Value

Returns what [dev.off\(\)](#) returns.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[devDone\(\)](#). Internally, [dev.off\(\)](#) is used.

devOptions	<i>Gets the default device options</i>
------------	--

Description

Gets the default device options as given by predefined devices options adjusted for the default arguments of the device function.

Usage

```
devOptions(type=c("bmp", "cairo_pdf", "cairo_ps", "eps", "jpeg", "jpeg2", "pdf", "pictex", "png", "png"))
```

Arguments

type	A character string specifying the device.
custom	If TRUE , also the default settings specific to this function is returned. For more details, see below.
special	A logical . For more details, see below.
...	Optional named arguments for setting new defaults. For more details, see below.
reset	If TRUE , the device options are reset to R defaults.

Details

If argument `special` is [TRUE](#), then the `'width'` and `'height'` options are adjusted according to the rules explained for argument `'paper'` in [pdf](#), [postscript](#), and [xfig](#).

Value

Returns a named [list](#). If the requested device does not exist (certain devices are OS specific), then an empty [list](#) is returned.

Setting new defaults

When setting device options, the `getOption("devOptions")[[type]]` option is modified. This means that for such options to be effective, any device function needs to query also such options, which for instance is done by [devNew\(\)](#).

Also, for certain devices (`eps`, `pdf`, `postscript`, `quartz`, `windows` and `x11`), builtin R device options are set.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

Examples

```
# Tabulate some of the default settings for known devices
knownDevices <- eval(formals(devOptions)$type)
fields <- c("width", "height", "bg", "fg", "pointsize")
opts <- sapply(knownDevices, FUN=function(type) {
  devOptions(type)[fields]
})
rownames(opts) <- fields
opts <- t(opts)
print(opts)
```

devSet	<i>Activates a device</i>
--------	---------------------------

Description

Activates a device.

Usage

```
devSet(which=dev.next(), ...)
```

Arguments

which	An index (numeric) or a label (character). If neither, then a label corresponding to the checksum of which as generated by digest is used.
...	Not used.

Value

Returns what [dev.set\(\)](#) returns.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[devOff\(\)](#) and [devDone\(\)](#). Internally, [dev.set\(\)](#) is used.

devSetLabel	<i>Sets the label of a device</i>
-------------	-----------------------------------

Description

Sets the label of a device.

Usage

```
devSetLabel(which=dev.cur(), label, ...)
```

Arguments

which	An index (numeric) or a label (character).
label	A character string.
...	Not used.

Value

Returns nothing.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[devGetLabel\(\)](#) and [devList\(\)](#).

dimNA<- *Sets the dimension of an object with the option to infer one dimension automatically*

Description

Sets the dimension of an object with the option to infer one dimension automatically. If one of the elements in the dimension [vector](#) is [NA](#), then its value is inferred from the length of the object and the other elements in the dimension vector. If the inferred dimension is not an [integer](#), an error is thrown.

Usage

```
## Default S3 replacement method:  
dimNA(x) <- value
```

Arguments

x An R object.
value [NULL](#) of a positive [numeric vector](#) with one optional [NA](#).

Value

Returns (invisibly) what [dim<-\(\)](#) returns (see [dim\(\)](#) for more details).

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[dim\(\)](#).

Examples

```
x <- 1:12  
dimNA(x) <- c(2,NA,3)  
stopifnot(dim(x) == as.integer(c(2,2,3)))
```

displayCode	<i>Displays the contents of a text file with line numbers and more</i>
-------------	--

Description

Displays the contents of a text file with line numbers and more.

Usage

```
## Default S3 method:  
displayCode(con=NULL, code=NULL, numerate=TRUE, lines=-1, wrap=79, highlight=NULL, pager=getOption("p
```

Arguments

con	A connection or a character string filename. If code is specified, this argument is ignored.
code	A character vector of code lines to be displayed.
numerate	If <code>TRUE</code> , line are numbers, otherwise not.
lines	If a single numeric , the maximum number of lines to show. If -1, all lines are shown. If a vector of numeric , the lines numbers to display.
wrap	The (output) column numeric where to wrap lines.
highlight	A vector of line number to be highlighted.
pager	If "none", code is not displayed in a pager, but only returned. For other options, see file.show() .
...	Additional arguments passed to file.show() , which is used to display the formatted code.

Value

Returns (invisibly) the formatted code as a [character](#) string.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[file.show\(\)](#).

Examples

```
file <- system.file("DESCRIPTION", package="R.utils")
cat("Displaying: ", file, "\n", sep="")
displayCode(file)

file <- system.file("NEWS", package="R.utils")
cat("Displaying: ", file, "\n", sep="")
displayCode(file, numerate=FALSE, lines=100:110, wrap=65)

file <- system.file("NEWS", package="R.utils")
cat("Displaying: ", file, "\n", sep="")
displayCode(file, lines=100:110, wrap=65, highlight=c(101,104:108))
```

doCall

*Executes a function call with option to ignore unused arguments***Description**

Executes a function call with option to ignore unused arguments.

Usage

```
## Default S3 method:
doCall(.fcn, ..., args=NULL, alwaysArgs=NULL, .functions=.fcn, .ignoreUnusedArgs=TRUE)
```

Arguments

.fcn	A character string naming the function to be called.
...	Named arguments to be passed to the function.
args	A list of additional named arguments that will be appended to the above arguments.
alwaysArgs	A list of additional named arguments that will be appended to the above arguments and that will <i>never</i> be ignore. This is useful if you want to pass arguments to a function that accepts arguments via
.functions	A character vector of function names whos arguments should be kept. This is useful when one function passes ... to another, e.g. loess .
.ignoreUnusedArgs	If TRUE , arguments that are not accepted by the function, will not be passed to it. Otherwise, all arguments are passed.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>).

See Also

[do.call\(\)](#).

Examples

```
doCall("plot", x=1:10, y=sin(1:10), col="red", dummyArg=54,
      alwaysArgs=list(xlab="x", ylab="y"),
      .functions=c("plot", "plot.xy"))
```

downloadFile.character

Downloads a file

Description

Downloads a file.

Usage

```
## S3 method for class 'character'
downloadFile(url, filename=basename(url), path=NULL, skip=TRUE, overwrite=!skip, ..., username=NULL, password=NULL)
```

Arguments

url	A character string specifying the URL to be downloaded.
filename, path	(optional) character strings specifying the local filename and the path of the downloaded file.
skip	If TRUE , an already downloaded file is skipped.
overwrite	If TRUE , an already downloaded file is overwritten, otherwise an error is thrown.
...	Additional arguments passed to download.file .
username, password	character strings specifying the username and password for authenticated downloads. The alternative is to specify these via the URL.
binary	If TRUE , the file is downloaded exactly "as is", that is, byte by byte (recommended). which means it willand the downloaded file is empty, the file
dropEmpty	If TRUE and the downloaded file is empty, the file is ignored and NULL is returned.
verbose	A logical , integer , or a Verbose object.

Details

Currently arguments username and password are only used for downloads via URL protocol 'https'. The 'https' protocol requires that 'wget' is available on the system.

Value

Returns the local pathname to the downloaded filename, or [NULL](#) if no file was downloaded.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

Internally `download.file` is used. That function may generate an empty file if the URL is not available.

Examples

```
## Not run:
pathname <- downloadFile("http://www.r-project.org/index.html", path="www.r-project.org/")
print(pathname)

## End(Not run)
```

env	<i>Creates a new environment, evaluates an expression therein, and returns the environment</i>
-----	--

Description

Creates a new environment, evaluates an expression therein, and returns the environment.

Usage

```
env(..., hash=FALSE, parent=parent.frame(), size=29)
```

Arguments

... Arguments passed to `evalq()`, particularly a `expression` to be evaluated inside the newly created `environment`.

hash, parent, size Arguments passed to `new.env()`.

Value

Returns an `environment`.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

References

[1] R-devel thread 'Create an environment and assign objects to it in one go?' on March 9-10, 2011.

See Also

Internally `new.env()` and `evalq()` are used.

Examples

```
x <- list();

x$case1 <- env({
  # Cut'n'pasted from elsewhere
  a <- 1;
  b <- 2;
});

x$case2 <- env({
  # Cut'n'pasted from elsewhere
  foo <- function(x) x^2;
  a <- foo(2);
  b <- 1;
  rm(foo); # Not needed anymore
});

# Turn into a list of lists
x <- lapply(x, FUN=as.list);

str(x);
```

eps

EPS graphics device

Description

Device driver for Encapsulated Postscript. This driver is the same as the postscript driver where some arguments have different default values.

Usage

```
eps(file="Rplot%03d.eps", width=7, height=7, horizontal=FALSE, paper="special", ...)
```

Arguments

file	Default file name (pattern).
width, height	The width and height of the figure.
horizontal	If FALSE , an horizontal EPS file is created, otherwise a portrait file is created.
paper	A character string specifying the paper type. Overrides the default of <code>postscript()</code> .
...	Other arguments accepted by <code>postscript()</code> .

Value

A plot device is opened; nothing is returned.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[postscript](#).

Examples

```
## Not run:
eps("foo.eps", width=7, height=7)

# is identical to

postscript("foo.eps", width=7, height=7, onefile=TRUE, horizontal=FALSE)

# and

dev.print(eps, "foo.eps", ...)

# is identical to

dev.print(postscript, "foo.eps", onefile=TRUE, horizontal=FALSE, paper="special", ...)

## End(Not run)
```

evalCapture

Evaluates an expression and captures the code and/or the output

Description

Evaluates an expression and captures the code and/or the output.

Usage

```
evalCapture(expr, code=TRUE, output=code, ..., trim=TRUE, collapse="
", envir=parent.frame())
```

Arguments

expr	The expression to be evaluated.
...	Additional arguments passed to sourceTo which in turn passes arguments to source() .
code	If TRUE , the deparsed code of the expression is echoed.
output	If TRUE , the output of each evaluated subexpression is echoed.
envir	The @environment in which the expression is evaluated.
trim	If TRUE , the captured rows are trimmed.
collapse	A character string used for collapsing the captured rows. If NULL , the rows are not collapsed.

Value

Returns a [character](#) string class 'CapturedEvaluation'.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

Examples

```
print(evalCapture({
  n <- 3;
  n;

  for (kk in 1:3) {
    printf("Iteration #%d\n", kk);
  }

  print(Sys.time());

  type <- "horse";
  type;
}))

## > n <- 3
## > n
## [1] 3
## > for (kk in 1:3) {
## +   printf("Iteration #%d\n", kk)
## + }
## Iteration #1
## Iteration #2
## Iteration #3
## > print(Sys.time())
## [1] "2011-11-06 11:06:32 PST"
## > type <- "horse"
## > type
## [1] "horse"
```

evalWithTimeout

Evaluate an R expression and interrupts it if it takes too long

Description

Evaluate an R expression and interrupts it if it takes too long.

Usage

```
evalWithTimeout(..., envir=parent.frame(), timeout, cpu=timeout, elapsed=timeout, onTimeout=c("error"
```

Arguments

...	The R expression to be evaluated as passed to <code>eval()</code> .
<code>envir</code>	The <code>environment</code> in which the expression should be evaluated.
<code>timeout</code> , <code>cpu</code> , <code>elapsed</code>	A <code>numeric</code> specifying the maximum number of seconds the expression is allowed to run before being interrupted by the timeout. The <code>cpu</code> and <code>elapsed</code> arguments can be used to specify whether time should be measured in CPU time or in wall time.
<code>onTimeout</code>	A <code>character</code> specifying what action to take if a timeout event occurs.

Value

Returns the results of the expression evaluated. If timed out, `NULL` is returned if `onTimeout` was "warning" or "silent". If "error" a `TimeoutException` is thrown.

Non-supported cases

It is not possible to interrupt/break out of a "readline" prompt (e.g. `readline()` and `readLines()`) using timeouts; the timeout exception will not be thrown until after the user completes the prompt (i.e. after pressing ENTER).

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

References

[1] R help thread 'Time out for a R Function' on 2010-12-06. <http://www.mail-archive.com/r-help@r-project.org/msg119344.html>

See Also

`setTimeLimit()`

Examples

```
# -----
# Function that takes "a long" time to run
# -----
foo <- function() {
  print("Tic");
  for (kk in 1:100) {
    print(kk);
    Sys.sleep(0.1);
  }
  print("Tac");
}
```

```

# -----
# Evaluate code, if it takes too long, generate
# a timeout by throwing a TimeoutException.
# -----
res <- NULL;
tryCatch({
  res <- evalWithTimeout({
    foo();
  }, timeout=1.08);
}, TimeoutException=function(ex) {
  cat("Timeout. Skipping.\n");
})

# -----
# Evaluate code, if it takes too long, generate
# a timeout returning NULL and generate a warning.
# -----
res <- evalWithTimeout({
  foo();
}, timeout=1.08, onTimeout="warning");

# -----
# Evaluate code, if it takes too long, generate
# a timeout, and return silently NULL.
# -----
res <- evalWithTimeout({
  foo();
}, timeout=1.08, onTimeout="silent");

```

extract.array	<i>Extract a subset of an array, matrix or a vector with unknown dimensions</i>
---------------	---

Description

Extract a subset of an array, matrix or a vector with unknown dimensions.

This method is useful when you do not know the number of dimensions of the object your wish to extract values from, cf. example.

Usage

```
## S3 method for class 'array'
extract(x, ..., indices=list(...), dims=names(indices), drop=FALSE)
```

Arguments

x	An array or a matrix .
...	These arguments are by default put into the indices list .
indices	A list of index vectors to be extracted.
dims	An vector of dimensions - one per element in indices - which will be coerced to integers . If <code>NULL</code> , it will default to <code>seq(along=indices)</code> .
drop	If <code>TRUE</code> , dimensions of length one are dropped, otherwise not.

Value

Returns an [array](#).

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[slice.index\(\)](#)

Examples

```
# -----
# Example using an array with a random number of dimensions
# -----
maxdim <- 4
dim <- sample(3:maxdim, size=sample(2:maxdim, size=1), replace=TRUE)
ndim <- length(dim)
dimnames <- list()
for (kk in 1:ndim)
  dimnames[[kk]] <- sprintf("%s%d", letters[kk], 1:dim[kk])
x <- 1:prod(dim)
x <- array(x, dim=dim, dimnames=dimnames)

cat("\nArray 'x':\n")
print(x)

cat("\nExtract 'x[2:3,...]':\n")
print(extract(x, "1"=2:3))

cat("\nExtract 'x[3,2:3,...]':\n")
print(extract(x, "1"=3,"2"=2:3))

cat("\nExtract 'x[... ,2:3]':\n")
print(extract(x, indices=2:3, dims=length(dim(x))))

# -----
```

```
# Assertions
# -----
y <- array(1:24, dim=c(2,3,4))
yA <- y[, ,2:3]
yB <- extract(y, indices=list(2:3), dims=length(dim(y)))
stopifnot(identical(yB, yA))

yA <- y[,2:3,2]
yB <- extract(y, indices=list(2:3,2), dims=c(2,3), drop=TRUE)
stopifnot(identical(yB, yA))
```

fileAccess

Checks the permission of a file or a directory

Description

Checks the permission of a file or a directory.

Usage

```
## Default S3 method:
fileAccess(pathname, mode=0, safe=TRUE, ...)
```

Arguments

pathname	A character string of the file or the directory to be checked.
mode	An integer (0,1,2,4), cf. file.access() .
safe	If TRUE , the permissions are tested more carefully, otherwise file.access() is used.
...	Not used.

Details

In R there is [file.access\(\)](#) for checking whether the permission of a file. Unfortunately, that function cannot be 100% trusted depending on platform used and file system queried, cf. [1].

Value

Returns an [integer](#); 0 if the permission exists, -1 if not.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

References

- [1] R-devel thread *file.access() on network (mounted) drive on Windows Vista?* on Nov 26, 2008.
- [2] Filesystem permissions, Wikipedia, 2010. http://en.wikipedia.org/wiki/Filesystem_permissions

See Also

[file.access\(\)](#)

Examples

```
# -----
# Current directory
# -----
path <- "."

# Test for existence
print(fileAccess(path, mode=0))
# Test for execute permission
print(fileAccess(path, mode=1))
# Test for write permission
print(fileAccess(path, mode=2))
# Test for read permission
print(fileAccess(path, mode=4))

# -----
# A temporary file
# -----
pathname <- tempfile()
cat(file=pathname, "Hello world!")

# Test for existence
print(fileAccess(pathname, mode=0))
# Test for execute permission
print(fileAccess(pathname, mode=1))
# Test for write permission
print(fileAccess(pathname, mode=2))
# Test for read permission
print(fileAccess(pathname, mode=4))

file.remove(pathname)

# -----
# The 'base' package directory
# -----
path <- system.file(package="base")

# Test for existence
print(fileAccess(path, mode=0))
```

```

# Test for execute permission
print(fileAccess(path, mode=1))
# Test for write permission
print(fileAccess(path, mode=2))
# Test for read permission
print(fileAccess(path, mode=4))

# -----
# The 'base' package DESCRIPTION file
# -----
pathname <- system.file("DESCRIPTION", package="base")

# Test for existence
print(fileAccess(pathname, mode=0))
# Test for execute permission
print(fileAccess(pathname, mode=1))
# Test for write permission
print(fileAccess(pathname, mode=2))
# Test for read permission
print(fileAccess(pathname, mode=4))

```

filePath	<i>Construct the path to a file from components and expands Windows Shortcuts along the pathname from root to leaf</i>
----------	--

Description

Construct the path to a file from components and expands Windows Shortcuts along the path-name from root to leaf. This function is backward compatible with `file.path()` when argument `removeUps=FALSE` and `expandLinks="none"`, except that a (character) `NA` is return if any argument is `NA`.

This function exists on all platforms, not only Windows systems.

Usage

```

## Default S3 method:
filePath(..., fsep=.Platform$file.sep, removeUps=TRUE, expandLinks=c("none", "any", "local", "relativ

```

Arguments

...	Arguments to be pasted together to a file path and then be parsed from the root to the leaf where Windows shortcut files are recognized and expanded according to argument which in each step.
fsep	the path separator to use.
removeUps	If <code>TRUE</code> , relative paths, for instance "foo/bar/./" are shortend into "foo/", but also "./" are removed from the final pathname, if possible.

expandLinks	A character string. If "none", Windows Shortcut files are ignored. If "local", the absolute target on the local file system is used. If "relative", the relative target is used. If "network", the network target is used. If "any", the first the local, then the relative and finally the network target is searched for.
mustExist	If TRUE and if the target does not exist, the original pathname, that is, argument pathname is returned. In all other cases the target is returned.
verbose	If TRUE , extra information is written while reading.

Details

If `expandLinks==TRUE`, each component, call it *parent*, in the absolute path is processed from the left to the right as follows: 1. If a "real" directory of name *parent* exists, it is followed. 2. Otherwise, if Microsoft Windows Shortcut file with name *parent.lnk* exists, it is read. If its local target exists, that is followed, otherwise its network target is followed. 3. If no valid existing directory was found in (1) or (2), the expanded this far followed by the rest of the pathname is returned quietly. 4. If all of the absolute path was expanded successfully the expanded absolute path is returned.

Value

Returns a **character** string.

