NAME

sevsem_t – Shore Event Semaphore Class

SYNOPSIS

```
#include <sthread.h>

/*
 *  Event semaphore
 */
class sevsem_t : public sthread_base_t {
public:
    NORET               sevsem_t(
      int                   is_post = 0,
      const char*           name = 0);
    NORET               ~sevsem_t();

    w_rc_t              post();
    w_rc_t              reset(int* pcnt = 0);
    w_rc_t              wait(int4_t timeout = WAIT_FOREVER);
    void                query(int& pcnt);

    void                setname(
      const char*           n1,
      const char*           n2 = 0);
};
```

DESCRIPTION

An event semaphore functions as a counting semaphore in that a thread can use an event semaphore to trigger execution of other threads. This is useful if, for example, one thread (producer) provides data to many other threads (consumer). Using an event semaphore frees the consumers from the trouble of polling to determine when new data is available.

sevsem_t(is_post, name)

The constructor creates an event semaphore. The *name* parameter is stored in the semaphore for debugging purposes. If *is_post* is **true,** the event semaphore is set to posted right after construction. Otherwise, the event semaphore is reset after construction.

~sevsem_t()

reset(pcnt)

The **reset** method resets the event semaphore (i.e.,reset post count to 0), and returns the number of times the semaphore was posted since it was last reset in *pcnt.* All threads that subsequently wait on this semaphore will be blocked.

post()

The **post** method posts the semaphore and increments the post count. All threads waiting on this semaphore are unblocked and resume execution. Threads that wait on the semaphore after the semaphore has been posted and before the next time it is reset, will not be blocked. If the semaphore is subsequently reset, threads that calls **wait()** will again be blocked.

If the semaphore is reset when **post** is called, the semaphore is posted and the post count is set to 1. If the semaphore is already posted when *post* is called, the post count is incremented, and an error, **stSEMPOSTED,** is returned to the calling thread.

wait(timeout)

The **wait** method waits on the event semaphore. If the semaphore is already posted, **wait** returns immediately and the thread continues to run. Otherwise, the thread is blocked until the semaphore is posted. Note that **wait** does not decrement the post count; only **reset** modifies the post count in any way.

query(pcnt)

The **query** method returns, in *pcnt*, the number of times the semaphore was posted since it was last reset.

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.

SEE ALSO

errors(sthread), sthread_t(sthread), smutex_t(sthread), scond_t(sthread), file_handlers(sthread), intro(sthread).