NAME

seqc_consistent, **seqc_read**, **seqc_write_begin**, **seqc_write_end** - lockless read algorithm

SYNOPSIS

#include <sys/seqc.h>

void
seqc_write_begin(*seqc_t *seqcp*);

void
seqc_write_end(*seqc_t *seqcp*);

seqc_t
seqc_read(*seqc_t *seqcp*);

seqc_t
seqc_consistent(*const seqc_t *seqcp, seqc_t oldseqc*);

DESCRIPTION

The **seqc** allows zero or more readers and zero or one writer to concurrently access an object, providing a consistent snapshot of the object for readers. No mutual exclusion between readers and writers is required, but readers may be starved indefinitely by writers.

The functions **seqc_write_begin**() and **seqc_write_end**() are used to create a transaction for writer, and notify the readers that the object will be modified.

The **seqc_read**() function returns the current sequence number. If a writer has started a transaction, this function will spin until the transaction has ended.

The **seqc_consistent**() function compares the sequence number with a previously fetched value. The *oldseqc* variable should contain a sequence number from the beginning of read transaction.

The reader at the end of a transaction checks if the sequence number has changed. If the sequence number didn't change the object wasn't modified, and fetched variables are valid. If the sequence number changed the object was modified and the fetch should be repeated. In case when sequence number is odd the object change is in progress and the reader will wait until the write will the sequence number will become even.

EXAMPLES

The following example for a writer changees the *var1* and *var2* variables in the *obj* structure:

```
lock_exclusive(&obj->lock);
seqc_write_begin(&obj->seqc);
obj->var1 = 1;
obj->var2 = 2;
seqc_write_end(&obj->seqc);
unlock_exclusive(&obj->lock);
```

The following example for a reader reads the *var1* and *var2* variables from the *obj* structure. In the case where the sequence number was changed it restarts the whole process.

```
int var1, var2;
seqc_t seqc;

for (;;) {
        seqc = seqc_read(&obj->seqc);
        var1 = obj->var1;
        var2 = obj->var2;
        if (seqc_consistent(&obj->seqc, seqc))
                break;
}
```

AUTHORS

The **seqc** functions was implemented by Mateusz Guzik <*mjg@FreeBSD.org*>. This manual page was written by
Mariusz Zaborski <*oshogbo@FreeBSD.org*>.

CAVEATS

There is no guarantee of progress for readers. In case when there are a lot of writers the reader can be starved. This concern may be solved by returning error after a few attempts.

Theoretically if reading takes a very long time, and when there are many writers the counter may overflow and wrap around to the same value. In that case the reader will not notice that the object was changed. Given that this needs 4 billion transactional writes across a single contended reader, it is unlikely to ever happen. This could be avoided by extending the interface to allow 64-bit counters.