On speed

Internal `file.exists()` is call while expanding the pathname. This is used to check if there exists a Windows shortcut file named 'foo.lnk' in 'path/foo/bar'. If it does, 'foo.lnk' has to be followed, and in other cases 'foo' is ordinary directory. The `file.exists()` is unfortunately a bit slow, which is why this function appears slow if called many times.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

`readWindowsShortcut()`. `file.path()`.

Examples

```
# Default
print(file.path("foo", "bar", "..", "name")) # "foo/bar/../name"

# Shorten pathname, if possible
print(filePath("foo", "bar", "..", "name")) # "foo/name"
print(filePath("foo/bar/../name"))       # "foo/name"

# Recognize Windows Shortcut files along the path, cf. Unix soft links
filename <- system.file("data-ex/HISTORY.LNK", package="R.utils")
print(filename)
filename <- filePath(filename, expandLinks="relative")
print(filename)
```

FileProgressBar *A progress bar that sets the size of a file accordingly*

Description

Package: R.utils

Class FileProgressBar

[Object](#)

~~|

~~+--[ProgressBar](#)

~~~~~|

~~~~~+--FileProgressBar

Directly known subclasses:

public static class **FileProgressBar**

extends [ProgressBar](#)

Usage

```
FileProgressBar(pathname=NULL, ...)
```

Arguments

pathname The pathname of the output file.

... Other arguments accepted by the [ProgressBar](#) constructor.

Details

A progress bar that sets the size of a file accordingly. This class useful to check the progress of a batch job by just querying the size of a file, for instance, via ftp.

Fields and Methods

Methods:

[remove](#) Removes the progress file for a file progress bar.

[update](#) Updates file progress bar.

Methods inherited from [ProgressBar](#):

as.character, getBarString, increase, isDone, reset, setMaxValue, setProgress, setStepLength, setTicks,

setValue, update

Methods inherited from Object:

\$, \$<-, [[, [[<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

Examples

```
## Not run:

# Creates a progress bar (of length 100) that displays it self as a file.
pb <- FileProgressBar("~/progress.simulation")
reset(pb)
while (!isDone(pb)) {
  x <- rnorm(3e4)
  increase(pb)
  Sys.sleep(0.01)
}

## End(Not run)
```

| | |
|-----------------|--|
| finalizeSession | <i>Function to call for finalizing the R session</i> |
|-----------------|--|

Description

Function to call for finalizing the R session. When called, all registered "onSessionExit" hooks (functions) are called. To define such hooks, use the [onSessionExit\(\)](#) function.

This method should not be used by the user.

Usage

```
## Default S3 method:
finalizeSession(...)
```

Arguments

... Not used.

Value

Returns (invisibly) the hooks successfully called.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[onSessionExit\(\)](#).

findSourceTraceback *Finds all 'srcfile' objects generated by source() in all call frames*

Description

Finds all 'srcfile' objects generated by source() in all call frames. This makes it possible to find out which files are currently scripted by [source\(\)](#).

Usage

```
## Default S3 method:
findSourceTraceback(...)
```

Arguments

... Not used.

Value

Returns a named list of objects of class 'srcfile'. The names of the list entries corresponds to the 'filename' value of each corresponding 'srcfile' object. The returned list is empty if [source\(\)](#) was not called.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

Examples

```
# -----
# Create two R script files where one source():s the other
# and both lists the traceback of filenames source():d.
# -----
path <- tempdir();
pathnameA <- Arguments$getWritablePathname("foo.R", path=path);
pathnameB <- Arguments$getWritablePathname("bar.R", path=path);

code <- 'cat("BEGIN foo.R\n")';
code <- c(code, 'print(findSourceTraceback());');
code <- c(code, sprintf('source("%s");', pathnameB));
code <- c(code, 'cat("END foo.R\n")');
code <- paste(code, collapse="\n");
```

```

cat(file=pathnameA, code);

code <- 'cat("BEGIN bar.R\n")';
code <- c(code, 'print(findSourceTraceback());');
code <- c(code, 'cat("END bar.R\n")');
code <- paste(code, collapse="\n");
cat(file=pathnameB, code);

# -----
# Source the first file
# -----
source(pathnameA, echo=TRUE);

```

getAbsolutePath *Gets the absolute pathname string*

Description

Gets the absolute pathname string.

Usage

```

## Default S3 method:
getAbsolutePath(pathname, workDirectory=getwd(), expandTilde=FALSE, ...)

```

Arguments

| | |
|---------------|---|
| pathname | A character string of the pathname to be converted into an absolute pathname. |
| workDirectory | A character string of the current working directory. |
| expandTilde | If <code>TRUE</code> , tilde (~) is expanded to the corresponding directory, otherwise not. |
| ... | Not used. |

Details

This method will replace replicated slashes ('/') with a single one, except for the double forward slashes prefixing a Microsoft Windows UNC (Universal Naming Convention) pathname.

Value

Returns a [character](#) string of the absolute pathname.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[isAbsolutePath\(\)](#).

| | |
|-----------|---|
| getParent | <i>Gets the string of the parent specified by this pathname</i> |
|-----------|---|

Description

Gets the string of the parent specified by this pathname. This is basically, by default the string before the last path separator of the absolute pathname.

Usage

```
## Default S3 method:
getParent(pathname, depth=1, fsep=.Platform$file.sep, ...)
```

Arguments

| | |
|----------|--|
| pathname | A character string of the pathname to be checked. |
| depth | An integer specifying how many generations up the path should go. |
| fsep | A character string of the file separator. |
| ... | Not used. |

Value

Returns a **character** string if the parent exists, otherwise **NULL**.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

| | |
|-----------------|---|
| getRelativePath | <i>Gets the relative pathname relative to a directory</i> |
|-----------------|---|

Description

Gets the relative pathname relative to a directory.

Usage

```
## Default S3 method:
getRelativePath(pathname, relativeTo=getwd(), caseSensitive=NULL, ...)
```

Arguments

| | |
|---------------|--|
| pathname | A character string of the pathname to be converted into an relative pathname. |
| relativeTo | A character string of the reference pathname. |
| caseSensitive | If TRUE , the comparison is case sensitive, otherwise not. If NULL , it is decided from the relative path. |
| ... | Not used. |

Details

In case the two paths are on different file systems, for instance, `C:/foo/bar/` and `D:/foo/`, the method returns `pathname` as is.

Value

Returns a `character` string of the relative pathname.

Non-case sensitive comparison

If `caseSensitive == NULL`, the relative path is used to decide if the comparison should be done in a case-sensitive mode or not. The current check is if it is a Windows path or not, that is, if the relative path starts with a device letter, then the comparison is non-case sensitive.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[getAbsolutePath\(\)](#). [isAbsolutePath\(\)](#).

Examples

```
getRelativePath("foo", "foo")           # "."
getRelativePath("foo/bar", "foo")       # "bar"
getRelativePath("foo/bar", "foo/bar/yah") # ".."
getRelativePath("foo/bar/cool", "foo/bar/yah/sub/") # "../../cool"
getRelativePath.default("/foo/bar/", "/bar/foo/") # "../../foo/bar"

# Windows
getRelativePath("C:/foo/bar/", "C:/bar/") # "../foo/bar"
getRelativePath("C:/foo/bar/", "D:/bar/") # "C:/foo/bar"
```

GString

Character string with advanced substitutions

Description

Package: R.utils

Class GString

character

~~|

~~+--GString

Directly known subclasses:

```
public static class GString
  extends character
```

Usage

```
GString(..., sep="")
```

Arguments

```
...          one or more objects, to be coerced to character vectors.
sep          A character string to separate the terms.
```

Fields and Methods**Methods:**

| | |
|---------------------------------|--|
| <code>as.character</code> | Gets the processed character string. |
| <code>getBuiltinDate</code> | Gets the current date. |
| <code>getBuiltinDatetime</code> | Gets the current date and time. |
| <code>getBuiltinHostname</code> | Gets the hostname of the system running R. |
| <code>getBuiltinOs</code> | Gets the operating system of the running machine. |
| <code>getBuiltinPid</code> | Gets the process id of the current R session. |
| <code>getBuiltinRhome</code> | Gets the path where R is installed. |
| <code>getBuiltinRversion</code> | Gets the current R version. |
| <code>getBuiltinTime</code> | Gets the current time. |
| <code>getBuiltinUsername</code> | Gets the username of the user running R. |
| <code>getRaw</code> | Gets the unprocessed GString. |
| <code>getVariableValue</code> | Gets a variable value given a name and attributes. |
| <code>parse</code> | Parses a GString. |
| <code>print</code> | Prints the processed GString. |

Methods inherited from character:

all.equal, as.data.frame, as.Date, as.POSIXlt, as.raster, downloadFile, formula, getDLLRegister-
dRoutines, isOpen, toAsciiRegExprPattern, toFileListTree, uses

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

Examples

```
# -----
# First example
```

```

# -----
who <- "world"

# Compare this...
cat(as.character(GString("Hello ${who}\n")))

# ...to this.
cat(GString("Hello ${who}\n"))

# Escaping
cat(as.character(GString("Hello \${who}\n")))

# -----
# Looping over vectors
# -----
x <- 1:5
y <- c("hello", "world")
cat(as.character(GString("(x,y)=(${x},${y})")), sep=" ")
cat("\n")

cat(as.character(GString("(x,y)=(${x},${capitalize}{y})")), sep=" ")
cat("\n")

# -----
# Predefined ("builtin") variables
# -----
cat(as.character(GString("Hello ${username} on host ${hostname} running R v${rversion} in process #${pid} on ${os}")))

# Other built-in variables/functions...
cat(as.character(GString("Current date: ${date}\n")))
cat(as.character(GString("Current date: ${format='%d/%m/%y'}[date]\n")))
cat(as.character(GString("Current time: ${time}\n")))

# -----
# Evaluating inline R code
# -----
cat(as.character(GString("Simple calculation: 1+1=${'1+1'}\n")))
cat(as.character(GString("Alternative current date: ${'date()'}\n")))

# -----
# Function values
# -----
# Call function rnorm with arguments n=1, i.e. rnorm(n=1)
cat(as.character(GString("Random normal number: ${n=1}{rnorm}\n")))

# -----
# Global search-replace feature

```

```

# -----
# Replace all '-' with '.'
cat(as.character(GString("Current date: ${date/-/}.\n")))
# Another example
cat(as.character(GString("Escaped string: 12*12=${'12*12'/1/}\n")))

# -----
# Defining new "builtin" function values
# -----
# Define your own builtin variables (functions)
setMethodS3("getBuiltinAletter", "GString", function(object, ...) {
  base::letters[runif(1, min=1, max=length(base::letters))]
})

cat(as.character(GString("A letter: ${aletter}\n")))
cat(as.character(GString("Another letter: ${aletter}\n")))

# Another example
setMethodS3("getBuiltinGstring", "GString", function(object, ...) {
  # Return another GString.
  GString("${date} ${time}")
})

cat(as.character(GString("Advanced example: ${gstring}\n")))

# Advanced example
setMethodS3("getBuiltinRunif", "GString", function(object, n=1, min=0, max=1, ...) {
  formatC(runif(n=n, min=min, max=max), ...)
})

cat(as.character(GString("A random number: ${runif}\n")))
n <- 5
cat(as.character(GString("${n} random numbers: ")))
cat(as.character(GString("${n}=[n, format='f']{runif}")))
cat("\n")

# Advanced options.
# Options are parsed as if they are elements in a list, e.g.
# list(n=runif(n=1,min=1,max=5), format='f')
cat(as.character(GString("$Random number of numbers: ")))
cat(as.character(GString("${n=runif(n=1,min=1,max=5), format='f']{runif}")))
cat("\n")

```

Description

Gzip/Gunzip a file.

Usage

```
## Default S3 method:  
gzip(filename, destname=sprintf("%s.gz", filename), overwrite=FALSE, remove=TRUE, BFR.SIZE=1e+07, ...
```

Arguments

| | |
|-----------|--|
| filename | Pathname of input file. |
| destname | Pathname of output file. |
| overwrite | If the output file already exists, then if <code>overwrite</code> is <code>TRUE</code> the file is silently overwriting, otherwise an exception is thrown. |
| remove | If <code>TRUE</code> , the input file is removed afterward, otherwise not. |
| BFR.SIZE | The number of bytes read in each chunk. |
| ... | Not used. |

Details

Internally `gzfile()` (see [connections](#)) is used to read (write) chunks to (from) the gzip file. If the process is interrupted before completed, the partially written output file is automatically removed.

Value

Returns the number of (input) bytes read.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

Examples

```
cat(file="foo.txt", "Hello world!")  
gzip("foo.txt")  
print(file.info("foo.txt.gz"))  
gunzip("foo.txt.gz")  
print(file.info("foo.txt"))  
file.remove("foo.txt")
```

| | |
|----------------|--|
| hasUrlProtocol | <i>Checks if one or several pathnames has a URL protocol</i> |
|----------------|--|

Description

Checks if one or several pathnames has a URL protocol.

Usage

```
## Default S3 method:
hasUrlProtocol(pathname, ...)
```

Arguments

| | |
|----------|--------------------------------------|
| pathname | A character vector . |
| ... | Not used. |

Value

Returns a [logical vector](#).

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

| | |
|--------|--|
| hpaste | <i>Concatenating vectors into human-readable strings</i> |
|--------|--|

Description

Concatenating vectors into human-readable strings such as "1, 2, 3, ..., 10".

Usage

```
## Default S3 method:
hpaste(..., sep="", collapse=", ", lastCollapse=NULL, maxHead=if (missing(lastCollapse)) 3 else Inf, m
```

Arguments

| | |
|------------------------|---|
| ... | Arguments to be pasted. |
| sep | A character string used to concatenate the arguments in ..., if more than one. |
| collapse, lastCollapse | The character strings to collapse the elements together, where lastCollapse is specifying the collapse string used between the last two elements. If lastCollapse is NULL (default), it is corresponds to using the default collapse. |

maxHead, maxTail, abbreviate

Non-negative [integers](#) (also [Inf](#)) specifying the maximum number of elements of the beginning and then end of the vector to be outputted. If $n = \text{length}(x)$ is greater than $\text{maxHead} + \text{maxTail} + 1$, then x is truncated to consist of $x[1:\text{maxHead}]$, [abbreviate](#), and $x[(n - \text{maxTail} + 1):n]$.

Details

`hpaste(..., sep=" ", maxHead=Inf)` corresponds to `paste(..., sep=" ", collapse=" ")`.

Value

Returns a [character](#) string.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

Internally [paste\(\)](#) is used.

Examples

```
# Some vectors
x <- 1:6
y <- 10:1
z <- LETTERS[x]

# -----
# Abbreviation of output vector
# -----
printf("x = %s.\n", hpaste(x))
## x = 1, 2, 3, ..., 6.

printf("x = %s.\n", hpaste(x, maxHead=2))
## x = 1, 2, ..., 6.

printf("x = %s.\n", hpaste(x, maxHead=3) # Default
## x = 1, 2, 3, ..., 6.

# It will never output 1, 2, 3, 4, ..., 6
printf("x = %s.\n", hpaste(x, maxHead=4))
## x = 1, 2, 3, 4, 5 and 6.

# Showing the tail
printf("x = %s.\n", hpaste(x, maxHead=1, maxTail=2))
## x = 1, ..., 5, 6.

# Turning off abbreviation
printf("y = %s.\n", hpaste(y, maxHead=Inf))
```

```

## y = 10, 9, 8, 7, 6, 5, 4, 3, 2, 1

## ...or simply
printf("y = %s.\n", paste(y, collapse=" "))
## y = 10, 9, 8, 7, 6, 5, 4, 3, 2, 1

# -----
# Adding a special separator before the last element
# -----
# Change last separator
printf("x = %s.\n", hpaste(x, lastCollapse=" and "))
## x = 1, 2, 3, 4, 5 and 6.

# -----
# Backward compatibility with paste()
# -----
s1 <- hpaste(x, maxHead=Inf)
s2 <- paste(x, collapse=" ")
printf("s = %s.\n", s1);
stopifnot(identical(s1, s2))

s1 <- hpaste('<', x, '>', maxHead=Inf)
s2 <- paste('<', x, '>', sep="", collapse=" ")
printf("s = %s.\n", s1);
stopifnot(identical(s1, s2))

s1 <- hpaste(x, y, z, sep="/", maxHead=Inf)
s2 <- paste(x, y, z, sep="/", collapse=" ")
printf("s = %s.\n", s1);
stopifnot(identical(s1, s2))

s1 <- hpaste(x, collapse=NULL, maxHead=Inf)
s2 <- paste(x, collapse=NULL)
stopifnot(identical(s1, s2))

```

`inAnyInterval.numeric` *Checks if a set of values are inside one or more intervals*

Description

Checks if a set of values are inside one or more intervals.

Usage

```

## S3 method for class 'numeric'
inAnyInterval(...)

```

Arguments

... Arguments passed to `*mapToIntervals()`.

Value

Returns a `logical vector`.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

`mapToIntervals()`.

insert

Insert values to a vector at certain positions

Description

Insert values to a vector at certain positions.

Usage

```
## Default S3 method:  
insert(x, ats, values=NA, useNames=TRUE, ...)
```

Arguments

| | |
|-----------------------|--|
| <code>x</code> | The <code>vector</code> of data values. |
| <code>ats</code> | The indices of <code>x</code> where the values should be inserted. |
| <code>values</code> | A <code>list</code> or a <code>vector</code> of the values to be inserted. |
| <code>useNames</code> | If <code>FALSE</code> , the names attribute is dropped/ignored, otherwise not. Only applied if argument <code>x</code> is named. |
| ... | Not used. |

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

Examples

```

# Insert at first position
y <- c(a=1, b=2, c=3)
print(y)
x <- insert(y, ats=1, values=rep(NA,2))
Ex <- c(NA,NA,y)
print(x)
stopifnot(identical(x,Ex))

x <- insert(y, ats=1, values=rep(NA,2), useNames=FALSE)
print(x)

# Insert at last position (names of 'values' are ignored
# because input vector has not names)
x <- insert(1:3, ats=4, values=c(d=2, e=1))
Ex <- c(1:3,2,1)
print(x)
stopifnot(identical(x,Ex))

# Insert in the middle of a vector
x <- insert(c(1,3,2,1), ats=2, values=2)
print(x)
stopifnot(identical(as.double(x),as.double(Ex)))

# Insert multiple vectors at multiple indices at once
x0 <- c(1:4, 8:11, 13:15)

x <- insert(x0, at=c(5,9), values=list(5:7,12))
print(x)
Ex <- 1:max(x)
stopifnot(identical(as.double(x),as.double(Ex)))

x <- insert(x0, at=c(5,9,12), values=list(5:7,12,16:18))
print(x)
Ex <- 1:max(x)
stopifnot(identical(as.double(x),as.double(Ex)))

# Insert missing indices
Ex <- 1:20
missing <- setdiff(Ex, x0)
x <- x0
for (m in missing)
  x <- insert(x, ats=m, values=m)
print(x)
stopifnot(identical(as.double(x),as.double(Ex)))

```

`intervalsToSeq.matrix` *Generates a vector of indices from a matrix of intervals*

Description

Generates a vector of indices from a matrix of intervals.

