Package ‘synchronicity’

August 29, 2016

Version 1.1.9.1
Date 2015-12-25
Title Boost Mutex Functionality in R
Author Michael J. Kane <kaneplusplus@gmail.com>
Maintainer Michael J. Kane <bigmemoryauthors@gmail.com>
Contact Michael J. Kane <bigmemoryauthors@gmail.com>
Imports methods, bigmemory.sri, Rcpp
LinkingTo BH, Rcpp
Description Boost mutex functionality in R.
License LGPL-3 | Apache License 2.0
URL http://www.bigmemory.org
LazyLoad yes
RoxygenNote 5.0.1
NeedsCompilation yes
OS_type unix
Repository CRAN
Date/Publication 2016-02-17 16:15:37

R topics documented:

synchronicity-package .............................................. 2
attach.mutex ......................................................... 3
boost.mutex .......................................................... 3
boost.mutex-class ..................................................... 4
boost.mutex.descriptor-class ......................................... 5
describe .............................................................. 5
description ........................................................... 6
descriptor-class ......................................................... 6
is.timed, timeout ....................................................... 7
lock, lock.shared, unlock ............................................. 8
This package provides support for synchronization via mutexes and may eventually support interprocess communication (ipc) and message passing.

Details

Package: synchronicity
Type: Package
Version: 1.1.9
Date: 2015-11-09
License: LGPL-3 | Apache License 2.0
OS_type: unix
URL: http://www.bigmemory.org
LazyLoad: yes

Author(s)

Michael J. Kane <bigmemoryauthors@gmail.com>

References

Boost Interprocess Library: http://www.boost.org/.

Examples

# No examples are provided here.
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)

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.
Value

This function returns a `boost::mutex` object.

Author(s)

Michael J. Kane <bigmemoryauthors@gmail.com>

See Also

`synchronicity`

Examples

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

---

### `boost::mutex-class`  
*Class “boost::mutex”*

Description

The `boost::mutex` class provides an R interface to the mutex functionality implemented in the Boost C++ library.

Objects from the Class

Unlike many R objects, objects should not be created by calls of the form `new("boost::mutex", ...)`. The function `boost::mutex()` is intended for the user.

Slots

- `isRead`: This is used internally to maintain state information and should not be touched by a user.
- `mutexInfoAddr`: Object of class "externalptr" which keeps track of information relevant to the mutex.

Extends

Class "mutex", directly.

Methods

- `describe` signature(`x = "boost::mutex"`): ...
- `is.timed` signature(`m = "boost::mutex"`): ...
- `lock.shared` signature(`m = "boost::mutex"`): ...
- `lock` signature(`m = "boost::mutex"`): ...
- `shared.name` signature(`m = "boost::mutex"`): ...
- `timeout` signature(`m = "boost::mutex"`): ...
- `unlock` signature(`m = "boost::mutex"`): ...
Author(s)

Michael J. Kane <bigmemoryauthors@gmail.com>

See Also

boost.mutex

Examples

showClass("boost.mutex")

Description

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

Slots

description the list of description information.

Description

The describe function returns information that is needed to “connect” to a mutex from another process. This connection is performed by the attach.mutex function.

Usage

describe(x)

Arguments

x a boost.mutex object

Value

The describe function returns a boost.mutex.descriptor object.

Author(s)

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

```r
m = boost_mutex()
m = attach_mutex(describe(m))
# Now, both m and mm specify the same mutex.
```

---

**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

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

```cpp
is.timed(m)
timeout(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.

Author(s)

Michael J. Kane <bigmemoryauthors@gmail.com>

See Also

synchronicity

Examples

```cpp
x = boost.mutex(timeout=5)
y = boost.mutex()
print(is.timed(x))
print(is.timed(x))
print(timeout(x))
print(timeout(y))
```
lock, lock.shared, unlock

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, ...)

Arguments

m an object derived from class 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 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

Author(s)

Michael J. Kane <bigmemoryauthors@gmail.com>

Examples

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

Is it a read (shared) mutex?

**Description**

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

**Usage**

```r
read(m)
```

## S4 method for signature 'boost.mutex'

```r
read(m)
```

**Arguments**

- `m` the mutex

**Value**

TRUE if the mutex is read (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**

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

**Arguments**

- `m` a `boost.mutex` object

**Value**

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

**Author(s)**

Michael J. Kane <bigmemoryauthors@gmail.com>
See Also

synchronicity

Examples

```python
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**

```python
uuid()
```

**Details**

The functions uses the boost uuid functionality.

**Value**

A unique string.

**Author(s)**

Michael J. Kane <bigmemoryauthors@gmail.com>

**References**

http://www.boost.org/doc/libs/1_42_0/libs/uuid/uuid.html

**Examples**

```python
print(uuid())
print(uuid())
```
Index

*Topic **classes**
boost.mutex-class, 4

*Topic **misc**
boost.mutex, 3
describe, 5
is.timed, timeout, 7
lock, lock.shared, unlock, 8
shared.name, 9

*Topic **programming**
boost.mutex, 3
describe, 5
is.timed, timeout, 7
lock, lock.shared, unlock, 8
shared.name, 9

attach.mutex, 3
attach.mutex, boost.mutex.descriptor-method (attach.mutex), 3
attach.mutex, character-method (attach.mutex), 3

boost.mutex, 3, 5
boost.mutex-class, 4
boost.mutex.descriptor-class, 5
describe, 5
describe, boost.mutex-method (boost.mutex-class), 4
description, 6
description, descriptor-method (description), 6
descriptor-class, 6

is.timed (is.timed, timeout), 7
is.timed, timeout, 7
is.timed, boost.mutex-method (boost.mutex-class), 4

lock (lock, lock.shared, unlock), 8
lock, lock.shared, unlock, 8

lock, boost.mutex-method (boost.mutex-class), 4
lock.shared, boost.mutex-method (boost.mutex-class), 4

mutex, 4
mutex-class (boost.mutex-class), 4

read, 9
read, boost.mutex-method (read), 9
shared.name, 9
shared.name, boost.mutex-method (boost.mutex-class), 4
synchronicity, 4, 7, 10
synchronicity (synchronicity-package), 2
synchronicity-package, 2
timeout (is.timed, timeout), 7
timeout, boost.mutex-method (boost.mutex-class), 4

unlock (lock, lock.shared, unlock), 8
unlock, boost.mutex-method (boost.mutex-class), 4

uuid, 10