

**NAME**

latch\_t – Data Vector Classes

**SYNOPSIS**

```
#include <latch.h>

// these are defined outside of class latch_t due to bugs in
// some C++ compilers
enum latch_mode_t { LATCH_NL = 0, LATCH_SH = 1, LATCH_EX = 2 };

class latch_t : public sthread_named_base_t {

public:
    NORET          latch_t(const char* const desc = 0);
    NORET          ~latch_t()    {};
#ifdef DEBUG
    friend ostream& operator<<(ostream&, const latch_t& l);
#endif

    inline void      setname(const char *const desc);
    w_rc_t          acquire(
        latch_mode_t      m,
        int               timeout = sthread_base_t::WAIT_FOREVER);
    w_rc_t          upgrade_if_not_block(
        bool&             would_block);
    void            release();
    bool            is_locked() const;
    bool            is_hot() const;
    int             lock_cnt() const;
    int             num_holders() const;
    int             held_by(const sthread_t* t) const;
    bool            is_mine() const;

private: // disabled methods
    NORET          latch_t(const latch_t&);
    latch_t&       operator=(const latch_t&);
};
```

**DESCRIPTION**

Latches are a read/write synchronization mechanism for threads, as opposed to locks which are used for synchronizing transactions. Latches are much lighter weight than locks, have no symbolic names, and have no deadlock detection.

**`latch_t(desc)`**

The constructor for a latch takes an string descriptor (name) for the latch. This name is useful for debugging and for the output operator.

**`setname(desc)`**

This method is used to change the descriptor associated with a latch.

**acquire(mode, timeout)**

The **acquire** method attempts to acquire the latch (for the thread that is running) in the desired *mode*. Valid values for *mode* are: **LATCH\_SH** indicating shared mode and **LATCH\_EX** indicating exclusive mode. If the latch cannot be acquired within the *timeout*, the method will return with a **stTIMEOUT** error. The number of times a latch is acquired by a thread is counted, so a corresponding call to **release** must be made for every successful call to **acquire**.

**upgrade\_if\_not\_block(would\_block)**

The **upgrade\_if\_not\_block** method attempts to upgrade the latch from shared to exclusive mode. If the upgrade would cause the thread to block, then the upgrade is not performed and *would\_block* is set to **true**.

**release()**

The **release** method releases the latch for the thread that calls it.

**is\_locked()**

The **is\_locked** method returns **true** if any thread holds the latch.

**is\_hot()**

The **is\_hot** method returns **true** if any thread is waiting for the latch.

**lock\_cnt()**

The **lock\_cnt** method returns the total number of outstanding acquires.

**num\_holders()**

The **num\_holders** method returns the total number of threads holding the latch.

**held\_by(thread)**

The **held\_by** method returns the number of times the latch is held by the thread *thread*.

**is\_mine()**

The **is\_mine** method returns **true** if the calling thread hold the latch in exclusive mode.

**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.

LATCH\_T (COMMON)

LATCH\_T (COMMON)

**COPYRIGHT**

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

**SEE ALSO**

**rsrc(common), lock(ssm), intro(common).**

**BUGS**

There is a limitation of four share-mode (LATCH\_SH) holders for a latch. Any additional threads attempting to acquire the latch will block.