Usage

```
## S3 method for class 'matrix'  
intervalsToSeq(fromTo, sort=FALSE, unique=FALSE, ...)
```

Arguments

| | |
|---------------------|---|
| <code>fromTo</code> | An Nx2 integer matrix . |
| <code>sort</code> | If TRUE , the returned indices are ordered. |
| <code>unique</code> | If TRUE , the returned indices are unique. |
| <code>...</code> | Not used. |

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[seqToIntervals\(\)](#).

Examples

```
## Not run: See example(seqToIntervals)
```

`intToBin` *Converts an integer to a binary/octal/hexadecimal number*

Description

Converts an integer to a binary/octal/hexadecimal number.

Usage

```
intToBin(x)  
intToOct(x)  
intToHex(x)
```

Arguments

x An *integer* to be converted.

Value

Returns a *character*.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

isAbsolutePath *Checks if this pathname is absolute*

Description

Checks if this pathname is absolute.

Usage

```
## Default S3 method:  
isAbsolutePath(pathname, ...)
```

Arguments

pathname A *character* string of the pathname to be checked.
... Not used.

Value

Returns a *TRUE* if the pathname is absolute, otherwise *FALSE*.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

| | |
|-------------|--|
| isDirectory | <i>Checks if the file specification is a directory</i> |
|-------------|--|

Description

Checks if the file specification is a directory.

Usage

```
## Default S3 method:
isDirectory(pathname, ...)
```

Arguments

| | |
|----------|---|
| pathname | A character string of the pathname to be checked. |
| ... | Not used. |

Value

Returns [TRUE](#) if the file specification is a directory, otherwise [FALSE](#) is returned.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

To check if it is a file see [isFile\(\)](#). Internally [file.info\(\)](#) is used.

| | |
|------------------|---|
| isEof.connection | <i>Checks if the current file position for a connection is at the 'End of File'</i> |
|------------------|---|

Description

Checks if the current file position for a connection is at the 'End of File'.

Usage

```
## S3 method for class 'connection'
isEof(con, ...)
```

Arguments

| | |
|-----|--------------------------------|
| con | A connection . |
| ... | Not used. |

Value

Returns a [logical](#).

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

For more information see [connection](#).

| | |
|--------|---|
| isFile | <i>Checks if the file specification is a file</i> |
|--------|---|

Description

Checks if the file specification is a file.

Usage

```
## Default S3 method:  
isFile(pathname, ...)
```

Arguments

| | |
|----------|---|
| pathname | A character string of the pathname to be checked. |
| ... | Not used. |

Value

Returns [TRUE](#) if the file specification is a file, otherwise [FALSE](#) is returned.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

To check if it is a directory see [isDirectory\(\)](#). Internally [file.info\(\)](#) is used.

isOpen.character *Checks if there is an open connection to a file*

Description

Checks if there is an open connection to a file.

Usage

```
## S3 method for class 'character'  
isOpen(pathname, rw=c("read", "write"), ...)
```

Arguments

| | |
|----------|--|
| pathname | An character string. |
| rw | A character vector . If "read", a file is considered to be open if there exist an open connection that can read from that file. If "write", a file is considered to be open if there exist an open connection that can write to that file. Both these values may be specified. |
| ... | Not used. |

Value

Returns [TRUE](#) if there exists a file [connection](#) that is open, otherwise [FALSE](#).

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

See `isOpen()` in [connections](#). [showConnections\(\)](#).

isPackageInstalled *Checks if a package is installed or not*

Description

Checks if a package is installed or not.

Usage

```
## Default S3 method:  
isPackageInstalled(package, ...)
```

Arguments

| | |
|---------|--------------------------|
| package | The name of the package. |
| ... | Not used. |

Value

Returns a [logical](#).

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[isPackageLoaded\(\)](#).

| | |
|-----------------|---|
| isPackageLoaded | <i>Checks if a package is loaded or not</i> |
|-----------------|---|

Description

Checks if a package is loaded or not. Note that, contrary to [require\(\)](#), this function does not load the package if not loaded.

Usage

```
## Default S3 method:  
isPackageLoaded(package, version=NULL, ...)
```

Arguments

| | |
|---------|--|
| package | The name of the package. |
| version | A character string specifying the version to test for. If <code>NULL</code> , any version is tested for. |
| ... | Not used. |

Value

Returns a [logical](#).

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

To check if a package is installed or not, see [isPackageInstalled\(\)](#).

| | |
|--------------|--|
| isReplicated | <i>Identifies all entries with replicated values</i> |
|--------------|--|

Description

Identifies all entries with replicated values, that is, with values that exist more than once.

Usage

```
isReplicated(x, ...)  
replicates(x, ...)
```

Arguments

`x` A [vector](#) of length `K`.
`...` Additional arguments passed to [duplicated\(\)](#).

Details

Let `reps <- isReplicated(x)`. Then it always holds that:

- `reps == rev(isReplicated(rev(x)))`
- `reps == duplicated(x) | duplicated(x, fromLast=TRUE)`
- `reps == !is.element(x, setdiff(x, unique(x[duplicated(x)])))`

Value

A [logical vector](#) of length `K`, where `TRUE` indicates that the value exists elsewhere, otherwise not.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

Internally [duplicated\(\)](#) is used. See also [isSingle\(\)](#).

Examples

```
x <- c(1,1,2,3,4,2,1)  
x <- base::letters[x]  
print(x)  
  
# Identify entries with replicated values  
reps <- isReplicated(x)  
print(x[reps])  
stopifnot(x[reps] == replicates(x))
```

```

# Identify entries with unique values
print(x[!reps])
stopifnot(x[!reps] == singles(x))

# - - - - -
# Validation
# - - - - -
x <- c(1,1,2,3,4,2,1)
x <- base::letters[x]
reps <- isReplicated(x)

stopifnot(all(table(x[reps]) > 1))
stopifnot(all(table(x[!reps]) == 1))
stopifnot(all(reps == rev(isReplicated(rev(x)))))
stopifnot(all(reps == duplicated(x) | duplicated(x, fromLast=TRUE)))
stopifnot(all(reps == !is.element(x, setdiff(x, unique(x[duplicated(x)]))))))
stopifnot(all(sort(c(singles(x), replicates(x))) == sort(x)))

# - - - - -
# Benchmarking singles()
# - - - - -
set.seed(0xBEEF)
n <- 1e6
x <- sample(1:(n/2), size=n, replace=TRUE)
t <- system.time({
  s <- isSingle(x)
})
print(sum(s))

t0 <- system.time({
  s0 <- !(x %in% x[duplicated(x)]);
})
print(t/t0)
stopifnot(all(s == s0))

```

isSingle

Identifies all entries that exists exactly ones

Description

Identifies all entries that exists exactly ones.

Usage

```

isSingle(x, ...)
singles(x, ...)

```

Arguments

x A [vector](#) of length K.
... Additional arguments passed to [isReplicated\(\)](#).

Value

A [logical vector](#) of length K, indicating whether the value is unique or not.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

Internally [isReplicated\(\)](#) is used.

| | |
|-------|---|
| isUrl | <i>Checks if one or several pathnames is URLs</i> |
|-------|---|

Description

Checks if one or several pathnames is URLs.

Usage

```
## Default S3 method:  
isUrl(pathname, ...)
```

Arguments

pathname A [character vector](#).
... Not used.

Value

Returns a [logical vector](#).

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

| | |
|--------|--|
| isZero | <i>Checks if a value is (close to) zero or not</i> |
|--------|--|

Description

Checks if a value (or a vector of values) is (close to) zero or not where "close" means if the absolute value is less than `neps*eps`. *Note that `x == 0` will not work in all cases.*

By default `eps` is the smallest possible floating point value that can be represented by the running machine, i.e. `.Machine$double.eps` and `neps` is one. By changing `neps` it is easy to adjust how close to zero "close" means without having to know the machine precision (or remembering how to get it).

Usage

```
## Default S3 method:  
isZero(x, neps=1, eps=.Machine$double.eps, ...)
```

Arguments

| | |
|-------------------|--|
| <code>x</code> | A vector of values. |
| <code>eps</code> | The smallest possible floating point. |
| <code>neps</code> | A scale factor of <code>eps</code> specifying how close to zero "close" means. If <code>eps</code> is the smallest value such that <code>1 + eps != 1</code> , i.e. <code>.Machine\$double.eps</code> , <code>neps</code> must be greater or equal to one. |
| <code>...</code> | Not used. |

Value

Returns a [logical vector](#) indicating if the elements are zero or not.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[all.equal\(\)](#). [Comparison](#). [.Machine](#).

Examples

```
x <- 0  
print(x == 0)      # TRUE  
print(isZero(x))  # TRUE  
  
x <- 1  
print(x == 0)      # FALSE  
print(isZero(x))  # FALSE
```

```

x <- .Machine$double.eps
print(x == 0)      # FALSE
print(isZero(x))  # FALSE

x <- 0.9*.Machine$double.eps
print(x == 0)      # FALSE
print(isZero(x))  # TRUE

# From help(Comparisons)
x1 <- 0.5 - 0.3
x2 <- 0.3 - 0.1
print(x1 - x2)
print(x1 == x2)           # FALSE on most machines
print(identical(all.equal(x1, x2), TRUE)) # TRUE everywhere
print(isZero(x1-x2))      # TRUE everywhere

```

 Java

Static class for Java related methods

Description

Package: R.utils

Class Java

Object

```

~~|
~~+--Java

```

Directly known subclasses:

```

public static class Java
extends Object

```

Static class that provides methods for reading and writing Java data types. Currently the following data types are supported: byte, short and int. R character strings can be written as UTF-8 formatted strings, which can be read by Java. Currently on Java String's that contain ASCII characters can be imported into R. The reason for this is that other characters are translated into non-eight bits data, e.g. 16- and 24-bits, which the readChar() method currently does not support.

Furthermore, the Java class defines some static constants describing the minimum and maximum value of some of the common Java data types: BYTE . MIN, BYTE . MAX SHORT . MIN, SHORT . MAX INT . MIN, INT . MAX LONG . MIN, and LONG . MAX.

Usage

```

Java()

```

Fields and Methods

Methods:

| | |
|-------------------------|--|
| <code>asByte</code> | Converts a numeric to a Java byte. |
| <code>asInt</code> | Converts an numeric to a Java integer. |
| <code>asLong</code> | Converts a numeric to a Java long. |
| <code>asShort</code> | Converts a numeric to a Java short. |
| <code>readByte</code> | Reads a Java formatted byte (8 bits) from a connection. |
| <code>readInt</code> | Reads a Java formatted int (32 bits) from a connection. |
| <code>readShort</code> | Reads a Java formatted short (16 bits) from a connection. |
| <code>readUTF</code> | Reads a Java (UTF-8) formatted string from a connection. |
| <code>writeByte</code> | Writes a byte (8 bits) to a connection in Java format. |
| <code>writeInt</code> | Writes a integer (32 bits) to a connection in Java format. |
| <code>writeShort</code> | Writes a short (16 bits) to a connection in Java format. |
| <code>writeUTF</code> | Writes a string to a connection in Java format (UTF-8). |

Methods inherited from Object:

`$`, `$<`, `[]`, `[[<`, `as.character`, `attach`, `attachLocally`, `clearCache`, `clearLookupCache`, `clone`, `detach`, `equals`, `extend`, `finalize`, `gc`, `getEnvironment`, `getFieldModifier`, `getFieldModifiers`, `getFields`, `getInstantiationTime`, `getStaticInstance`, `hasField`, `hashCode`, `ll`, `load`, `objectSize`, `print`, `registerFinalizer`, `save`

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

Examples

```
pathname <- tempfile()

# Open the temporary file for writing
out <- file(pathname, open="wb")
b <- -128:127
Java$writeByte(out, b)
s <- -32768:32767
Java$writeShort(out, s)
i <- c(-2147483648, -2147483647, -1, 0, +1, 2147483646, 2147483647);
Java$writeInt(out, i)
str <- "This R string was written (using the UTF-8 format) using\nthe static methods of the Java class in the R.io pa
Java$writeUTF(out, str)
close(out)

# Open the temporary file for reading
inn <- file(pathname, open="rb")

bfr <- Java$readByte(inn, n=length(b))
cat("Read ", length(bfr), " bytes.\n", sep="")
if (!identical(bfr, b))
  throw("Failed to read the same data that was written.")
```

```
bfr <- Java$readShort(inn, n=length(s))
cat("Read ", length(bfr), " shorts.\n", sep="")
if (!identical(bfr, s))
  throw("Failed to read the same data that was written.")

bfr <- Java$readInt(inn, n=length(i))
cat("Read ", length(bfr), " ints.\n", sep="")
if (!identical(bfr, i))
  throw("Failed to read the same data that was written.")

bfr <- Java$readUTF(inn)
cat("Read ", nchar(bfr), " UTF characters:\n", "'", bfr, "'\n", sep="")

close(inn)

file.remove(pathname)
```

jpeg2

A JPEG device for Bitmap Files via GhostScript

Description

A JPEG device for Bitmap Files via GhostScript.

Usage

```
jpeg2(filename, width=480, height=480, res=144, type="jpeg", ...)
```

Arguments

| | |
|---------------|--|
| filename | The name of the file to be produced. |
| width, height | The width and height (in pixels) of the result image. |
| res | The resolution of the image. |
| type | The output type. See dev2bitmap for details. |
| ... | Additional arguments passed to <code>bitmap()</code> . |

Value

Returns nothing.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

| | |
|--------------|--|
| lastModified | <i>Gets the time when the file was last modified</i> |
|--------------|--|

Description

Gets the time when the file was last modified. The time is returned as a POSIXct object.

Usage

```
## Default S3 method:
lastModified(pathname, ...)
```

Arguments

| | |
|----------|--|
| pathname | A character string of the pathname to be checked. |
| ... | Not used. |

Value

Returns POSIXct object specifying when the file was last modified. If the file does not exist or it is a directory, 0 is returned.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

Internally `file.info()` is used.

| | |
|-----------|----------------------------|
| LComments | <i>The LComments class</i> |
|-----------|----------------------------|

Description

Package: R.utils

Class LComments

Object

```
~~|
~~+--SmartComments
~~~~~|
~~~~~+--VComments
~~~~~|
~~~~~+--LComments
```

Directly known subclasses:

```
public static class LComments  
extends VComments
```

The LComments class.

This class, is almost identical to the super class, except that the constructor has different defaults.

Usage

```
LComments(letter="L", verboseName="log", ...)
```

Arguments

| | |
|-------------|---------------------------------|
| letter | The smart letter. |
| verboseName | The name of the verbose object. |
| ... | Not used. |

Fields and Methods**Methods:**

No methods defined.

Methods inherited from VComments:

convertComment, reset, validate

Methods inherited from SmartComments:

compile, convertComment, parse, reset, validate

Methods inherited from Object:

\$, \$<-, [], [[<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

listDirectory

Gets the file names in the directory

Description

Gets the file names in the directory.

Contrary to `list.files()`, this method guarantees to work recursively. Moreover, when subdirectories are processed recursively, directory names are also returned.

Usage

```
## Default S3 method:  
listDirectory(pathname, pattern=NULL, recursive=FALSE, allNames=FALSE, fullNames=FALSE, ...)
```

Arguments

| | |
|-----------|---|
| pathname | A pathname to be listed. |
| pattern | A character string of the filename pattern passed. See list.files() for more details. |
| recursive | If TRUE , subdirectories are recursively processed, otherwise not. |
| allNames | If TRUE , also files starting with a period are returned. |
| fullNames | If TRUE , the full path names are returned. |
| ... | Not used. |

Value

Returns a [vector](#) of file names.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

Internally [list.files\(\)](#) is used.

| | |
|------------|--|
| loadObject | <i>Method to load object from a file or a connection</i> |
|------------|--|

Description

Method to load object from a file or a connection, which previously have been saved using [saveObject\(\)](#).

Usage

```
## Default S3 method:  
loadObject(file, path=NULL, ...)
```

Arguments

| | |
|------|---|
| file | A filename or connection to read the object from. |
| path | The path where the file exists. |
| ... | Not used. |

Details

The main difference from this method and `load()` in the **base** package, is that this one returns the object read rather than storing it in the global environment by its default name. This makes it possible to load objects back using any variable name.

Value

Returns the save object.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

`saveObject()` to save an object to file. Internally `load()` is used. See also `loadToEnv()`.

mapToIntervals.numeric

Maps values to intervals

Description

Maps values to intervals by returning an index **vector** specifying the (first) interval that each value maps to, if any.

Usage

```
## S3 method for class 'numeric'
mapToIntervals(x, intervals, includeLower=TRUE, includeUpper=TRUE, ...)
```

Arguments

| | |
|---|---|
| <code>x</code> | A numeric vector of K values to be matched. |
| <code>intervals</code> | The N intervals to be matched against. If an Nx2 numeric matrix , the first column should be the lower bounds and the second column the upper bounds of each interval. If a numeric vector of length 2N, each consecutive pair should be the lower and upper bounds of an interval. |
| <code>includeLower, includeUpper</code> | If TRUE , the lower (upper) bound of <i>each</i> interval is included in the test, otherwise not. |
| <code>...</code> | Not used. |

Value

Returns an **integer vector** of length K. Values that do not map to any interval have return value **NA**.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[inAnyInterval\(\)](#). [match\(\)](#). [findInterval\(\)](#). [cut\(\)](#).

mergeIntervals.numeric

Merges intervals

Description

Merges intervals by returning an index [vector](#) specifying the (first) interval that each value maps to, if any.

Usage

```
## S3 method for class 'numeric'  
mergeIntervals(intervals, ...)
```

Arguments

| | |
|------------------------|--|
| <code>intervals</code> | The N intervals to be merged. If an Nx2 numeric matrix , the first column should be the lower bounds and the second column the upper bounds of each interval. If a numeric vector of length 2N, each consecutive pair should be the lower and upper bounds of an interval. |
| <code>...</code> | Not used. |

Details

The upper and lower bounds are considered to be inclusive, that is, all intervals are interpreted to be of form [a,b]. There is currently no way to specify intervals with open bounds, e.g. (a,b).

Furthermore, the bounds are currently treated as real values. For instance, merging [0,1] and [2,3] will return the same intervals. Note, if integer intervals were treated specially, we would merge these intervals to integer interval [0,3] == {0,1,2,3}.

Value

Returns a [matrix](#) (or a [vector](#)) of M intervals, where M <= N. The intervals are ordered by their lower bounds. The @mode of the returned intervals is the same as the mode of the input intervals.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[inAnyInterval\(\). match\(\)](#).

| | |
|----------|---|
| makedirs | <i>Creates a directory including any necessary but nonexistent parent directories</i> |
|----------|---|

Description

Creates a directory including any necessary but nonexistent parent directories.

Usage

```
## Default S3 method:  
makedirs(pathname, ...)
```

Arguments

| | |
|----------|---|
| pathname | A character string of the pathname to be checked. |
| ... | Not used. |

Value

Returns [TRUE](#) if the directory was successfully created, otherwise [FALSE](#). Note that if the directory already exists, [FALSE](#) is returned.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

Internally [dir.create\(\)](#) is used.

