Package 'rtape'

October 14, 2022

Maintainer Miron B. Kursa <M.Kursa@icm.edu.pl>

License GPL-2

Title Manage and manipulate large collections of R objects stored as tape-like files

Type Package

LazyLoad yes

Author Miron B. Kursa

Description Storing huge data in RData format causes problems because of the necessity to load the whole file to the memory in order to access and manipulate objects inside such file; rtape is a simple solution to this problem. The package contains several wrappers of R built-in serialize/unserialize mechanism allowing user to quickly append objects to a tape-like file and later iterate over them requiring only one copy of each stored object to reside in memory a time.

Version 2.2

URL http://mbq.me/rtape

Date 2011-05-10

Collate 'guessFileFormat.R' 'makeFileFormat.R' 'rtapeAdd.R' 'rtape_apply.R' 'rtapeAsList.R' 'rtapeFilter.R' 'rtapeLapply.R' 'rtapeRerecord.R' 'shared.R'

Repository CRAN

Date/Publication 2012-07-31 16:06:41

NeedsCompilation no

R topics documented:

`guessFileFormat` *Automatically pick proper tape file format.*

Description

Automatically pick proper tape file format.

Usage

```
guessFileFormat(fName)
```

Arguments

`fName`	Name of the the file to guess format of; if the file is not-existing, the function returns default file format.

Details

This function guesses the tape file format from the file header or assumes default (gzip) if given non-existing file name. Main package functions use this routine to automatically setup file format; if you really need to control it, see `makeFileFormat`.

Value

The function to be passed to the `fileFormat*` arguments of other `rtape`'s functions.

Author(s)

Miron B. Kursa <M.Kursa@icm.edu.pl>

makeFileFormat *Setting tape file format/compression.*

Description

Setting tape file format/compression.

Usage

```
makeFileFormat(compression="gz", compressionLevel=ifelse(compression == "bz", 9, 6))
```

Arguments

compression	Name of the compression algorithm; should be one of the "gz", "bz", "xz". Exact name should be given. See connections for details.
compressionLevel	compression parameter passed to gzfile, bzfile or xzfile.

Details

rtape uses R connections to store data; this function creates a function that is used to create a connection by the other rtape's functions. Changing its parameters allows advanced user to change compression format/level and thus control the speed/file size trade-off. The default values (gzip, 6th level of compression) should give performance similar to this of save.

Value

The function to be passed to the fileFormat* arguments of other rtape's functions.

Note

Effectively, this function is needed only to set up the format of the new, blank tape (i.e. in the first call to rtapeAdd or for altering compression along with tape reconstruction operations performed by rtapeRerecord or rtapeFilter); when dealing with already existing tapes, the guessFileFormat will recognise the right format from the file header.

Author(s)

Miron B. Kursa <M.Kursa@icm.edu.pl>

rtapeAdd *Add object to the tape.*

Description

Add object to the tape.

Usage

```
rtapeAdd(fName, what, skipNULLs=FALSE, fileFormat=guessFileFormat(fName),
    safe=FALSE, retryTime=0.1)
```

Arguments

fName	Name of the tape file.
what	Object to be stored.
skipNULLs	If true and what is NULL, nothing is written to the tape.
fileFormat	File format; should be left default. See guessFileFormat and makeFileFormat for details.
safe	If "try" or "retry", rtape will use dirlock to ensure that no other rtape safe appending is in progress. In case of conflict, the function in "try" mode immediately returns FALSE and does not try again, while in "retry" mode it sleeps retryTime seconds and tries again till the dirlock is opened.
retryTime	If safe is "retry", this parameter sets the interval between writing attempts. Expressed in seconds.

Details

This function serializes and appends a given object on the end of the tape file. If the tape file does not exists, it is created.

Note

Remember to use the same fileFormat value to all writes to a certain tape (or use default format guesser to guarantee this); if not, the tape will become unreadable. For the same reason, don't try to put custom headers/footers or append other data inside tape stream. This function is thread/process safe only if you use safe mode. However, in this case it may jam on a broken dirlock (for instance when the locking R process crashed during write); you may fix this problem manually by removing the locking dir. Its name is always .rtape_<tape file name>_lock. Waiting in retry mode is performed via Sys.sleep, so it is not a busy wait.

Author(s)

Miron B. Kursa <M.Kursa@icm.edu.pl>

Examples

```
unlink('tmp.tape')
#Record something on the tape
rtapeAdd('tmp.tape',iris)
rtapeAdd('tmp.tape',sin(1:10))
#Read whole tape to the list, so we could examine it
rtapeAsList('tmp.tape')->stored
print(stored)
unlink('tmp.tape')
```

`rtapeAsList` *Load the whole tape as a list.*

Description

Load the whole tape as a list.

Usage

```
rtapeAsList(fNames)
```

Arguments

`fNames`	Name of the tape file to read; if this argument is a vector of several names, function behaves as reading a single tape made of all those tapes joined in a given order.

Details

This function reads are the objects from the tape, in the order they were written on it, and returns them as a list.

Value

A list containing all the objects stored on the tape.

Author(s)

Miron B. Kursa <M.Kursa@icm.edu.pl>

Examples

```
unlink('tmp.tape')
#Record something on the tape
rtapeAdd('tmp.tape',iris)
rtapeAdd('tmp.tape',sin(1:10))
#Read whole tape to the list, so we could examine it
rtapeAsList('tmp.tape')->stored
print(stored)
unlink('tmp.tape')
```

`rtapeFilter` *Rerecord the tape dropping certain objects.*

Description

Rerecord the tape dropping certain objects.

