**NAME**

smutex_t – Shore Mutex Class

**SYNOPSIS**

```
#include <sthread.h>

/*
 *  Mutual Exclusion
 */
class smutex_t : public sthread_named_base_t {
public:
    NORET               smutex_t(const char* name = 0);
    NORET               ~smutex_t();
    w_rc_t              acquire(int4_t timeout = WAIT_FOREVER);
    void                release();
    bool                is_mine() const;
    bool                is_locked() const;
};
```

**DESCRIPTION**

An application can use a mutex to protect a shared resource from simultaneous access by multiple threads. Mutexes are blocking locks. Only one thread at a time is allowed to hold a mutex; additional threads trying to acquire the mutex will block until the mutex is free.

In order to access a shared resource, a thread must own the mutex that is protecting the resource. Ownership is obtained by *acquiring* the mutex. If another thread already owns the mutex, the requesting thread is blocked. If the mutex is not owned, ownership is granted to the requesting thread so it can access the shared resource. When a thread is finished using the shared resource, it relinquish its ownership of the mutex, thereby enabling another thread to gain ownership.

**smutex_t(name)**

The constructor creates a mutex. The *name* parameter is stored in the mutex for debugging purposes.

**~smutex_t()**

The destructor checks to make sure no thread holds the mutex. If one does, it is a fatal error.

**acquire(timeout)**

The **acquire** method gives the current thread ownership of the mutex. If the mutex is not owned, **acquire** sets its ownership and returns immediately with success. If the mutex is already owned, the calling thread is blocked until either the owning thread releases the mutex, or the specified *timeout* limit is reached.

If more than one thread is blocked on the mutex, the thread with the highest priority level is the first to be unblocked and given ownership of the mutex. If more than 1 of the waiting threads have the same priority, then FIFO ordering is used to determine which thread is unblocked and given ownership.

*Warning:* a thread calling **acquire** twice in a row (i.e.,without releasing the mutex before the second acquire) will be blocked forever.

**release()**

The **release** method releases ownership of the mutex.

**ERRORS**

TODO.

**EXAMPLES**

TODO.

**VERSION**

This manual page applies to Version 2.0 of the Shore Storage Manager.

**SPONSORSHIP**

The Shore project is sponsored by the Advanced Research Project Agency, ARPA order number 018 (formerly 8230), monitored by the U.S. Army Research Laboratory under contract DAAB07-91-C-Q518. Further funding for this work was provided by DARPA through Rome Research Laboratory Contract No. F30602-97-2-0247.

**COPYRIGHT**

Copyright (c) 1994-1999, Computer Sciences Department, University of Wisconsin -- Madison. All Rights Reserved.

**SEE ALSO**

**errors(sthread), sthread_t(sthread), scond_t(sthread), sevsem_t(sthread), file_handlers(sthread), intro(sthread).**