NullVerbose *A Verbose class ignoring everything*

Description

Package: R.utils

Class NullVerbose

[Object](#)

~~|

~~+--[Verbose](#)

~~~~~|

~~~~~+--NullVerbose

Directly known subclasses:

public static class **NullVerbose**

extends [Verbose](#)

A Verbose class ignoring everything.

Usage

NullVerbose(...)

Arguments

... Ignored.

Fields and Methods

Methods:

| | |
|---------------------------|---|
| cat | - |
| enter | - |
| evaluate | - |
| exit | - |
| header | - |
| isOn | Checks if the output is on. |
| isVisible | Checks if a certain verbose level will be shown or not. |
| newline | - |
| print | - |
| printf | - |
| ruler | - |

```

str          -
summary     -
writeRaw    All output methods.

```

Methods inherited from Verbose:

as.character, as.double, as.logical, capture, cat, enter, equals, evaluate, exit, getThreshold, getTimestampFormat, header, isOn, isVisible, less, more, newline, off, on, popState, print, printf, pushState, ruler, setDefaultLevel, setThreshold, setTimestampFormat, str, summary, timestamp, timestampOff, timestampOn, warnings, writeRaw

Methods inherited from Object:

\$, \$<-, [[, [[<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

Examples

```

verbose <- Verbose()
cat(verbose, "A verbose messages")

verbose <- NullVerbose()
cat(verbose, "A verbose messages") # Ignored

```

| | |
|------------------|---|
| onGarbageCollect | <i>Registers a function to be called when the R garbage collector is (detected to be) running</i> |
|------------------|---|

Description

Registers a function to be called when the R garbage collector is (detected to be) running.

Usage

```

## Default S3 method:
onGarbageCollect(fcn, action=c("prepend", "append", "replace"), ...)

```

Arguments

```

fcn          A function to be called without argument.
action       A character string specifying how the hook function is added to list of hooks.
...         Not used.

```

Value

Returns (invisibly) the hooks successfully called.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

Examples

```
## Not run:
onGarbageCollect(function(...) {
  cat("The R garbage collector is running!\n");
})

## End(Not run)
```

| | |
|---------------|--|
| onSessionExit | <i>Registers a function to be called when the R session finishes</i> |
|---------------|--|

Description

Registers a function to be called when the R session finishes.

Usage

```
## Default S3 method:
onSessionExit(fcn, action=c("prepend", "append", "replace"), ...)
```

Arguments

| | |
|--------|--|
| fcn | A function to be called without argument. |
| action | A character string specifying how the hook function is added to list of hooks. |
| ... | Not used. |

Details

Functions registered this way are called when [finalizeSession\(\)](#) is called. Moreover, when this package is loaded, the `.Last()` function is modified such that `finalizeSession()` is called. However, note that `.Last()` is *not* guaranteed to be called when the R session finished. For instance, the user may quit R by calling `quit(callLast=FALSE)`. Moreover, when R is run in batch mode, `.Last()` is never called.

Value

Returns (invisibly) the hooks successfully called.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[.Last\(\)](#), [finalizeSession\(\)](#).

Examples

```
## Not run:
onSessionExit(function(...) {
  cat("Bye bye world!\n");
})

quit()

## End(Not run)
```

Options

The Options class

Description

Package: R.utils

Class Options**Object**

~~|
~~+--Options

Directly known subclasses:

[Settings](#)

```
public static class Options
extends Object
```

A class to set and get either options stored in a [list](#) tree structure.

Each option has a pathname. The format of a pathname is similar to a (Unix) filesystem pathname, e.g. "graphics/cex". See examples for more details.

Usage

```
Options(options=list(), ...)
```

Arguments

options A tree `list` structure of options.
 ... Not used.

Details

Note, this class and its methods do *not* operate on the global options structure defined in R (`options`).

Value

The constructor returns an Options object.

Fields and Methods**Methods:**

| | |
|---------------------------|---|
| <code>as.character</code> | Returns a character string version of this object. |
| <code>as.list</code> | Gets a list representation of the options. |
| <code>equals</code> | Checks if this object is equal to another Options object. |
| <code>getLeaves</code> | Gets all (non-list) options in a flat list. |
| <code>getOption</code> | Gets an option. |
| <code>hasOption</code> | Checks if an option exists. |
| <code>names</code> | Gets the full pathname of all (non-list) options. |
| <code>nbrOfOptions</code> | Gets the number of options set. |
| <code>setOption</code> | Sets an option. |
| <code>str</code> | Prints the structure of the options. |

Methods inherited from Object:

`$`, `$<-`, `[[`, `[[<-`, `as.character`, `attach`, `attachLocally`, `clearCache`, `clearLookupCache`, `clone`, `detach`, `equals`, `extend`, `finalize`, `gc`, `getEnvironment`, `getFieldModifier`, `getFieldModifiers`, `getFields`, `getInstantiationTime`, `getStaticInstance`, `hasField`, `hashCode`, `ll`, `load`, `objectSize`, `print`, `registerFinalizer`, `save`

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

Examples

```
local <- Options()

# Query a missing option
cex <- getOption(local, "graphics/cex")
cat("graphics/cex =", cex, "\n") # Returns NULL

# Query a missing option with default value
cex <- getOption(local, "graphics/cex", defaultValue=1)
cat("graphics/cex =", cex, "\n") # Returns NULL
```

```

# Set option and get previous value
oldCex <- setOption(local, "graphics/cex", 2)
cat("previous graphics/cex =", oldCex, "\n") # Returns NULL

# Set option again and get previous value
oldCex <- setOption(local, "graphics/cex", 3)
cat("previous graphics/cex =", oldCex, "\n") # Returns 2

# Query a missing option with default value, which is ignored
cex <- getOption(local, "graphics/cex", defaultValue=1)
cat("graphics/cex =", cex, "\n") # Returns 3

# Query multiple options with multiple default values
multi <- getOption(local, c("graphics/cex", "graphics/pch"), c(1,2))
print(multi);

# Check existence of multiple options
has <- hasOption(local, c("graphics/cex", "graphics/pch"))
print(has);

# Get a subtree of options
graphics <- getOption(local, "graphics")
print(graphics)

# Get the complete tree of options
all <- getOption(local)
print(all)

```

patchCode

Patches installed and loaded packages and more

Description

Patches installed and loaded packages and more.

Usage

```

## Default S3 method:
patchCode(paths=NULL, recursive=TRUE, suppressWarnings=TRUE, knownExtensions=c("R", "r", "S", "s"), ve

```

Arguments

paths The path to the directory (and subdirectories) which contains source code that will patch loaded packages. If `NULL`, the patch path is given by the option `R_PATCHES`, If the latter is not set, the system environment with the same name is used. If neither is given, then `~/R-patches/` is used.

| | |
|------------------|---|
| recursive | If TRUE , source code in subdirectories will also get loaded. |
| suppressWarnings | If TRUE , warnings will be suppressed, otherwise not. |
| knownExtensions | A character vector of filename extensions used to identify source code files. All other files are ignored. |
| verbose | If TRUE , extra information is printed while patching, otherwise not. |
| ... | Not used. |

Details

The method will look for source code files (recursively or not) that match known filename extensions. Each found source code file is then `source()`d.

If the search is recursive, subdirectories are entered if and only if either (1) the name of the subdirectory is the same as a *loaded* (and installed) package, or (2) if there is no installed package with that name. The latter allows common code to be organized in directories although it is still not assigned to packages.

Each of the directories given by argument paths will be processed one by one. This makes it possible to have more than one file tree containing patches.

To set an options, see `options()`. To set a system environment, see `Sys.setenv()`. The character `;` is interpreted as a separator. Due to incompatibility with Windows pathnames, `:` is *not* a valid separator.

Value

Returns (invisibly) the number of files sourced.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

`source()`. `library()`. `relibrary()`.

Examples

```
## Not run:
# Patch all source code files in the current directory
patchCode(".")

# Patch all source code files in R_PATCHES
options("R_PATCHES"="~/R-patches/")
# alternatively, Sys.setenv("R_PATCHES"="~/R-patches/")
patchCode()

## End(Not run)
```

png2 *A PNG device for Bitmap Files via GhostScript*

Description

A PNG device for Bitmap Files via GhostScript.

Usage

```
png2(filename, width=480, height=480, res=144, type="png256", ...)
```

Arguments

| | |
|---------------|--|
| filename | The name of the file to be produced. |
| width, height | The width and height (in pixels) of the result image. |
| res | The resolution of the image. |
| type | The output type. See dev2bitmap for details. |
| ... | Additional arguments passed to <code>bitmap()</code> . |

Value

Returns nothing.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

popBackupFile *Drops a backup suffix from the backup pathname*

Description

Drops a backup suffix from the backup pathname and, by default, restores an existing backup file accordingly by renaming it.

Usage

```
## Default S3 method:
popBackupFile(filename, path=NULL, suffix=".bak", isFile=TRUE, onMissing=c("ignore", "error"), drop=T)
```

Arguments

| | |
|-----------|--|
| filename | The filename of the backup file. |
| path | The path of the file. |
| suffix | The suffix of the filename to be dropped. |
| isFile | If TRUE , the backup file must exist and will be renamed. If FALSE , it is only the pathname string that will be modified. For details, see below. |
| onMissing | A character string specifying what to do if the backup file does not exist. |
| drop | If TRUE , the backup file will be dropped in case the original file already exists or was successfully restored. |
| ... | Not used. |
| verbose | A logical or Verbose . |

Value

Returns the pathname with the backup suffix dropped.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

See [pushBackupFile\(\)](#) for more details and an example.

popTemporaryFile *Drops a temporary suffix from the temporary pathname*

Description

Drops a temporary suffix from the temporary pathname and, by default, renames an existing temporary file accordingly.

Usage

```
## Default S3 method:
popTemporaryFile(filename, path=NULL, suffix=".tmp", isFile=TRUE, ..., verbose=FALSE)
```

Arguments

| | |
|----------|---|
| filename | The filename of the temporary file. |
| path | The path of the temporary file. |
| suffix | The suffix of the temporary filename to be dropped. |
| isFile | If TRUE , the temporary file must exist and will be renamed. If FALSE , it is only the pathname string that will be modified. For details, see below. |
| ... | Not used. |
| verbose | A logical or Verbose . |

Details

If `isFile` is `FALSE`, the pathname where the suffix of the temporary pathname has been dropped is returned. If `isFile` is `TRUE`, the temporary file is renamed. Then, if the temporary file does not exist or it was not successfully renamed, an exception is thrown.

Value

Returns the pathname with the temporary suffix dropped.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

See `pushTemporaryFile()` for more details and an example.

| | |
|--------|---------------------------------|
| printf | <i>C-style formatted output</i> |
|--------|---------------------------------|

Description

C-style formatted output.

Usage

```
## Default S3 method:  
printf(fmt, ..., sep="", file="")
```

Arguments

| | |
|-------------------|---|
| <code>fmt</code> | A character vector of format strings. See same argument for <code>sprintf()</code> . |
| <code>...</code> | Additional arguments <code>sprintf()</code> . |
| <code>sep</code> | A character vector of strings to append after each element. |
| <code>file</code> | A connection , or a character of a file to print to. See same argument for <code>cat()</code> . |

Value

Returns nothing.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

For C-style formatting of [character](#) strings, see [sprintf\(\)](#).

Examples

```
cat("Hello world\n")
printf("Hello world\n")

x <- 1.23
cat(sprintf("x=%.2f\n", x))
printf("x=%.2f\n", x)

y <- 4.56
cat(sprintf(c("x=%.2f\n", "y=%.2f\n"), c(x,y)), sep="")
printf(c("x=%.2f\n", "y=%.2f\n"), c(x,y))
```

ProgressBar

Provides text based counting progress bar

Description

Package: R.utils

Class ProgressBar

[Object](#)

~~|

~~+--ProgressBar

Directly known subclasses:

[FileProgressBar](#)

public static class **ProgressBar**

extends [Object](#)

Usage

```
ProgressBar(max=100, ticks=10, stepLength=1, newlineWhenDone=TRUE)
```

Arguments

max The maximum number of steps.

ticks Put visual "ticks" every ticks step.

stepLength The default length for each increase.

newlineWhenDone

If **TRUE**, a newline is outputted when bar is updated, when done, otherwise not.

Fields and Methods**Methods:**

| | |
|----------------------------|--|
| <code>as.character</code> | Gets a string description of the progress bar. |
| <code>getBarString</code> | Gets the progress bar string to be displayed. |
| <code>increase</code> | Increases (steps) progress bar. |
| <code>isDone</code> | Checks if progress bar is completed. |
| <code>reset</code> | Reset progress bar. |
| <code>setMaxValue</code> | Sets maximum value. |
| <code>setProgress</code> | Sets current progress. |
| <code>setStepLength</code> | Sets default step length. |
| <code>setTicks</code> | Sets values for which ticks should be visible. |
| <code>setValue</code> | Sets current value. |
| <code>update</code> | Updates progress bar. |

Methods inherited from Object:

\$. \$<-, [[, [[<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

Examples

```
# A progress bar with default step length one.
pb <- ProgressBar(max=42)
reset(pb)
while (!isDone(pb)) {
  x <- rnorm(3e4)
  increase(pb)
  Sys.sleep(0.02)
}
cat("\n")

# A "faster" progress bar with default step length 1.4.
pb <- ProgressBar(max=42, stepLength=1.4)
reset(pb)
while (!isDone(pb)) {
  x <- rnorm(3e4)
  increase(pb)
  Sys.sleep(0.02)
}

cat("\n")
```

| | |
|----------------|--|
| pushBackupFile | <i>Appends a backup suffix to the pathname</i> |
|----------------|--|

Description

Appends a backup suffix to the pathname and, optionally, renames an existing file accordingly.

In combination with `popBackupFile()`, this method is useful for creating a backup of a file and restoring it.

Usage

```
## Default S3 method:  
pushBackupFile(filename, path=NULL, suffix=".bak", isFile=TRUE, onMissing=c("ignore", "error"), copy=
```

Arguments

| | |
|-----------|--|
| filename | The filename of the file to backup. |
| path | The path of the file. |
| suffix | The suffix to be appended. |
| isFile | If <code>TRUE</code> , the file must exist and will be renamed on the file system. If <code>FALSE</code> , it is only the pathname string that will be modified. For details, see below. |
| onMissing | A <code>character</code> string specifying what to do if the file does not exist. |
| copy | If <code>TRUE</code> , an existing original file remains after creating the backup copy, otherwise it is dropped. |
| overwrite | If <code>TRUE</code> , any existing backup files are overwritten, otherwise an exception is thrown. |
| ... | Not used. |
| verbose | A <code>logical</code> or <code>Verbose</code> . |

Value

Returns the pathname with the suffix appended.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

`popBackupFile()`.

Examples

```

# Create a file
pathname <- "foobar.txt";
cat(file=pathname, "File v1\n");

# -----
# (a) Backup and restore a file
# -----
# Turn it into a backup file
pathnameB <- pushBackupFile(pathname, verbose=TRUE);
print(pathnameB);

# Restore main file from backup
pathnameR <- popBackupFile(pathnameB, verbose=TRUE);
print(pathnameR);

# -----
# (b) Backup, create a new file and frop backup file
# -----
# Turn it into a backup file
pathnameB <- pushBackupFile(pathname, verbose=TRUE);
print(pathnameB);

# Create a new file
cat(file=pathname, "File v2\n");

# Drop backup because a new main file was successfully created
pathnameR <- popBackupFile(pathnameB, verbose=TRUE);
print(pathnameR);

```

pushTemporaryFile *Appends a temporary suffix to the pathname*

Description

Appends a temporary suffix to the pathname and, optionally, renames an existing file accordingly.

In combination with [popTemporaryFile\(\)](#), this method is useful for creating a file/writing data to file *atomically*, by first writing to a temporary file which is the renamed. If for some reason the generation of the file was interrupted, for instance by a user interrupt or a power failure, then it is only the temporary file that is incomplete.

Usage

```

## Default S3 method:
pushTemporaryFile(filename, path=NULL, suffix=".tmp", isFile=FALSE, ..., verbose=FALSE)

```

Arguments

| | |
|----------|--|
| filename | The filename of the file. |
| path | The path of the file. |
| suffix | The suffix to be appended. |
| isFile | If TRUE , the file must exist and will be renamed on the file system. If FALSE , it is only the pathname string that will be modified. For details, see below. |
| ... | Not used. |
| verbose | A logical or Verbose . |

Details

If `isFile` is **FALSE**, the pathname where the suffix of the temporary pathname has been added is returned. If `isFile` is **TRUE**, the file is also renamed. Then, if the file does not exist or it was not successfully renamed, an exception is thrown.

Value

Returns the pathname with the suffix appended.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[popTemporaryFile\(\)](#).

Examples

```
createAtomically <- function(pathname, ...) {
  cat("Pathname: ", pathname, "\n", sep="");

  # Generate a file atomically, i.e. the file will either be
  # complete or not created at all. If interrupted while
  # writing, only a temporary file will exist/remain.
  pathnameT <- pushTemporaryFile(pathname);
  cat("Temporary pathname: ", pathnameT, "\n", sep="");

  cat(file=pathnameT, "This file was created atomically:\n");
  for (kk in 1:10) {
    cat(file=pathnameT, kk, "\n", append=TRUE);
    Sys.sleep(0.1);
  }
  cat(file=pathnameT, "END OF FILE\n", append=TRUE);

  # Rename the temporary file
  pathname <- popTemporaryFile(pathnameT);
}
```

```
    pathname;
} # createAtomically()

pathname <- tempfile();

tryCatch({
  # Try to interrupt the process while writing...
  pathname <- createAtomically(pathname);
}, interrupt=function(intr) {
  str(intr);
})

# ...and this will throw an exception
bfr <- readLines(pathname);
cat(bfr, sep="\n");
```

| | |
|----------------|--|
| queryRCmdCheck | <i>Gets the on R CMD check if the current R session was launched by it</i> |
|----------------|--|

Description

Gets the on R CMD check if the current R session was launched by it.

Usage

```
queryRCmdCheck(...)
```

Arguments

... Not used.

Value

Returns `character` string "checkingTests" if 'R CMD check' runs one of the package tests, and "checkingExamples" if it runs one of the package examples. If the current R session was not launched by 'R CMD check', then "notRunning" is returned.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

Examples

```
status <- queryRCmdCheck()
if (status != "notRunning") {
  cat("The current R session was launched by R CMD check. Status: ", status, "\n")
} else {
  cat("The current R session was not launched by R CMD check.\n")
}
```

```

}

# Display how R was launched
print(base::commandArgs())

# Display loaded packages etc.
print(search())

# Display current working directory
print(getwd())

```

| | |
|------------------|---|
| readBinFragments | <i>Reads binary data from disjoint sections of a connection or a file</i> |
|------------------|---|

Description

Reads binary data from disjoint sections of a connection or a file.

Usage

```
## Default S3 method:
readBinFragments(con, what, idxs=1, origin=c("current", "start"), size=NA, ..., verbose=FALSE)
```

Arguments

| | |
|---------|--|
| con | A connection or the pathname of an existing file. |
| what | A character string or an object specifying the the data type (mode()) to be read. |
| idxs | A vector of (non-duplicated) indices or a Nx2 matrix of N from-to index intervals specifying the elements to be read. Positions are either relative to the start or the current location of the file/connection as given by argument origin. |
| origin | A character string specify whether the indices in argument idxs are relative to the "start" or the "current" position of the file/connection. |
| size | The size of the data type to be read. If NA , the natural size of the data type is used. |
| ... | Additional arguments passed to readBin() . |
| verbose | A logical or a Verbose object. |

Value

Returns a [vector](#) of the requested [mode\(\)](#).

