NAME

ost::Conditional −

A conditional variable synchcronization object for one to one and one to many signal and control events between processes.

SYNOPSIS

```
#include <thread.h>
```

Inherited by **ost::Buffer**.

Public Member Functions

Conditional (const char *id=NULL)
Create an instance of a conditional.

virtual **˜Conditional** ()
Destroy the conditional.

void **signal** (bool broadcast)
Signal a conditional object and a waiting threads.

bool **wait** (**timeout_t** timer=0, bool locked=false)
Wait to be signaled from another thread.

void **enterMutex** (void)
Locks the conditional's mutex for this thread.

void **lock** (void)
In the future we will use lock in place of enterMutex since the conditional composite is not a recursive mutex, and hence using enterMutex may cause confusion in expectation with the behavior of the **Mutex** *class.*

bool **tryEnterMutex** (void)
Tries to lock the conditional for the current thread.

bool **test** (void)

void **leaveMutex** (void)
Leaving a mutex frees that mutex for use by another thread.

void **unlock** (void)

Detailed Description

A conditional variable synchcronization object for one to one and one to many signal and control events between processes.

Conditional variables may wait for and receive signals to notify when to resume or perform operations. Multiple waiting threads may be woken with a broadcast signal.

Warning:
While this class inherits from **Mutex**, the methods of the class **Conditional** just handle the system conditional variable, so the user is responsible for calling enterMutex and leaveMutex so as to avoid race conditions. Another thing to note is that if you have several threads waiting on one condition, not uncommon in thread pools, each thread must take care to manually unlock the mutex if cancellation occurs. Otherwise the first thread cancelled will deadlock the rest of the thread.

Author:
David Sugar conditional.

Constructor & Destructor Documentation

ost::Conditional::Conditional (const char * id = `NULL`)
Create an instance of a conditional. **Parameters:**
id name of conditional, optional for deadlock testing.

virtual ost::Conditional::˜Conditional () `[virtual]`
Destroy the conditional.

Member Function Documentation

void ost::Conditional::enterMutex (void)
Locks the conditional's mutex for this thread. Remember that Conditional's mutex is NOT a recursive mutex!

See also:
leaveMutex

void ost::Conditional::leaveMutex (void)
Leaving a mutex frees that mutex for use by another thread. **See also:**
enterMutex

void ost::Conditional::lock (void) `[inline]`
In the future we will use lock in place of enterMutex since the conditional composite is not a recursive mutex, and hence using enterMutex may cause confusion in expectation with the behavior of the **Mutex** class. **See also:**
enterMutex

void ost::Conditional::signal (bool broadcast)
Signal a conditional object and a waiting threads. **Parameters:**
broadcast this signal to all waiting threads if true.

bool ost::Conditional::test (void) `[inline]`
bool ost::Conditional::tryEnterMutex (void)
Tries to lock the conditional for the current thread. Behaves like **enterMutex** , except that it doesn't block the calling thread.

Returns:
true if locking the mutex was succesful otherwise false

See also:
enterMutex

leaveMutex

void ost::Conditional::unlock (void) `[inline]`
bool ost::Conditional::wait (timeout_t timer = `0`**, bool locked** = `false`**)**
Wait to be signaled from another thread. **Parameters:**
timer time period to wait.
locked flag if already locked the mutex.

Author

Generated automatically by Doxygen for GNU CommonC++ from the source code.