Package 'synchronicity'

May 9, 2026

Version 1.3.10

Title Boost Mutex Functionality in R

Imports methods, bigmemory.sri, Rcpp, uuid

LinkingTo BH, Rcpp

Description Boost mutex functionality in R.

License LGPL-2 | Apache License 2.0

URL http://www.bigmemory.org

LazyLoad yes

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation yes

Author Michael J. Kane [aut, cre] (ORCID:
<https://orcid.org/0000-0003-1899-6662>)

Maintainer Michael J. Kane <bigmemoryauthors@gmail.com>

Repository CRAN

Date/Publication 2024-01-10 17:10:01 UTC

Contents

`attach.mutex` *Attach to an existing mutex.*

Description

Attach to an existing mutex using either a file or description object

Usage

```
attach.mutex(obj, ...)

## S4 method for signature 'character'
attach.mutex(obj, ...)

## S4 method for signature 'boost.mutex.descriptor'
attach.mutex(obj, ...)
```

Arguments

`obj`	the descriptor object.
`...`	other arguments needed by attach.

Value

A mutex.

`boost.mutex` *Create a boost.mutex object*

Description

This function creates a `boost.mutex` object.

Usage

```
boost.mutex(sharedName = NULL, timeout = NULL, create = TRUE)
```

Arguments

`sharedName`	The name of the shared resource corresponding to the mutex. By default a universal unique identifier is supplied.
`timeout`	The amount of time (in seconds) that the mutex should try to attempt to get a lock. By default no timeout is supplied and the mutex will attempt to acquire the lock indefinitely.
`create`	Should the mutex be created or are we attaching to an existing on. Default is `TRUE`.

Examples

```
# Create a boost.mutex object with default resource name and no timeout.
x = boost.mutex()
rm(x)
gc()
```

`boost.mutex-class` *The boost.mutex class*

Description

The boost.mutex class

`boost.mutex.descriptor-class`
An S4 class holding boost.mutex description information.

Description

Objects of class description allow users to "attach" to existing mutexes within or across processes.

Slots

`description` the list of description information.

`describe,boost.mutex-method`
Describe the boost.mutex object

Description

The information required to "attach" to an existing mutex object.

Usage

```
## S4 method for signature 'boost.mutex'
describe(x)
```

Arguments

`x`	the boost mutex object to describe.

description *Accessor for descriptor objects*

Description

Retrieve the list of description information from a descriptor object.

Usage

```
description(x)

## S4 method for signature 'descriptor'
description(x)
```

Arguments

x	the descriptor object.

Value

a list of description information.

descriptor-class *An S4 class holding mutex description information.*

Description

Objects of class description allow users to "attach" to existing mutexes within or across processes.

Slots

`description` the list of description information.

is.timed *Timeout operations for boost.mutex objects*

Description

The `is.timed` function tells if a `boost.mutex` object has a timeout. The `timeout` function tells how long a mutex will wait for a timeout.

Usage

```
is.timed(m)

## S4 method for signature 'boost.mutex'
is.timed(m)
```

Arguments

`m`	a `boost.mutex` object to get timeout information for.

Value

`is.timed` returns `TRUE` if the object has a timeout and `FALSE` otherwise. If a timeout has been set `timeout` returns the number of seconds a `boost.mutex` object will attempt to acquire a lock and `NULL` otherwise.

Examples

```
x = boost.mutex(timeout=5)
y = boost.mutex()
print(is.timed(x))
print(is.timed(y))
print(timeout(x))
print(timeout(y))
```

lock *Lock and Unlock a Mutex*

Description

The `lock` and `unlock` functions allow a user to specify exclusive or shared access to a resource.

Usage

```
lock(m, ...)

lock.shared(m, ...)

unlock(m, ...)

unlock.shared(m, ...)
```

Arguments

m	a mutex.
...	options associated with the mutex being used including block which forces the mutex to return immediately after trying to acquire a lock.

Details

A call to lock gives exclusive access to a resource; no other mutex may acquire a lock. A call to to lock.shared allows other mutexes to acquire a shared lock on the resource. When shared lock is called while a exclusive lock has been acquired, the shared lock will block until the exclusive lock is release. Likewise, if an exclusive lock is called while a shared lock has been acquired, the exclusive lock will block until the shared lock is released.

Value

The function returns TRUE if the lock is successfully called and FALSE otherwise.

Examples

```
m = boost.mutex()
lock(m)
# Some code that needs to be synchronized...
unlock(m)
```

---

mutex-class *The boost.mutex class*

Description

The boost.mutex class

shared *Is it a shared mutex?*

Description

Tells the user if a mutex is a shared mutex. If it is not then it must be a write (exclusive) mutex.

Usage

```
shared(m)

## S4 method for signature 'boost.mutex'
shared(m)
```

Arguments

m	the mutex

Value

TRUE if the mutex is shared, FALSE otherwise.

shared.name *The name of a mutex's shared resource*

Description

This function returns the shared resource associated with a `boost.mutex` object.

Usage

```
shared.name(m)
```

Arguments

m	a `boost.mutex` object

Value

A string specifying the shared resource associated with the given `boost.mutex` object.

Examples

```
x = boost.mutex()
print(shared.name(x))
```

uuid *Create a universal unique identifier.*

Description

This function creates an identifier that will be (with high probability) unique on a single machine or group of machines.

Usage

```
uuid()
```

Value

A unique string.

Examples

```
print(uuid())
```

Index