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[writeBinFragments\(\)](#).

Examples

```
# -----
# Create a data file
# -----
data <- 1:255
size <- 2
pathname <- tempfile("exampleReadBinFragments")
writeBin(con=pathname, data, size=size)

# -----
# Read and write using index vectors
# -----
cat("Read file...\n")
# Read every 16:th byte in the file
idxs <- seq(from=1, to=255, by=16)
x <- readBinFragments(pathname, what="integer", size=size, signed=FALSE, idxs=idxs)
stopifnot(identical(x, data[idxs]))
print(x)
# Read every 16:th byte in a connection starting with the 6th.
idxs <- idxs + 5L;
x <- readBinFragments(pathname, what="integer", size=size, signed=FALSE, idxs=idxs)
stopifnot(identical(x, data[idxs]))
print(x)
cat("Read file...done\n")

cat("Write file...\n")
# Update every 16:th byte in the file
idxs <- seq(from=1, to=255, by=16)
x0 <- data[idxs]
writeBinFragments(pathname, idxs=idxs, rev(x0), size=size)
x <- readBinFragments(pathname, what="integer", size=size, signed=FALSE, idxs=idxs)
print(x)
stopifnot(identical(rev(x0), x))

# Update every 16:th byte in the file
idxs <- seq(from=1, to=255, by=16)
writeBinFragments(pathname, idxs=idxs, rev(x), size=size)
x <- readBinFragments(pathname, what="integer", size=size, signed=FALSE, idxs=idxs)
print(x)
stopifnot(identical(x0, x))

# Assert everything is as expected
# Read the complete file
x <- readBin(pathname, what="integer", size=size, signed=FALSE, n=length(data))
stopifnot(identical(x, data))
cat("Write file...done\n")
```

```

# -----
# Ditto but via a connection
# -----
cat("Read connection...\n")
# Read every 16:th byte in a connection
con <- file(pathname, open="rb")
idxs <- seq(from=1, to=255, by=16)
x <- readBinFragments(con, what="integer", size=size, signed=FALSE, idxs=idxs)
stopifnot(identical(x, data[idxs]))
print(x)

# Read every 16:th byte in a connection starting with the 6th.
idxs <- idxs + 5L;
x <- readBinFragments(con, what="integer", size=size, signed=FALSE, idxs=idxs, origin="start")
stopifnot(identical(x, data[idxs]))
print(x)
close(con)
cat("Read connection...done\n")

# Update every 16:th byte in a connection
cat("Write connection...\n")
con <- file(pathname, open="r+b")
idxs <- seq(from=1, to=255, by=16)
x0 <- data[idxs]
writeBinFragments(pathname, idxs=idxs, rev(x0), size=size)
x <- readBinFragments(pathname, what="integer", size=size, signed=FALSE, idxs=idxs)
print(x)
stopifnot(identical(rev(x0), x))

# Update every 16:th byte in the file
idxs <- seq(from=1, to=255, by=16)
writeBinFragments(pathname, idxs=idxs, rev(x), size=size)
x <- readBinFragments(pathname, what="integer", size=size, signed=FALSE, idxs=idxs, origin="start")
print(x)
stopifnot(identical(x0, x))

close(con)

# Assert everything is as expected
# Read the complete file
x <- readBin(pathname, what="integer", size=size, signed=FALSE, n=length(data))
stopifnot(identical(x, data))
cat("Write connection...done\n")

# -----
# Clean up
# -----
file.remove(pathname)

```

| | |
|------------|--|
| readRdHelp | <i>Reads one or more Rd help files in a certain format</i> |
|------------|--|

Description

Reads one or more Rd help files in a certain format.

Usage

```
## Default S3 method:
readRdHelp(..., format=c("text", "html", "latex", "rd"), drop=TRUE)
```

Arguments

| | |
|--------|---|
| ... | Arguments passed to help . |
| format | A character string specifying the return type. |
| drop | If FALSE or more than one help entry is found, the result is returned as a list . |

Value

Returns a [list](#) of [character](#) strings or a single [character](#) string.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

| | |
|-----------|-------------------------------------|
| readTable | <i>Reads a file in table format</i> |
|-----------|-------------------------------------|

Description

Reads a file in table format and creates a data frame from it, with cases corresponding to lines and variables to fields in the file.

WARNING: This method is very much in an alpha stage. Expect it to change.

This method is an extension to the default [read.table](#) function in R. It is possible to specify a column name to column class map such that the column classes are automatically assigned from the column header in the file.

In addition, it is possible to read any subset of rows. The method is optimized such that only columns and rows that are of interest are parsed and read into R's memory. This minimizes memory usage at the same time as it speeds up the reading.

Usage

```
## Default S3 method:
readTable(file, colClasses=NULL, isPatterns=FALSE, defColClass=NA, header=FALSE, skip=0, nrows=-1, row
```

Arguments

| | |
|-------------|---|
| file | A connection or a filename. If a filename, the path specified by path is added to the front of the filename. Unopened files are opened and closed at the end. |
| colClasses | Either a named or an unnamed character vector . If unnamed, it specified the column classes just as used by read.table . If it is a named vector, names(colClasses) are used to match the column names read (this requires that header=TRUE) and the column classes are set to the corresponding values. |
| isPatterns | If TRUE , the matching of names(colClasses) to the read column names is done by regular expressions matching. |
| defColClass | If the column class map specified by a named colClasses argument does not match some of the read column names, the column class is by default set to this class. The default is to read the columns in an "as is" way. |
| header | If TRUE , column names are read from the file. |
| skip | The number of lines (commented or non-commented) to skip before trying to read the header or alternatively the data table. |
| nrows | The number of rows to read of the data table. Ignored if rows is specified. |
| rows | An row index vector specifying which rows of the table to read, e.g. row one is the row following the header. Non-existing rows are ignored. Note that rows are returned in the same order they are requested and duplicated rows are also returned. |
| col.names | Same as in read.table() . |
| check.names | Same as in read.table() , but default value is FALSE here. |
| path | If file is a filename, this path is added to it, otherwise ignored. |
| ... | Arguments passed to read.table used internally. |
| stripQuotes | If TRUE , quotes are stripped from values before being parse. This argument is only effective when method=="readLines". |
| method | If "readLines", (readLines()) is used internally to first only read rows of interest, which is then passed to read.table() . If "intervals", contiguous intervals are first identified in the rows of interest. These intervals are the read one by one using read.table() . The latter methods is faster and especially more memory efficient if the intervals are not too many, where as the former is preferred if many "scattered" rows are to be read. |
| verbose | A logical or a Verbose object. |

Value

Returns a [data.frame](#).

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[readTableIndex\(\)](#). [read.table.colClasses\(\)](#).

| | |
|----------------|--|
| readTableIndex | <i>Reads a single column from file in table format</i> |
|----------------|--|

Description

Reads a single column from file in table format, which can then be used as an index-to-row (look-up) map for fast access to a subset of rows using `readTable()`.

Usage

```
## Default S3 method:  
readTableIndex(..., indexColumn=1, colClass="character", verbose=FALSE)
```

Arguments

| | |
|--------------------------|--|
| <code>indexColumn</code> | An single integer of the index column. |
| <code>colClass</code> | A single character specifying the class of the index column. |
| <code>...</code> | Arguments passed to <code>readTable()</code> used internally. |
| <code>verbose</code> | A logical or a Verbose object. |

Value

Returns a [vector](#).

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[readTable\(\)](#).

Examples

```
## Not run:  
# File containing data table to be access many times  
filename <- "somefile.txt"  
  
# Create a look-up index  
index <- readTableIndex(filename)  
  
# Keys of interest  
keys <- c("foo", "bar", "wah")  
  
# Read only those keys and do it fast  
df <- readTable(filename, rows=match(keys, index))  
  
## End(Not run)
```

readWindowsShortcut *Reads a Microsoft Windows Shortcut (.lnk file)*

Description

Reads a Microsoft Windows Shortcut (.lnk file).

Usage

```
## Default S3 method:  
readWindowsShortcut(con, verbose=FALSE, ...)
```

Arguments

| | |
|---------|--|
| con | A connection or a character string (filename). |
| verbose | If TRUE , extra information is written while reading. |
| ... | Not used. |

Details

The MIME type for a Windows Shortcut file is `application/x-ms-shortcut`.

Value

Returns a [list](#) structure.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

References

- [1] Wotsit's Format, <http://www.wotsit.org/>, 2005.
- [2] Hager J, *The Windows Shortcut File Format* (as reverse-engineered by), version 1.0.
- [3] Microsoft Developer Network, *IShellLink Interface*, 2008. <http://msdn2.microsoft.com/en-us/library/bb774950.aspx>
- [4] Andrews D, *Parsing Windows Shortcuts (lnk) files in java*, `comp.lang.java.help`, Aug 1999. http://groups.google.com/group/comp.lang.java.help/browse_thread/thread/a2e147b07d5480a2/
- [5] Multiple authors, *Windows shell links* (in Tcl), TcLer's Wiki, April 2008. <http://wiki.tcl.tk/1844>
- [6] Daniel S. Bensen, *Shortcut File Format (.lnk)*, `Stdlib.com`, April 24, 2009. <http://www.stdlib.com/art6-Shortcut-File-Format-lnk.html>
- [7] [MS-SHLLINK]: Shell Link (.LNK) Binary File Format, Microsoft Inc., September 25, 2009.

See Also

[createWindowsShortcut\(\)](#) [filePath](#)

Examples

```

pathname <- system.file("data-ex/HISTORY.LNK", package="R.utils")
lnk <- readWindowsShortcut(pathname)

# Print all information
print(lnk)

# Get the relative path to the target file
history <- file.path(dirname(pathname), lnk$relativePath)

# Alternatively, everything in one call
history <- filePath(pathname, expandLinks="relative")

```

relibrary

Reloads a package

Description

Reloads a package. This function works exactly the same as `library`, *except* that it reloads the package if it already loaded. This is useful for developers. For more information see `library`.

Usage

```
relibrary(package, character.only=FALSE, warn.conflicts=FALSE, ...)
```

Arguments

| | |
|-----------------------------|--|
| <code>package</code> | Name or <code>character</code> string giving the name of a package. |
| <code>character.only</code> | A <code>logical</code> indicating whether package can be assumed to be character strings. |
| <code>warn.conflicts</code> | If <code>TRUE</code> , warnings are printed about conflicts from reattaching of the package, unless that package contains an object <code>.conflicts.OK</code> . |
| <code>...</code> | Any other arguments that <code>library()</code> takes. |

Details

While `relibrary` is reloading a package the option `relibrary` will be set to name of the package currently reloaded. This can be useful if the package to be reloaded would like save away data until it is loaded again.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

See `library()`.

| | |
|-----------------|----------------------------|
| removeDirectory | <i>Removes a directory</i> |
|-----------------|----------------------------|

Description

Removes a directory, and if requested, also its contents.

Usage

```
## Default S3 method:  
removeDirectory(path, recursive=FALSE, mustExist=TRUE, ...)
```

Arguments

| | |
|-----------|---|
| path | A character string specifying the directory to be removed. |
| recursive | If TRUE , subdirectories and files are also removed. If FALSE , and directory is non-empty, an exception is thrown. |
| mustExist | If TRUE , and the directory does not exist, an exception is thrown. |
| ... | Not used. |

Value

Returns (invisibly) [TRUE](#), the directory was successfully removed, otherwise [FALSE](#), unless an exception is thrown.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

Internally [unlink\(\)](#) is used.

| | |
|----------|---|
| resample | <i>Sample values from a set of elements</i> |
|----------|---|

Description

Sample values from a set of elements. Contrary to [sample\(\)](#), this function also works as expected when there is only one element in the set to be sampled, cf. [1]. This function originates from the example code of [sample\(\)](#) as of R v2.12.0.

Usage

```
## Default S3 method:  
resample(x, ...)
```

Arguments

x A [vector](#) of any length and data type.
... Additional arguments passed to [sample.int\(\)](#).

Value

Returns a sampled [vector](#) of the same data types as argument x.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

References

[1] Henrik Bengtsson, *Using sample() to sample one value from a single value?*, R-devel mailing list, 2010-11-03.

See Also

Internally [sample\(\)](#) is used.

resetWarnings

Resets recorded warnings

Description

Resets recorded warnings.

Usage

```
## Default S3 method:  
resetWarnings(...)
```

Arguments

... Not used.

Value

Returns (invisibly) the number of warnings removed.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[warnings\(\)](#)

| | |
|------------|--|
| saveObject | <i>Saves an object to a file or a connection</i> |
|------------|--|

Description

Saves an object to a file or a connection.

Usage

```
## Default S3 method:  
saveObject(object, file=NULL, path=NULL, compress=TRUE, ..., safe=TRUE)
```

Arguments

| | |
|----------|---|
| object | The object to be saved. |
| file | A filename or connection where the object should be saved. If <code>NULL</code> , the filename will be the hash code of the object plus ".xdr". |
| path | Optional path, if file is a filename. |
| compress | If <code>TRUE</code> , the file is compressed to, otherwise not. |
| ... | Other arguments accepted by <code>save()</code> in the base package. |
| safe | If <code>TRUE</code> and file is a file, then, in order to lower the risk for incomplete files, the object is first written to a temporary file, which is then renamed to the final name. |

Value

Returns (invisibly) the pathname or the [connection](#).

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[loadObject\(\)](#) to load an object from file. [digest](#) for how hash codes are calculated from an object.

| | |
|--------------------|--|
| seqToHumanReadable | <i>Gets a short human readable string representation of an vector of indices</i> |
|--------------------|--|

Description

Gets a short human readable string representation of an vector of indices.

Usage

```
## Default S3 method:  
seqToHumanReadable(idx, delimiter="-", collapse=", ", ...)
```

Arguments

| | |
|-----------|---|
| idx | A vector of integer indices. |
| delimiter | A character string delimiter. |
| collapse | A character string used to collapse subsequences. |
| ... | Not used. |

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[seqToIntervals\(\)](#).

Examples

```
print(seqToHumanReadable(1:10)) # "1-10"  
print(seqToHumanReadable(c(1:10, 15:18, 20))) # "1-10, 15-18, 20"
```

| | |
|----------------|---|
| seqToIntervals | <i>Gets all contiguous intervals of a vector of indices</i> |
|----------------|---|

Description

Gets all contiguous intervals of a vector of indices.

Usage

```
## Default S3 method:  
seqToIntervals(idx, ...)
```

Arguments

idx A [vector](#) of [integer](#) indices.
... Not used.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[*intervalsToSeq\(\)](#). To identify sequences of *equal* values, see [rle\(\)](#).

Examples

```
x <- 1:10
y <- seqToIntervals(x)
print(y) # [1 10]

x <- c(1:10, 15:18, 20)
y <- seqToIntervals(x)
print(y) # [1 10; 15 18; 20 20]

z <- intervalsToSeq(y)
print(z)
stopifnot(all.equal(x,z))
```

setOption

Sets a option in R

Description

Sets a option in R by specifying its name as a [character](#) string.

Usage

```
## Default S3 method:
setOption(x, value, ...)
```

Arguments

x The name of the option to be set.
value The new value of the option.
... Not used.

Value

Returns (invisibly) the previous value of the option.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

See [getOption\(\)](#) and "base::options".

Settings

Class for applicational settings

Description

Package: R.utils

Class Settings

[Object](#)

~~|

~~+--[Options](#)

~~~~~|

~~~~~+--Settings

Directly known subclasses:

public static class **Settings**

extends [Options](#)

Class for applicational settings.

Usage

`Settings(basename=NULL, ...)`

Arguments

`basename` A [character](#) string of the basename of the settings file.

`...` Arguments passed to constructor of superclass [Options](#).

Fields and Methods**Methods:**

[findSettings](#)

Searches for the settings file in one or several directories.

[getLoadedPathname](#)

Gets the pathname of the settings file loaded.

[isModified](#)

Checks if settings has been modified compared to whats on file.

| | |
|----------------------------|--|
| <code>loadAnywhere</code> | Loads settings from file. |
| <code>promptAndSave</code> | Prompt user to save modified settings. |
| <code>saveAnywhere</code> | Saves settings to file. |

Methods inherited from Options:

`as.character`, `as.list`, `equals`, `getLeaves`, `getOption`, `hasOption`, `names`, `nbrOfOptions`, `setOption`, `str`

Methods inherited from Object:

`$`, `$<-`, `[[`, `[[<-`, `as.character`, `attach`, `attachLocally`, `clearCache`, `clearLookupCache`, `clone`, `detach`, `equals`, `extend`, `finalize`, `gc`, `getEnvironment`, `getFieldModifier`, `getFieldModifiers`, `getFields`, `getInstantiationTime`, `getStaticInstance`, `hasField`, `hashCode`, `ll`, `load`, `objectSize`, `print`, `registerFinalizer`, `save`

Load settings with package and save on exit

Here is a generic `.First.lib()` function for loading settings with package. It also (almost) assures that the package is detached when R finishes. See `onSessionExit()` why it is not guaranteed!

The almost generic `.Last.lib()` function, which will prompt user to save settings, is called when a package is detached.

It is custom to put these functions in a file named `zzz.R`.

.First.lib():

```
.First.lib <- function(libname, pkgname) {
  # Write a welcome message when package is loaded
  pkg <- Package(pkgname);
  assign(pkgname, pkg, pos=getPosition(pkg));

  # Read settings file "<pkgname>Settings" and store it in package
  # variable '<pkgname>Settings'.
  varname <- paste(pkgname, "Settings");
  basename <- paste(".", varname, sep="");
  settings <- Settings$loadAnywhere(basename, verbose=TRUE);
  if (is.null(settings))
    settings <- Settings(basename);
  assign(varname, settings, pos=getPosition(pkg));

  # Detach package when R finishes, which will save package settings too.
  onSessionExit(function(...) detachPackage(pkgname));

  packageStartupMessage(getName(pkg), " v", getVersion(pkg),
    " (", getDate(pkg), ") successfully loaded. See ?", pkgname,
    " for help.\n", sep="");
} # .First.lib()
```

.Last.lib():

```
.Last.lib <- function(libpath) {
```

```

pkgname <- "<package name>";

# Prompt and save package settings when package is detached.
varname <- paste(pkgname, "Settings", sep="");
if (exists(varname)) {
  settings <- get(varname);
  if (inherits(settings, "Settings"))
    promptAndSave(settings);
}
} # .Last.lib()

```

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

Examples

```

# Load settings from file, or create default settings
basename <- "some.settings"
settings <- Settings$loadAnywhere(basename)
if (is.null(settings))
  settings <- Settings(basename)

# Set default options, if missing.
setOption(settings, "graphics/verbose", TRUE, overwrite=FALSE)
setOption(settings, "io/verbose", Verbose(threshold=-1), overwrite=FALSE)

# Save and reload settings
path <- tempdir()
saveAnywhere(settings, path=path)
settings2 <- Settings$loadAnywhere(basename, paths=path)

# Clean up
file.remove(getLoadedPathname(settings2))

# Assert correctness
stopifnot(equals(settings, settings2))

```

SmartComments

Abstract class SmartComments

Description

Package: R.utils

Class SmartComments

Object

```
~~|
~~+--SmartComments
```

Directly known subclasses:

[LComments](#), [VComments](#)

```
public abstract static class SmartComments
  extends Object
```

Abstract class SmartComments.