Usage

```
rtapeFilter(FUN, fNamesIn, fNameOut=fNamesIn, moreArgs,
    fileFormatOut=guessFileFormat(fNameOut))
```

Arguments

`FUN`	Callback function which gets the current object and returns a boolean value that directs `rtape` to either keep (for `TRUE`) or discard it.
`fNamesIn`	Name of the tape file to read; if this argument is a vector of several names, function behaves as reading a single tape made of all those tapes joined in a given order.
`fNameOut`	Name of the tape to which store the output of filtering; if this file is one of the input files, this file is overwritten with the output; otherwise the output is appended to this tape. This must be a one-element vector.
`moreArgs`	Additional arguments to `FUN`, given as a list.
`fileFormatOut`	File format; should be left default. See `guessFileFormat` and `makeFileFormat` for details.

Details

This function reads the objects from one tape, executes a callback function on them and leaves/appends to the other tape only those for which callback returned `TRUE`.

Note

Overwriting is NOT realised in place, rather by a creation of a temporary file and then using it to overwrite the filtered tape.

Author(s)

Miron B. Kursa <M.Kursa@icm.edu.pl>

Examples

```
unlink(c('tmp.tape'))
#Record something
for(i in 1:10) rtapeAdd('tmp.tape',i)

#Discard even numbers
```

```
rtapeFilter(function(x) (x%%2)==1,'tmp.tape')

#Check it out
unlist(rtapeAsList('tmp.tape'))->A
print(A);
stopifnot(all(A==c(1,3,5,7,9)))

#Time to clean up
unlink(c('tmp.tape'))
```

`rtapeLapply` *Iterate over tape, gathering results.*

Description

Iterate over tape, gathering results.

Usage

```
rtapeLapply(fNames, FUN, ...)
```

Arguments

`fNames`	Name of the tape file to read; if this argument is a vector of several names, function behaves as reading a single tape made of all those tapes joined in a given order.
`FUN`	Callback function.
`...`	Additional parameters to `FUN`.

Details

This function read the tape from the oldest to the newest writes and executes the callback function on each read object. Logically, it is an equivalent to `lapply(rtapeAsList(fName),FUN,...)`, but it is optimized to store only the currently processed object in the memory.

Value

A list containing results of `FUN` calls.

Author(s)

Miron B. Kursa <M.Kursa@icm.edu.pl>

Examples

```
unlink('tmp.tape')
#Record something on the tape
rtapeAdd('tmp.tape',runif(3))
rtapeAdd('tmp.tape',rnorm(3))

#Print tape contents
rtape_apply('tmp.tape',print)
unlink('tmp.tape')
```

`rtapeRerecord` *Rerecord the tape.*

Description

Rerecord the tape.

Usage

```
rtapeRerecord(FUN, fNamesIn, fNameOut=fNamesIn, moreArgs, skipNULLs=FALSE,
    fileFormatOut=guessFileFormat(fNameOut))
```

Arguments

`FUN`	Callback function which transforms the objects.
`fNamesIn`	Name of the tape file to read; if this argument is a vector of several names, function behaves as reading a single tape made of all those tapes joined in a given order.
`fNameOut`	Name of the tape to which store the output of filtering; if this file is one of the input files, this file is overwritten with the output; otherwise the output is appended to this tape. This must be a one-element vector.
`moreArgs`	Additional arguments to `FUN`, given as a list.
`skipNULLs`	If true, all the `NULL`s returned by `FUN` are not appended to the output. Useful to remove some objects from the tape; see `rtapeFilter` for convenience function to do just this task.
`fileFormatOut`	File format; should be left default. See `guessFileFormat` and `makeFileFormat` for details.

Details

This function reads the objects from one tape, executes a callback function on them and updates them with/appends to the other tape the objects that the callback has returned.

Note

Overwriting is NOT realised in place, rather by a creation of a temporary file and then using it to overwrite the filtered tape.

Author(s)

Miron B. Kursa <M.Kursa@icm.edu.pl>

Examples

```
unlink(c('tmp.tape','tmp2.tape'))
#Record something
for(i in 1:10) rtapeAdd('tmp.tape',i)

#Multiply each object by two
rtapeRerecord('*','tmp.tape','tmp2.tape',moreArgs=list(2))

#Check it out
unlist(rtapeAsList('tmp.tape'))->A
B<-unlist(rtapeAsList('tmp2.tape'))
print(A);print(B)
stopifnot(all(A==B/2))

#Now do the same in-place:
rtapeRerecord('*','tmp.tape',moreArgs=list(2))
unlist(rtapeAsList('tmp.tape'))->B2
stopifnot(all(A==B2/2))

#Time to clean up
unlink(c('tmp.tape','tmp2.tape'))
```

---

rtape_apply *Iterate over tape, discarding results.*

Description

Iterate over tape, discarding results.

Usage

```
rtape_apply(fNames, FUN, ...)
```

Arguments

fNames	Name of the tape file to read; if this argument is a vector of several names, function behaves as reading a single tape made of all those tapes joined in a given order.
FUN	Callback function.
...	Additional parameters to FUN.

Details

This function read the tape from the oldest to the newest writes and executes the callback function on each read object. Logically, it is an equivalent to `ignore<-lapply(rtapeAsList(fName),FUN,...)`, but it is optimized to store only the currently processed object in the memory and to discard `FUN` results as soon as they appear.

Author(s)

Miron B. Kursa <M.Kursa@icm.edu.pl>

Examples

```
unlink('tmp.tape')
#Record something on the tape
rtapeAdd('tmp.tape',runif(3))
rtapeAdd('tmp.tape',rnorm(3))

#Print tape contents
rtape_apply('tmp.tape',print)
unlink('tmp.tape')
```

Index