Usage

```
SmartComments(letter=NA, ...)
```

Arguments

| | |
|--------|--------------------------------------|
| letter | A single character . |
| ... | Not used. |

Details

A "smart" source-code comment is an R comment, which start with a '#', but is followed by a single letter, then a single symbol and a second '#' and then an option character string, and there must not be any code before the comment on the same line. In summary, a smart comment line has format: <white spaces>#<letter><symbol># <some text>.

Example code with two smart comments (VComments):

```
x <- 2
#V1# threshold=-1
#Vc# A v-comment log message
cat("Hello world")
```

which after compilation becomes

```
x <- 2
verbose <- Verbose(threshold=-1)
if (verbose) { cat(verbose, "A v-comment log message"); }
cat("Hello world")
```

Fields and Methods**Methods:**

| | |
|-----------------------------|--|
| <code>compile</code> | Preprocess a vector of code lines. |
| <code>convertComment</code> | Converts a single smart comment to R code. |
| <code>parse</code> | Parses one single smart comment. |
| <code>reset</code> | Resets a SmartComments compiler. |
| <code>validate</code> | Validates the compiled lines. |

Methods inherited from Object:

`$`, `$<-`, `[[`, `[[<-`, `as.character`, `attach`, `attachLocally`, `clearCache`, `clearLookupCache`, `clone`, `detach`, `equals`, `extend`, `finalize`, `gc`, `getEnvironment`, `getFieldModifier`, `getFieldModifiers`, `getFields`, `getInstantiationTime`, `getStaticInstance`, `hasField`, `hashCode`, `ll`, `load`, `objectSize`, `print`, `registerFinalizer`, `save`

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

`VComments`.

| | |
|-----------------|--|
| sourceDirectory | <i>Sources files recursively to either local or global environment</i> |
|-----------------|--|

Description

Sources files recursively to either local or global environment.

Usage

```
## Default S3 method:
sourceDirectory(path, pattern=".*[.](r|R|s|S|q)([.](lnk|LNK))*$", recursive=TRUE, envir=parent.frame)
```

Arguments

| | |
|------------------------|---|
| <code>path</code> | A path to a directory to be sourced. |
| <code>pattern</code> | A regular expression file name pattern to identify source code files. |
| <code>recursive</code> | If <code>TRUE</code> , subdirectories are recursively sourced first, otherwise not. |
| <code>envir</code> | An <code>environment</code> in which the code should be evaluated. |
| <code>onError</code> | If an error occurs, the error may stop the job, give a warning, or silently be skipped. |
| <code>verbose</code> | A <code>logical</code> or a <code>Verbose</code> object. |
| <code>...</code> | Additional arguments passed to <code>sourceTo()</code> . |

Value

Returns a `vector` of the full pathnames of the files sourced.

Details

Subdirectories and files in each (sub-)directory are sourced in lexicographic order.

Hooks

This method does not provide hooks, but the internally used `sourceTo()` does.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

`sourceTo()` and compare to `source()`.

sourceTo

Parses and evaluates code from a file or a connection

Description

Parses and evaluates code from a file or a connection. This has the same effect as if `source(..., local=TRUE)` would have been called from within the given environment. This is useful when setting up a new local working environment.

Usage

```
## Default S3 method:
sourceTo(file, path=NULL, chdir=FALSE, ..., local=TRUE, envir=parent.frame(), modifiedOnly=FALSE)
```

Arguments

| | |
|--------------|---|
| file | A connection or a character string giving the pathname of the file or URL to read from. |
| path | An optional character string giving the path to the file. Ignored if file is a connection. |
| chdir | If TRUE and file is a pathname, the R working directory is temporarily changed to the directory containing file for evaluating. |
| ... | Arguments to <code>source()</code> . If argument file is not explicitly given, the first argument is assumed to be the file argument. This argument is converted into a string by <code>as.character()</code> . |
| local | If FALSE , evaluation is done in the global environment, otherwise in the calling environment. |
| envir | An environment in which <code>source()</code> should be called. If NULL , the global environment is used. |
| modifiedOnly | If TRUE , the file is sourced only if modified since the last time it was sourced, otherwise regardless. |

Value

Return the result of `source()`.

Hooks

This methods recognizes the hook `sourceTo/onPreprocess`, which is called after the lines in file has been read, but before they have been parsed by the R parser, cf. `parse()`. An `onPreprocess` hook function should take a `character vector` of code lines and return a `character vector` of code lines. This can for instance be used to pre-process R source code with special directives such as `VComments`.

Note that only one hook function can be used for this function, otherwise an error is generated.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

`sourceDirectory()`. `sys.source()` and `source()`.

Examples

```
# - - - - -
# Example 1
# - - - - -
cat("=== Example 1 =====\n")

foo <- function(file, ...) {
  cat("Local objects before calling sourceTo():\n")
  print(ls())

  res <- sourceTo(file, ...)

  cat("Local objects after calling sourceTo():\n")
  print(ls())
}

cat("Global objects before calling foo():\n")
lsBefore <- NA
lsBefore <- ls()
foo(file=textConnection(c('a <- 1', 'b <- 2'))))

cat("Global objects after calling foo():\n")
stopifnot(length(setdiff(ls(), lsBefore)) == 0)

# - - - - -
# Example 2 - with VComments preprocessor
# - - - - -
cat("=== Example 2 =====\n")
```

```

preprocessor <- function(lines, ...) {
  cat("-----\n")
  cat("Source code before preprocessing:\n")
  displayCode(code=lines, pager="console")
  cat("-----\n")
  cat("Source code after preprocessing:\n")
  lines <- VComments$compile(lines)
  displayCode(code=lines, pager="console")
  cat("-----\n")
  lines
}

oldHooks <- getHook("sourceTo/onPreprocess")
setHook("sourceTo/onPreprocess", preprocessor, action="replace")
code <- c(
  'x <- 2',
  '#V1# threshold=-1',
  '#Vc# A v-comment log message',
  'print("Hello world")'
)
fh <- textConnection(code)
sourceTo(fh)
setHook("sourceTo/onPreprocess", oldHooks, action="replace")

```

splitByPattern

Splits a single character string by pattern

Description

Splits a single character string by pattern. The main difference compared to `strsplit()` is that this method also returns the part of the string that matched the pattern. Also, it only takes a single character string.

Usage

```
## Default S3 method:
splitByPattern(str, pattern, ...)
```

Arguments

| | |
|----------------------|--|
| <code>str</code> | A single character string to be split. |
| <code>pattern</code> | A regular expression character string. |
| <code>...</code> | Not used. |

Value

Returns a named [character vector](#) with names equal to "TRUE" if element is a pattern part and "FALSE" otherwise.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

Compare to [strsplit\(\)](#).

Examples

```
rspCode <- "<body>Hello <%=\"world\"%></body>"
rspParts <- splitByPattern(rspCode, pattern="<%.*%>")
cat(rspCode, "\n")
print(rspParts)
```

stext

Writes text in the margin along the sides of a plot

Description

Writes text in the margin along the sides of a plot.

Usage

```
## Default S3 method:
stext(text, side=1, line=0, pos=0.5, margin=c(0.2, 0.2), charDim=c(strwidth("M", cex = cex), strheight("M", cex = cex)), ...)
```

Arguments

| | |
|---------|---|
| text | The text to be written. See mtext for details. |
| side | An integer specifying which side to write the text on. See mtext for details. |
| line | A numeric specifying on which line to write on. |
| pos | A numeric , often in [0,1], specifying the position of the text relative to the left and right edges. |
| margin | A numeric vector length two specifying the text margin. |
| charDim | A numeric vector length two specifying the size of a typical symbol. |
| cex | A numeric specifying the character expansion factor. |
| ... | Additional arguments passed to mtext . |

Value

Returns what [mtext](#) returns.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

Internally `mtext` is used.

| | |
|----------|-----------------------------------|
| subplots | <i>Creates a grid of subplots</i> |
|----------|-----------------------------------|

Description

Creates a grid of subplots in the current figure. If arguments `nrow` and `ncol` are given a `nrow`-by-`ncol` grid of subplots are created. If only argument `n` is given then a `r`-by-`s` grid is created where $|r-s| \leq 1$, i.e. a square or almost a square of subplots is created. If `n` and `nrow` is given then a grid with `nrow` rows and at least `n` subplots are created. Similar if `n` and `ncol` is given. The argument `byrow` specifies if the order of the subplots should be rowwise (`byrow=TRUE`) or columnwise.

Usage

```
## Default S3 method:  
subplots(n=1, nrow=NULL, ncol=NULL, byrow=TRUE, ...)
```

Arguments

| | |
|--------------------|---|
| <code>n</code> | If given, the minimum number of subplots. |
| <code>nrow</code> | If given, the number of rows the grid of subplots should contain. |
| <code>ncol</code> | If given, the number of columns the grid of subplots should contain. |
| <code>byrow</code> | If <code>TRUE</code> , the panels are ordered row by row in the grid, otherwise column by column. |
| <code>...</code> | Not used. |

Value

Returns the `matrix` containing the order of plots.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

`layout` and `layout.show()`.

Examples

```

subplots(nrow=2, ncol=3) # 2-by-3 grid of subplots
subplots(n=6, nrow=2)   # 2-by-3 grid of subplots
subplots(n=5, ncol=2)   # 3-by-2 grid of subplots
subplots(1)             # (Reset) to a 1-by-1 grid of subplots
subplots(2)             # 1-by-2 grid of subplots
subplots(3)             # 2-by-2 grid of subplots
l <- subplots(8)        # 3-by-3 grid of subplots
layout.show(length(l))

```

System

*Static class to query information about the system***Description**

Package: R.utils

Class System**Object**

```

~~|
~~+--System

```

Directly known subclasses:

```

public static class System
extends Object

```

The System class contains several useful class fields and methods. It cannot be instantiated.

Fields and Methods**Methods:**

| | |
|--|--|
| currentTimeMillis | Get the current time in milliseconds. |
| findGhostscript | Searches for the ghostview binary on the current system. |
| findGraphicsDevice | Searches for a working PNG device. |
| getHostname | Retrieves the computer name of the current host. |
| getMappedDrivesOnWindows | - |
| getUsername | Retrieves the name of the user running R. |
| mapDriveOnWindows | - |
| openBrowser | Opens an HTML document using the OS default HTML browser. |
| parseDebian | Parses a string, file or connection for Debian formatted parameters. |
| unmapDriveOnWindows | - |

Methods inherited from Object:

\$, \$<-, [[, [[<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

| | |
|---------------|---|
| TextStatusBar | <i>A status bar at the R prompt that can be updated</i> |
|---------------|---|

Description

Package: R.utils

Class TextStatusBar

[Object](#)

~~|

~~+---TextStatusBar

Directly known subclasses:

```
public static class TextStatusBar
extends Object
```

A status bar at the R prompt that can be updated.

Usage

```
TextStatusBar(fmt=paste("%-", getOption("width") - 1, "s", sep = ""), ...)
```

Arguments

| | |
|-----|---|
| fmt | A character format string to be used by sprintf() . Default is a left-aligned string of full width. |
| ... | Named arguments to be passed to sprintf() together with the format string. |

Details

A label with name `hfill` can be used for automatic horizontal filling. It must be [numeric](#) and be immediate before a string label such that a `hfill` label and the following string label together specifies an `sprintf` format such as `"%*-s"`. The value of `hfill` will be set such that the resulting status bar has width equal to `getOption("width")-1` (the reason for the `-1` is to prevent the text status bar from writing into the next line). If more than one `hfill` label is used their widths will be uniformly distributed. Left over spaces will be distributed between `hfill` labels with initial values of one.

Fields and Methods**Methods:**

| | |
|---------------------------|---|
| <code>flush</code> | Flushes the output. |
| <code>getLabel</code> | Gets the current value of a label. |
| <code>newline</code> | Writes a newline. |
| <code>popMessage</code> | Adds a message above the status bar. |
| <code>setLabel</code> | Sets the value of a label. |
| <code>setLabels</code> | Sets new values of given labels. |
| <code>update</code> | Updates the status bar (visually). |
| <code>updateLabels</code> | Sets the new values of given labels and updates the status bar. |

Methods inherited from Object:

`$`, `$<-`, `[[`, `[[<-`, `as.character`, `attach`, `attachLocally`, `clearCache`, `clearLookupCache`, `clone`, `detach`, `equals`, `extend`, `finalize`, `gc`, `getEnvironment`, `getFieldModifier`, `getFieldModifiers`, `getFields`, `getInstantiationTime`, `getStaticInstance`, `hasField`, `hashCode`, `ll`, `load`, `objectSize`, `print`, `registerFinalizer`, `save`

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

Examples

```
# -----
# Read all HTML files in the base package
# -----
path <- system.file("html", package="base");
files <- list.files(path, full.names=TRUE)
nfiles <- length(files)

cat(sprintf("Reading %d files in %s:\n", nfiles, path))

# Create a status bar with four labels
sb <- TextStatusBar("File: %-*s [%3.0f%% %6.0f lines %-8s]",
                    hfill=1, file="", progress=0, nlines=0, time="")

nlines <- 0
for (kk in seq(length=nfiles)) {
  file <- files[kk]

  # Update the status bar
  if (sb) {
    setLabel(sb, "progress", 100*kk/nfiles)
    if (kk %% 10 == 1 || kk == nfiles)
      setLabel(sb, "file", substr(basename(file), 1, 44))

    size <- file.info(file)$size/1024;
    # popMessage() calls update() too
  }
}
```

```

    popMessage(sb, sprintf("Processing %s (%.2fkB)",
                           basename(file), size))
    flush(sb)
  }

  # Read the file
  lines <- readLines(file)
  nlines <- nlines + length(lines)

  # Emulate slow process
  if (interactive()) {
    Sys.sleep(rexp(1, rate=40))
  }

  # Update the status bar
  if (sb) {
    setLabel(sb, "nlines", nlines)
    setLabel(sb, "time", format(Sys.time(), "%H:%M:%S"))
    update(sb)
  }
}
cat("\n")

```

TimeoutException

TimeoutException represents timeout errors

Description

Package: R.utils

Class TimeoutException

Object

```

~|
~+--try-error
~~~~~|
~~~~~+--condition
~~~~~|
~~~~~+--error
~~~~~|
~~~~~+--simpleError
~~~~~|
~~~~~+--Exception
~~~~~|
~~~~~+--TimeoutException

```

Directly known subclasses:

```
public static class TimeoutException  
extends Exception
```

TimeoutException represents timeout errors occurring when a set of R expressions executed did not finish in time.

Usage

```
TimeoutException(..., cpu=NA, elapsed=NA)
```

Arguments

| | |
|--------------|--|
| ... | Any arguments accepted by Exception . |
| cpu, elapsed | The maximum time the R expressions were allowed to be running before the timeout occurred as measured in CPU time and (physically) elapsed time. |

Fields and Methods

Methods:

[getMessage](#) Gets the message of the exception.

Methods inherited from [Exception](#):

as.character, getCall, getLastException, getMessage, getStackTrace, getWhen, print, printStackTrace, throw

Methods inherited from [error](#):

as.character, throw

Methods inherited from [condition](#):

as.character, conditionCall, conditionMessage, print

Methods inherited from [Object](#):

\$, \$<-, [[, [[<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

For detailed information about exceptions see [Exception](#).

`toEPS`*Methods for creating image files of a specific format*

Description

Methods for creating image files of a specific format.

Usage

```
toEPS(name, ..., field="fullname")
```

Arguments

| | |
|--------------------|--|
| <code>name</code> | A character string specifying the name of the image file. |
| <code>...</code> | Additional arguments passed to devEval() , e.g. <code>tags</code> and <code>aspectRatio</code> . |
| <code>field</code> | Passed to devEval() . |

Value

Returns by default the so called *fullname* of the generated image file. The fullname of an file is the filename without the extension.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

These functions are wrappers for [devEval\(\)](#). See [devOptions\(\)](#) to change the default dimensions for a specific device type.

`touchFile`*Updates the timestamp of a file*

Description

Updates the timestamp of a file. Currently, it is only possible to change the timestamp specifying when the file was last modified, and time can only be set to the current time.

Usage

```
## Default S3 method:  
touchFile(pathname, ...)
```

Arguments

pathname A **character** specifying the file to be updated.
... Not used.

Value

Returns (invisibly) the old timestamp.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

References

[1] R-devel mailing list thread *Unix-like touch to update modification timestamp of file?*, started on 2008-02-26. <http://tolstoy.newcastle.edu.au/R/e4/devel/08/02/0654.html>

See Also

[file.info\(\)](#).

Examples

```
# 1. Create a file
pathname <- tempfile()
cat(file=pathname, "Hello world!")
md5a <- digest::digest(pathname, file=TRUE)

# 2. Current time stamp
ta <- file.info(pathname)$mtime
print(ta)

# 3. Update time stamp
Sys.sleep(1.2)
touchFile(pathname)
tb <- file.info(pathname)$mtime
print(tb)

# 4. Verify that the timestamp got updated
stopifnot(tb > ta)

# 5. Verify that the contents did not change
md5b <- digest::digest(pathname, file=TRUE)
stopifnot(identical(md5a, md5b))
```

| | |
|-------|---------------------------------------|
| toUrl | <i>Converts a pathname into a URL</i> |
|-------|---------------------------------------|

Description

Converts a pathname into a URL starting with file://.

Usage

```
## Default S3 method:  
toUrl(pathname, safe=TRUE, ...)
```

Arguments

| | |
|----------|---|
| pathname | A character string of the pathname to be made into a URL. |
| safe | If TRUE , certain "unsafe" characters are escaped. |
| ... | Not used. |

Value

Returns a [character](#) string.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[URLencode](#).

| | |
|--------------|---|
| unwrap.array | <i>Unwrap an array, matrix or a vector to an array of more dimensions</i> |
|--------------|---|

Description

Unwrap an array, matrix or a vector to an array of more dimensions. This is done by splitting up each dimension into several dimension based on the names of that dimension.

Usage

```
## S3 method for class 'array'  
unwrap(x, split=rep("[.]", length(dim(x))), drop=FALSE, ...)
```

Arguments

| | |
|-------|--|
| x | An array or a matrix . |
| split | A list or a character vector . If a list , it should contain functions that takes a character vector as the first argument and optional ... arguments. Each function should split the vector into a list of same length and where all elements contains the same number of parts. If a character vector , each element split[i] is replaced by a function call function(names, ...) strsplit(names, split=split[i]). |
| drop | If TRUE , dimensions of of length one are dropped, otherwise not. |
| ... | Arguments passed to the split functions . |

Details

Although not tested thoroughly, unwrap() should be the inverse of wrap() such that identical(unwrap(wrap(x)), x) holds.

Value

Returns an [array](#).

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[*wrap\(\)](#).

Examples

```
## Not run: See ?wrap.array for an example
```

VComments

The VComments class

Description

Package: R.utils

Class VComments

[Object](#)

```
~~|
```

```
~~+--SmartComments
```

```
~~~~~|
```

```
~~~~~+--VComments
```

Directly known subclasses:[LComments](#)

```
public static class VComments
  extends SmartComments
```

The VComments class.

Usage

```
VComments(letter="V", verboseName="verbose", ...)
```

Arguments

| | |
|-------------|---------------------------------|
| letter | The smart letter. |
| verboseName | The name of the verbose object. |
| ... | Not used. |

Details

The 'v' in VComments stands for 'verbose', because of its relationship to the [Verbose](#) class. Here is a list of VComments and the R code that replaces each of them by the compiler:

Constructors

- `\#V0\#[<args>]` - `NullVerbose(<args>)`
- `\#V1\#[<args>]` - `Verbose(<args>)`

Controls

- `\#V=\#[<variable>]` - Sets the name of the `<verbose>` object. Default is 'verbose'.
- `\#V^\#[<threshold>]` - `setThreshold(<verbose>, <threshold>)`
- `\#V?\#[<expression>]` - `if (isVisible(<verbose>)) { <expression> }`
- `\#V@\#[<level>]` - `setDefaultLevel(<verbose>, <level>)`
- `\#Vm\#[<method> <args>]` - `<method>(<verbose>, <args>)`

Enters and exits

- `\#V+\#[<message>]` - `enter(<verbose>, <message>)`
- `\#V-\#[<message>]` - `exit(<verbose>, <message>)`
- `\#V!\#[<message>]` - `pushState(<verbose>)`
`on.exit(popState(<verbose>))`
If `<message>`, `enter(<verbose>, <message>)`

Simple output

- `\#Vn\#<ignored>` - `newline(<verbose>)`
- `\#Vr\#<ignored>` - `ruler(<verbose>)`
- `\#Vt\#<ignored>` - `timestamp(<verbose>)`
- `\#Vw\#[<title>]` - `warnings(<verbose>, <title>)`

Output messages

- `\#Vc\#[<message>]` - `cat(<verbose>, <message>)`
- `\#Ve\#<expression>` - `eval(<verbose>, <expression>)`
- `\#Vh\#<message>` - `header(<verbose>, <message>)`
- `\#Vp\#<object>` - `print(<verbose>, <object>)`
- `\#Vs\#<object>` - `summary(<verbose>, <object>)`
- `\#Vz\#<object>` - `str(<verbose>, <object>)`

Fields and Methods

Methods:

| | |
|-----------------------------|---------------------------------------|
| <code>convertComment</code> | Converts a verbose comment to R code. |
| <code>reset</code> | Resets a VComments compiler. |
| <code>validate</code> | Validates the compiled lines. |

Methods inherited from SmartComments:

`compile`, `convertComment`, `parse`, `reset`, `validate`

Methods inherited from Object:

`$`, `$<-`, `[[`, `[[<-`, `as.character`, `attach`, `attachLocally`, `clearCache`, `clearLookupCache`, `clone`, `detach`, `equals`, `extend`, `finalize`, `gc`, `getEnvironment`, `getFieldModifier`, `getFieldModifiers`, `getFields`, `getInstantiationTime`, `getStaticInstance`, `hasField`, `hashCode`, `ll`, `load`, `objectSize`, `print`, `registerFinalizer`, `save`

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

Examples

```
filename <- system.file("data-ex/exampleVComments.R", package="R.utils")
lines <- readLines(filename)

cat("Code before preprocessing:\n")
displayCode(code=lines, pager="console")

lines <- VComments$compile(lines)

cat("Code after preprocessing:\n")
displayCode(code=lines, pager="console")
```

 Verbose

 Class to writing verbose messages to a connection or file

Description

Package: R.utils

Class Verbose

[Object](#)

~~|

~~+--Verbose

Directly known subclasses:

[MultiVerbose](#), [NullVerbose](#)

public static class **Verbose**

extends [Object](#)

Class to writing verbose messages to a connection or file.

Usage

```
Verbose(con=stderr(), on=TRUE, threshold=0, asGString=TRUE, timestamp=FALSE, removeFile=TRUE, core=TR
```

Arguments

| | |
|------------|--|
| con | A connection or a character string filename. |
| on | A logical indicating if the writer is on or off. |
| threshold | A numeric threshold that the level argument of any write method has to be equal to or larger than in order to the message being written. Thus, the lower the threshold is the more and more details will be outputted. |
| timestamp | If TRUE , each output is preceded with a timestamp. |
| removeFile | If TRUE and con is a filename, the file is first deleted, if it exists. |
| asGString | If TRUE , all messages are interpreted as GString before being output, otherwise not. |
| core | Internal use only. |
| ... | Not used. |

Fields and Methods

Methods:

[as.character](#) Returns a character string version of this object.

| | |
|---------------------------------|---|
| <code>as.double</code> | Gets a numeric value of this object. |
| <code>as.logical</code> | Gets a logical value of this object. |
| <code>capture</code> | Captures output of a function. |
| <code>cat</code> | Concatenates and prints objects if above threshold. |
| <code>enter</code> | Writes a message and indents the following output. |
| <code>equals</code> | Checks if this object is equal to another. |
| <code>evaluate</code> | Evaluates a function and prints its results if above threshold. |
| <code>exit</code> | Writes a message and unindents the following output. |
| <code>getThreshold</code> | Gets current verbose threshold. |
| <code>getTimestampFormat</code> | Gets the default timestamp format. |
| <code>header</code> | Writes a header. |
| <code>isOn</code> | Checks if the output is on. |
| <code>isVisible</code> | Checks if a certain verbose level will be shown or not. |
| <code>less</code> | Creates a cloned instance with a higher threshold. |
| <code>more</code> | Creates a cloned instance with a lower threshold. |
| <code>newline</code> | Writes one or several empty lines. |
| <code>off</code> | Turn off the output. |
| <code>on</code> | Turn on the output. |
| <code>popState</code> | - |
| <code>print</code> | Prints objects if above threshold. |
| <code>printf</code> | Formats and prints object if above threshold. |
| <code>pushState</code> | Pushes the current indentation state of the Verbose object. |
| <code>ruler</code> | Writes a ruler. |
| <code>setDefaultLevel</code> | Sets the current default verbose level. |
| <code>setThreshold</code> | Sets verbose threshold. |
| <code>setTimestampFormat</code> | Sets the default timestamp format. |
| <code>str</code> | Prints the structure of an object if above threshold. |
| <code>summary</code> | Generates a summary of an object if above threshold. |
| <code>timestamp</code> | Writes a timestamp. |
| <code>timestampOff</code> | - |
| <code>timestampOn</code> | Turns automatic timestamping on and off. |
| <code>warnings</code> | Outputs any warnings recorded. |
| <code>writeRaw</code> | Writes objects if above threshold. |

Methods inherited from Object:

`$`, `$<-`, `[]`, `[[<-`, `as.character`, `attach`, `attachLocally`, `clearCache`, `clearLookupCache`, `clone`, `detach`, `equals`, `extend`, `finalize`, `gc`, `getEnvironment`, `getFieldModifier`, `getFieldModifiers`, `getFields`, `getInstantiationTime`, `getStaticInstance`, `hasField`, `hashCode`, `ll`, `load`, `objectSize`, `print`, `registerFinalizer`, `save`

Output levels

As a guideline, use the following levels when outputting verbose/debug message using the Verbose class. For a message to be shown, the output level must be greater than (not equal to) current threshold. Thus, the lower the threshold is set, the more messages will be seen.

- `<= -100` Only for debug messages, i.e. messages containing all necessary information for debugging purposes and to find bugs in the code. Normally these messages are so detailed so

they will be a pain for the regular user, but very useful for bug reporting and bug tracking by the developer.

- -99 – -11 Detailed verbose messages. These will typically be useful for the user to understand what is going on and do some simple debugging fixing problems typically due to themselves and not due to bugs in the code.
- -10 – -1 Verbose messages. For example, these will typically report the name of the file to be read, the current step in a sequence of analysis steps and so on. These message are not very useful for debugging.
- 0 Default level in all output methods and default threshold. Thus, by default, messages at level 0 are not shown.
- $\geq +1$ Message that are always outputted (if threshold is kept at 0). We recommend not to output message at this level, because methods should be quiet by default (at the default threshold 0).

A compatibility trick and a speed-up trick

If you want to include calls to `Verbose` in a package of yours in order to debug code, but not use it otherwise, you might not want to load `R.utils` all the time, but only for debugging. To achieve this, the value of a reference variable to a `Verbose` class is always set to `TRUE`, cf. typically an Object reference has value `NA`. This makes it possible to use the reference variable as a first test before calling `Verbose` methods. Example:

```
foo <- function(..., verbose=FALSE) {
  # enter() will never be called if verbose==FALSE, thus no error.
  verbose && enter(verbose, "Loading")
}
```

Thus, `R.utils` is not required for `foo()`, but for `foo(verbose==Verbose(level=-1))` it is.

Moreover, if using the `NullVerbose` class for ignoring all verbose messages, the above trick will indeed speed up the code, because the value of a `NullVerbose` reference variable is always `FALSE`.

Extending the Verbose class

If extending this class, make sure to output messages via `*writeRaw()` or one of the other output methods (which in turn all call the former). This guarantees that `*writeRaw()` has full control of the output, e.g. this makes it possible to split output to standard output and to file.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[NullVerbose](#).

Examples

```

verbose <- Verbose(threshold=-1)

header(verbose, "A verbose writer example", padding=0)

enter(verbose, "Analysis A")
for (kk in 1:10) {
  printf(verbose, "step %d\n", kk)
  if (kk == 2) {
    cat(verbose, "Turning ON automatic timestamps")
    timestampOn(verbose);
  } else if (kk == 4) {
    timestampOff(verbose);
    cat(verbose, "Turned OFF automatic timestamps")
    cat(verbose, "Turning OFF verbose messages for steps ", kk, "-6")
    off(verbose)
  } else if (kk == 6) {
    on(verbose)
    cat(verbose, "Turned ON verbose messages just before step ", kk+1)
  }

  if (kk %in% c(5,8)) {
    enter(verbose, "Sub analysis ", kk)
    for (jj in c("i", "ii", "iii")) {
      cat(verbose, "part ", jj)
    }
    exit(verbose)
  }
}
cat(verbose, "All steps completed!")
exit(verbose)

ruler(verbose)
cat(verbose, "Demo of some other methods:")
str(verbose, c(a=1, b=2, c=3))
print(verbose, c(a=1, b=2, c=3))
summary(verbose, c(a=1, b=2, c=3))
evaluate(verbose, rnorm, n=3, mean=2, sd=3)

ruler(verbose)
newline(verbose)

```

whichVector.logical *Identifies TRUE elements in a logical vector*

Description

Identifies TRUE elements in a logical vector. This method is faster than `which()` for **logical vectors**, especially when there are no missing values.

Usage

```
## S3 method for class 'logical'
whichVector(x, na.rm=TRUE, use.names=TRUE, ...)
```

Arguments

| | |
|-----------|--|
| x | A logical vector of length N. |
| na.rm | If TRUE , missing values are treated as FALSE , otherwise they are returned as NA . |
| use.names | If TRUE , the names attribute is preserved, otherwise it is not return. |
| ... | Not used. |

Value

Returns an **integer vector** of length less or equal to N.

Benchmarking

Simple comparison on R v2.7.1 on Windows XP, show that this implementation can be more than twice as fast as `which()`, especially when there are no missing value (and `na.rm=FALSE`) is used.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

`which()`

Examples

```
# -----
# Simulate two large named logical vectors,
# one with missing values one with out
# -----
N <- 1e6;

# Vector #1
x <- sample(c(TRUE, FALSE), size=N, replace=TRUE);
names(x) <- seq_along(x);

# Vector #2
y <- x
y[sample(N, size=0.1*N)] <- NA;

# -----
# Validate consistency
# -----
stopifnot(identical(which(x), whichVector(x)));
stopifnot(identical(which(y), whichVector(y)));
```

```

# -----
# Benchmarking
# -----
# Number of iterations
K <- 5;

t1 <- 0;
for (kk in 1:K) {
  t1 <- t1 + system.time({ idxs1 <- which(x) });
};

t2 <- 0;
for (kk in 1:K) {
  t2 <- t2 + system.time({ idxs2 <- whichVector(x, na.rm=FALSE) });
};

cat(sprintf("whichVector(x, na.rm=FALSE)/which(x): %.2f\n", (t2/t1)[3]));
stopifnot(identical(idxs1, idxs2));

t1 <- 0;
for (kk in 1:K) {
  t1 <- t1 + system.time({ idxs1 <- which(y) });
};

t2 <- 0;
for (kk in 1:K) {
  t2 <- t2 + system.time({ idxs2 <- whichVector(y) });
};

cat(sprintf("whichVector(y)/which(y): %.2f\n", (t2/t1)[3]));
stopifnot(identical(idxs1, idxs2));

```

wrap.array

Reshape an array or a matrix by permuting and/or joining dimensions

Description

Reshape an array or a matrix by permuting and/or joining dimensions.

A useful application of this is to reshape a multidimensional [array](#) to a [matrix](#), which then can be saved to file using for instance `write.table()`.

Usage

```
## S3 method for class 'array'
wrap(x, map=list(NA), sep=".", ...)
```

Arguments

| | |
|-----|--|
| x | An array or a matrix . |
| map | A list of length equal to the number of dimensions in the reshaped array. Each element should be an integer vector specifying the dimensions to be joined in corresponding new dimension. One element may equal NA to indicate that that dimension should be a join of all non-specified (remaining) dimensions. Default is to wrap everything into a vector . |
| sep | A character pasting joined dimension names. |
| ... | Not used. |

Details

If the indices in `unlist(map)` is in a non-increasing order, `aperm()` will be called, which requires reshuffling of array elements in memory. In all other cases, the reshaping of the array does not require this, but only fast modifications of attributes `dim` and `dimnames`.

Value

Returns an [array](#) of length(`map`) dimensions, where the first dimension is of size `prod(map[[1]])`, the second `prod(map[[2]])`, and so on.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[*unwrap\(\)](#). See [aperm\(\)](#).

Examples

```
# Create a 3x2x3 array
dim <- c(3,2,3)
ndim <- length(dim)
dimnames <- list()
for (kk in 1:ndim)
  dimnames[[kk]] <- sprintf("%s%d", letters[kk], 1:dim[kk])
x <- 1:prod(dim)
x <- array(x, dim=dim, dimnames=dimnames)

cat("Array 'x':\n")
print(x)

cat("\nReshape 'x' to its identity:\n")
y <- wrap(x, map=list(1, 2, 3))
print(y)
# Assert correctness of reshaping
```

```

stopifnot(identical(y, x))

cat("\nReshape 'x' by swapping dimensions 2 and 3, i.e. aperm(x, perm=c(1,3,2)):\n")
y <- wrap(x, map=list(1, 3, 2))
print(y)
# Assert correctness of reshaping
stopifnot(identical(y, aperm(x, perm=c(1,3,2))))

cat("\nWrap 'x' to a matrix 'y' by keeping dimension 1 and joining the others:\n")
y <- wrap(x, map=list(1, NA))
print(y)
# Assert correctness of reshaping
for (aa in dimnames(x)[[1]]) {
  for (bb in dimnames(x)[[2]]) {
    for (cc in dimnames(x)[[3]]) {
      tt <- paste(bb, cc, sep=".")
      stopifnot(identical(y[aa,tt], x[aa,bb,cc]))
    }
  }
}

cat("\nUnwrap matrix 'y' back to array 'x':\n")
z <- unwrap(y)
print(z)
stopifnot(identical(z,x))

cat("\nWrap a matrix 'y' to a vector and back again:\n")
x <- matrix(1:8, nrow=2, dimnames=list(letters[1:2], 1:4))
y <- wrap(x)
z <- unwrap(y)
print(z)
stopifnot(identical(z,x))

cat("\nWrap and unwrap a randomly sized and shaped array 'x2':\n")
maxdim <- 5
dim <- sample(1:maxdim, size=sample(2:maxdim))
ndim <- length(dim)
dimnames <- list()
for (kk in 1:ndim)
  dimnames[[kk]] <- sprintf("%s%d", letters[kk], 1:dim[kk])
x2 <- 1:prod(dim)
x2 <- array(x2, dim=dim, dimnames=dimnames)

cat("\nArray 'x2':\n")
print(x2)

# Number of dimensions of wrapped array
ndim2 <- sample(1:(ndim-1), size=1)

```

```

# Create a random map for joining dimensions
splits <- NULL;
if (ndim > 2)
  splits <- sort(sample(2:(ndim-1), size=ndim2-1))
splits <- c(0, splits, ndim);
map <- list();
for (kk in 1:ndim2)
  map[[kk]] <- (splits[kk]+1):splits[kk+1];

cat("\nRandom 'map':\n")
print(map)

cat("\nArray 'y2':\n")
y2 <- wrap(x2, map=map)
print(y2)

cat("\nArray 'x2':\n")
z2 <- unwrap(y2)
print(z2)

stopifnot(identical(z2,x2))

```

| | |
|-------------------|--|
| writeBinFragments | <i>Writes binary data to disjoint sections of a connection or a file</i> |
|-------------------|--|

Description

Writes binary data to disjoint sections of a connection or a file.

Usage

```
## Default S3 method:
writeBinFragments(con, object, idxs, size=NA, ...)
```

Arguments

| | |
|--------|--|
| con | A connection or the pathname of an existing file. |
| object | A vector of objects to be written. |
| idxs | A vector of (non-duplicated) indices or a Nx2 matrix of N from-to index intervals specifying the elements to be read. Positions are always relative to the start of the file/connection. |
| size | The size of the data type to be read. If NA , the natural size of the data type is used. |
| ... | Additional arguments passed to writeBin() . |

Value

Returns nothing.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[readBinFragments\(\)](#).

Examples

```
## Not run: # See example(readBinFragments.connection)
```

```
writeDataFrame.data.frame
```

Writes a data.frame to tabular text file

Description

Writes a data.frame to tabular text file with an optional header.

Usage

```
## S3 method for class 'data.frame'
writeDataFrame(data, file, path=NULL, sep=" ", quote=FALSE, row.names=FALSE, col.names=TRUE, ..., header=FALSE)
```

Arguments

| | |
|---------------------------------------|--|
| data | A data.frame . |
| file | A connection or a filename to write to. |
| path | The directory where the file will be written. |
| sep, quote, row.names, col.names, ... | Additional arguments passed to write.table . |
| header | An optional named list of header rows to be written at the beginning of the file. If NULL , no header will be written. |
| createdBy, createdOn, nbrOfRows | If non- NULL , common header rows to be added to the header. |
| headerPrefix | A character string specifying the prefix of each header row. |
| headerSep | A character string specifying the character separating the header name and header values. |
| append | If TRUE , the output is appended to an existing file. |
| overwrite | If TRUE , an existing file is overwritten. |

Value

Returns (invisibly) the pathname to the file written (or the [connection](#) written to).

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

`write.table.readTable()`.

Index

*Topic **IO**

- createFileAtomically, 21
- createLink, 23
- createWindowsShortcut, 24
- dimNA< -, 36
- displayCode, 37
- evalWithTimeout, 43
- fileAccess, 47
- filePath, 49
- findSourceTraceback, 53
- getAbsolutePath, 54
- getParent, 55
- getRelativePath, 55
- hasUrlProtocol, 61
- isAbsolutePath, 67
- isDirectory, 68
- isFile, 69
- isOpen.character, 70
- isUrl, 74
- lastModified, 79
- LComments, 79
- listDirectory, 80
- loadObject, 81
- makedirs, 84
- NullVerbose, 85
- popBackupFile, 92
- popTemporaryFile, 93
- pushBackupFile, 97
- pushTemporaryFile, 98
- readBinFragments, 101
- readTable, 104
- readTableIndex, 106
- readWindowsShortcut, 107
- removeDirectory, 109
- resample, 109
- saveObject, 111
- Settings, 114
- SmartComments, 116
- sourceDirectory, 118

- sourceTo, 119

- TextStatusBar, 125

- touchFile, 129

- toUrl, 131

- VComments, 132

- Verbose, 135

- writeBinFragments, 143

- writeDataFrame.data.frame, 144

*Topic **attribute**

- intervalsToSeq.matrix, 66

- seqToHumanReadable, 112

- seqToIntervals, 112

*Topic **character**

- as.character.binmode, 9

- intToBin, 66

*Topic **classes**

- Arguments, 7

- Assert, 10

- FileProgressBar, 51

- GString, 56

- Java, 76

- LComments, 79

- NullVerbose, 85

- Options, 88

- ProgressBar, 95

- Settings, 114

- SmartComments, 116

- System, 124

- TextStatusBar, 125

- TimeoutException, 127

- VComments, 132

- Verbose, 135

*Topic **device**

- devDone, 27

- devEval, 27

- devGetLabel, 29

- devIsOpen, 30

- devList, 31

- devNew, 32

- devOff, 33
- devOptions, 33
- devSet, 35
- devSetLabel, 35
- env, 40
- eps, 41
- jpeg2, 78
- png2, 92
- toEPS, 129
- *Topic **error**
 - TimeoutException, 127
- *Topic **file**
 - bunzip2, 12
 - copyDirectory, 19
 - createLink, 23
 - createWindowsShortcut, 24
 - dimNA< -, 36
 - displayCode, 37
 - downloadFile.character, 39
 - gzip, 59
 - readWindowsShortcut, 107
 - touchFile, 129
- *Topic **logic**
 - isZero, 75
- *Topic **manip**
 - arrayIndex, 8
 - as.character.binmode, 9
 - dataFrame, 25
 - insert, 64
 - intToBin, 66
- *Topic **methods**
 - attachLocally.list, 11
 - callHooks.function, 14
 - downloadFile.character, 39
 - extract.array, 45
 - inAnyInterval.numeric, 63
 - intervalsToSeq.matrix, 66
 - isEof.connection, 68
 - isOpen.character, 70
 - mapToIntervals.numeric, 82
 - mergeIntervals.numeric, 83
 - TimeoutException, 127
 - unwrap.array, 131
 - whichVector.logical, 138
 - wrap.array, 140
 - writeDataFrame.data.frame, 144
- *Topic **package**
 - isPackageInstalled, 70
 - isPackageLoaded, 71
 - R.utils-package, 4
- *Topic **programming**
 - addFinalizerToLast, 6
 - Arguments, 7
 - as.character.binmode, 9
 - attachLocally.list, 11
 - bunzip2, 12
 - callHooks, 13
 - callHooks.function, 14
 - capitalize, 15
 - colClasses, 16
 - commandArgs, 17
 - countLines, 20
 - createFileAtomically, 21
 - detachPackage, 26
 - doCall, 38
 - downloadFile.character, 39
 - evalWithTimeout, 43
 - extract.array, 45
 - fileAccess, 47
 - finalizeSession, 52
 - findSourceTraceback, 53
 - getAbsolutePath, 54
 - getParent, 55
 - getRelativePath, 55
 - gzip, 59
 - hasUrlProtocol, 61
 - hpaste, 61
 - inAnyInterval.numeric, 63
 - intToBin, 66
 - isAbsolutePath, 67
 - isDirectory, 68
 - isFile, 69
 - isUrl, 74
 - lastModified, 79
 - LComments, 79
 - listDirectory, 80
 - loadObject, 81
 - mapToIntervals.numeric, 82
 - mergeIntervals.numeric, 83
 - mkdirs, 84
 - NullVerbose, 85
 - onGarbageCollect, 86
 - onSessionExit, 87
 - Options, 88
 - patchCode, 90
 - popBackupFile, 92

- popTemporaryFile, 93
- pushBackupFile, 97
- pushTemporaryFile, 98
- readRdHelp, 104
- removeDirectory, 109
- resample, 109
- resetWarnings, 110
- saveObject, 111
- setOption, 113
- Settings, 114
- SmartComments, 116
- sourceDirectory, 118
- sourceTo, 119
- splitByPattern, 121
- TextStatusBar, 125
- TimeoutException, 127
- touchFile, 129
- toUrl, 131
- unwrap.array, 131
- VComments, 132
- Verbose, 135
- whichVector.logical, 138
- wrap.array, 140
- *Topic **utilities**
 - arrayIndex, 8
 - attachLocally.list, 11
 - createFileAtomically, 21
 - dataFrame, 25
 - devDone, 27
 - devEval, 27
 - devGetLabel, 29
 - devIsOpen, 30
 - devList, 31
 - devNew, 32
 - devOff, 33
 - devOptions, 33
 - devSet, 35
 - devSetLabel, 35
 - env, 40
 - evalCapture, 42
 - inAnyInterval.numeric, 63
 - isOpen.character, 70
 - isPackageInstalled, 70
 - isPackageLoaded, 71
 - mapToIntervals.numeric, 82
 - mergeIntervals.numeric, 83
 - patchCode, 90
 - popBackupFile, 92
 - popTemporaryFile, 93
 - printf, 94
 - pushBackupFile, 97
 - pushTemporaryFile, 98
 - relibrary, 108
 - toEPS, 129
 - *intervalsToSeq, 113
 - *mapToIntervals, 64
 - *unwrap, 141
 - *wrap, 132
 - *writeRaw, 137
 - .Last, 88
 - .Machine, 75
- addFinalizerToLast, 6
- all.equal, 75
- aperm(), 141
- Arguments, 5, 7
- array, 46, 132, 140, 141
- arrayInd, 9
- arrayIndex, 8
- as.character, 57, 89, 96, 135
- as.character.binmode, 9
- as.double, 136
- as.list, 89
- as.logical, 136
- asByte, 77
- asInt, 77
- asLong, 77
- Assert, 5, 10
- asShort, 77
- attach, 11
- attachLocally, 11
- attachLocally.data.frame
 - (attachLocally.list), 11
- attachLocally.environment
 - (attachLocally.list), 11
- attachLocally.list, 11
- bunzip2, 12
- callHooks, 13, 15
- callHooks.function, 14
- callHooks.list, 13
- callHooks.list(callHooks.function), 14
- capitalize, 15
- capture, 136
- cat, 94, 136

- character, [9](#), [11](#), [13](#), [15](#), [16](#), [18](#), [20](#), [23](#),
 - [25–30](#), [32–35](#), [37–39](#), [41–44](#), [47](#), [50](#),
 - [54–57](#), [61](#), [62](#), [67–71](#), [74](#), [79](#), [81](#), [84](#),
 - [86](#), [87](#), [91](#), [93–95](#), [97](#), [100](#), [101](#),
 - [104–109](#), [112–114](#), [117](#), [119–121](#),
 - [125](#), [129–132](#), [135](#), [141](#), [144](#)
- check, [10](#)
- colClasses, [16](#), [105](#)
- commandArgs, [17](#), [17](#), [18](#)
- Comparison, [75](#)
- compile, [118](#)
- connection, [20](#), [37](#), [68–70](#), [81](#), [94](#), [101](#), [105](#),
 - [107](#), [111](#), [119](#), [135](#), [143](#), [144](#)
- connections, [12](#), [60](#), [70](#)
- convertComment, [118](#), [134](#)
- copyDirectory, [19](#)
- countLines, [20](#)
- createFileAtomically, [21](#)
- createLink, [23](#)
- createWindowsShortcut, [23](#), [24](#), [107](#)
- currentTimeMillis, [124](#)
- cut, [83](#)

- data.frame, [11](#), [25](#), [105](#), [144](#)
- dataFrame, [25](#)
- decapitalize (capitalize), [15](#)
- detach, [26](#)
- detachPackage, [26](#)
- dev.interactive, [27](#)
- dev.list, [31](#)
- dev.off, [33](#)
- dev.set, [35](#)
- dev2bitmap, [78](#), [92](#)
- devDone, [27](#), [32](#), [33](#), [35](#)
- devEval, [27](#), [32](#), [129](#)
- devGetLabel, [29](#), [36](#)
- devIsOpen, [30](#)
- devList, [29](#), [31](#), [36](#)
- devNew, [28](#), [32](#), [34](#)
- devOff, [27](#), [32](#), [33](#), [35](#)
- devOptions, [28](#), [33](#), [129](#)
- devSet, [35](#)
- devSetLabel, [29](#), [35](#)
- digest, [35](#), [111](#)
- dim, [36](#)
- dimNA< -, [36](#)
- dimNA<- (dimNA< -), [36](#)
- dir.create, [84](#)
- displayCode, [37](#)

- do.call, [38](#)
- doCall, [38](#)
- download.file, [39](#), [40](#)
- downloadFile.character, [39](#)
- downloadFile.duplicated, [72](#)

- enter, [136](#)
- env, [40](#)
- environment, [11](#), [28](#), [40](#), [44](#), [118](#), [119](#)
- eps, [41](#)
- equals, [89](#), [136](#)
- eval, [44](#)
- evalCapture, [42](#)
- evalq, [40](#)
- evaluate, [136](#)
- evalWithTimeout, [43](#)
- Exception, [127](#), [128](#)
- exit, [136](#)
- expression, [28](#), [40](#), [42](#)
- extract.array, [45](#)
- extract.default (extract.array), [45](#)
- extract.matrix (extract.array), [45](#)

- FALSE, [6](#), [11](#), [18](#), [26](#), [30](#), [41](#), [64](#), [67–70](#), [84](#), [93](#),
 - [94](#), [97](#), [99](#), [104](#), [105](#), [109](#), [119](#), [137](#),
 - [139](#)
- file.access, [47](#), [48](#)
- file.copy, [20](#)
- file.info, [68](#), [69](#), [79](#), [130](#)
- file.path, [49](#), [50](#)
- file.show, [37](#)
- file.symlink, [23](#)
- fileAccess, [47](#)
- filePath, [49](#), [107](#)
- FileProgressBar, [5](#), [51](#), [95](#)
- finalizeSession, [52](#), [87](#), [88](#)
- findGhostscript, [124](#)
- findGraphicsDevice, [124](#)
- findInterval, [83](#)
- findSettings, [114](#)
- findSourceTraceback, [53](#)
- flush, [126](#)
- function, [15](#), [21](#), [32](#), [86](#), [87](#), [132](#)

- getAbsolutePath, [54](#), [56](#)
- getBarString, [96](#)
- getBuiltinDate, [57](#)
- getBuiltinDatetime, [57](#)
- getBuiltinHostname, [57](#)

- getBuiltinOs, 57
- getBuiltinPid, 57
- getBuiltinRhome, 57
- getBuiltinRversion, 57
- getBuiltinTime, 57
- getBuiltinUsername, 57
- getCharacters, 7
- getDoubles, 7
- getEnvironment, 7
- getHostname, 124
- getIndices, 7
- getInstanceOf, 7
- getIntegers, 7
- getLabel, 126
- getLeaves, 89
- getLoadedPathname, 114
- getLogicals, 7
- getMessage, 128
- getNumerics, 7
- getOption, 89, 114
- getParent, 55
- getRaw, 57
- getReadablePathname, 8
- getReadablePathnames, 8
- getRegularExpression, 8
- getRelativePath, 55
- getThreshold, 136
- getTimestampFormat, 136
- getUsername, 124
- getVariableValue, 57
- getVector, 8
- getVerbose, 8
- getWritablePathname, 8
- GString, 5, 56, 135
- gunzip (gzip), 59
- gzip, 59

- hasOption, 89
- hasUrlProtocol, 61
- header, 136
- help, 104
- hpaste, 61

- inAnyInterval, 83, 84
- inAnyInterval.numeric, 63
- increase, 96
- Inf, 62
- inherits, 10
- insert, 64

- integer, 8, 20, 25, 31, 36, 39, 46, 47, 55, 62, 66, 67, 82, 106, 112, 113, 122, 139, 141
- intervalsToSeq.matrix, 66
- intToBin, 9, 66
- intToHex (intToBin), 66
- intToOct (intToBin), 66
- isAbsolutePath, 54, 56, 67
- isDirectory, 68, 69
- isDone, 96
- isEof.connection, 68
- isFile, 68, 69
- isMatrix, 10
- isModified, 114
- isOn, 85, 136
- isOpen.character, 70
- isPackageInstalled, 70, 71
- isPackageLoaded, 71, 71
- isReplicated, 72, 74
- isScalar, 10
- isSingle, 72, 73
- isUrl, 74
- isVector, 10
- isVisible, 85, 136
- isZero, 75

- Java, 5, 76
- jpeg2, 78

- lastModified, 79
- layout, 123
- LComments, 5, 79, 117, 133
- less, 136
- library, 91, 108
- list, 11, 15, 18, 28, 32, 34, 38, 46, 64, 88, 89, 104, 107, 132, 141, 144
- list.files, 81
- listDirectory, 80
- load, 82
- loadAnywhere, 115
- loadObject, 81, 111
- loadToEnv, 82
- loess, 38
- logical, 18, 21, 34, 39, 61, 64, 69, 71, 72, 74, 75, 93, 97, 99, 101, 105, 106, 108, 118, 135, 138, 139

- mapToIntervals, 64
- mapToIntervals.numeric, 82

- match, 83, 84
- matrix, 8, 46, 66, 82, 83, 101, 123, 132, 140, 141, 143
- mergeIntervals.numeric, 83
- mkdirs, 84
- mode, 101
- more, 136
- mtext, 122, 123
- MultiVerbose, 135

- NA, 36, 49, 82, 101, 137, 139, 141, 143
- names, 89
- nbrOfOptions, 89
- new.env, 40
- newline, 126, 136
- NULL, 11, 15, 18, 23, 32, 36, 39, 42, 44, 46, 55, 61, 71, 90, 111, 119, 144
- NullVerbose, 85, 135, 137
- numeric, 8, 27, 29, 30, 32, 33, 35–37, 44, 82, 83, 122, 125, 135

- Object, 7, 10, 51, 76, 79, 85, 88, 95, 114, 117, 124, 125, 127, 132, 135
- octmode, 9
- off, 136
- on, 136
- onGarbageCollect, 86
- onSessionExit, 7, 52, 53, 87, 115
- openBrowser, 124
- Options, 5, 88, 114
- options, 89, 91

- par, 32
- parse, 57, 118, 120
- parseDebian, 124
- paste, 62
- patchCode, 90
- pdf, 34
- Platform, 18
- png2, 92
- popBackupFile, 22, 92, 97
- popMessage, 126
- popTemporaryFile, 21, 22, 93, 98, 99
- postscript, 34, 42
- print, 57, 136
- printf, 94, 136
- ProgressBar, 5, 51, 95
- promptAndSave, 115
- pushBackupFile, 22, 93, 97
- pushState, 136
- pushTemporaryFile, 21, 22, 94, 98

- queryRCmdCheck, 100

- R.utils (R.utils-package), 4
- R.utils-package, 4
- read.table, 16, 25, 104, 105
- readBin, 101
- readBinFragments, 101, 144
- readByte, 77
- readInt, 77
- readline, 44
- readLines, 44
- readRdHelp, 104
- readShort, 77
- readTable, 104, 106, 145
- readTableIndex, 105, 106
- readUTF, 77
- readWindowsShortcut, 24, 50, 107
- relibrary, 91, 108
- remove, 51
- removeDirectory, 109
- replicates (isReplicated), 72
- require, 71
- resample, 109
- reset, 96, 118, 134
- resetWarnings, 110
- rle, 113
- ruler, 136

- sample, 109, 110
- sample.int, 110
- saveAnywhere, 115
- saveObject, 81, 82, 111
- seqToHumanReadable, 112
- seqToIntervals, 66, 112, 112
- setDefaultLevel, 136
- setLabel, 126
- setLabels, 126
- setMaxValue, 96
- setOption, 89, 113
- setProgress, 96
- setStepLength, 96
- setThreshold, 136
- setTicks, 96
- setTimeLimit, 44
- setTimestampFormat, 136
- Settings, 5, 88, 114

- setValue, 96
- showConnections, 70
- singles (isSingle), 73
- slice.index, 46
- SmartComments, 5, 79, 116, 132, 133
- source, 42, 53, 91, 119, 120
- sourceDirectory, 118, 120
- sourceTo, 42, 118, 119, 119
- splitByPattern, 121
- sprintf, 16, 94, 95, 125
- stext, 122
- str, 89, 136
- strsplit, 121, 122
- subplots, 123
- summary, 136
- Sys.setenv, 91
- sys.source, 120
- System, 5, 124

- TextStatusBar, 125
- TimeoutException, 44, 127
- timestamp, 136
- timestampOn, 136
- toCamelCase, 16
- toEPS, 129
- toPDF (toEPS), 129
- toPNG (toEPS), 129
- toSVG (toEPS), 129
- touchFile, 129
- toUrl, 131
- TRUE, 6, 12, 13, 18, 20, 21, 23, 24, 26, 28, 30, 34, 37–39, 42, 46, 47, 49, 50, 54, 55, 60, 66–70, 72, 81, 82, 84, 91, 93–95, 97, 99, 105, 107–109, 111, 118, 119, 123, 131, 132, 135, 137, 139, 144

- unlink, 109
- unwrap.array, 131
- unwrap.data.frame (unwrap.array), 131
- unwrap.default (unwrap.array), 131
- unwrap.matrix (unwrap.array), 131
- update, 51, 96, 126
- updateLabels, 126
- URLEncode, 131

- validate, 118, 134
- VComments, 5, 79, 80, 117, 118, 120, 132
- vector, 8, 11, 15, 16, 18, 20, 23, 25, 31, 36–38, 46, 61, 64, 70, 72, 74, 75, 81–83, 91, 94, 101, 105, 106, 110, 112, 113, 118, 120–122, 132, 138, 139, 141, 143
- Verbose, 5, 21, 39, 85, 93, 97, 99, 101, 105, 106, 118, 133, 135

- warning, 91
- warnings, 110, 136
- which, 9, 138, 139
- whichVector.logical, 138
- whichVector.matrix (whichVector.logical), 138
- wrap.array, 140
- wrap.data.frame (wrap.array), 140
- wrap.matrix (wrap.array), 140
- write.table, 144, 145
- writeBin, 143
- writeBinFragments, 102, 143
- writeByte, 77
- writeDataFrame.data.frame, 144
- writeInt, 77
- writeRaw, 86, 136
- writeShort, 77
- writeUTF, 77

- xfig, 34