```
**********************************************************
    6960 Thu Dec 19 21:34:34 2013
new/usr/src/man/man3c/rwlock.3c
4327 rwlock(3c): Formatting issues and typos
**********************************************************
'\" te
.\"  Copyright (c) 1998 Sun Microsystems, Inc.  All Rights Reserved
.\" The contents of this file are subject to the terms of the Common Development
.\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
.\" When distributing Covered Code, include this CDDL HEADER in each file and in
.TH RWLOCK 3C "Dec 19, 2013"
.TH RWLOCK 3C "May 14, 1998"
.SH NAME
rwlock, rwlock_init, rwlock_destroy, rw_rdlock, rw_wrlock, rw_tryrdlock,
rw_trywrlock, rw_unlock \- multiple readers, single writer locks
.SH SYNOPSIS
.LP
.nf
cc -mt [ \fIflag\fR... ] \fIfile\fR...[ \fIlibrary\fR... ]

#include <synch.h>

\fBint\fR \fBrwlock_init\fR(\fBrwlock_t *\fR\fIrwlp\fR, \fBint\fR \fItype\fR, \f
\fBint\fR \fBrwlock_init\fR(\fBrwlock_t *\fR\fIrwlp\fR, \fBint\fR \fItype\fR, \f
.fi

.LP
.nf
\fBint\fR \fBrwlock_destroy\fR(\fBrwlock_t *\fR\fIrwlp\fR);
.fi

.LP
.nf
\fBint\fR \fBrw_rdlock\fR(\fBrwlock_t *\fR\fIrwlp\fR);
.fi

.LP
.nf
\fBint\fR \fBrw_wrlock\fR(\fBrwlock_t *\fR\fIrwlp\fR);
.fi

.LP
.nf
\fBint\fR \fBrw_unlock\fR(\fBrwlock_t *\fR\fIrwlp\fR);
.fi

.LP
.nf
\fBint\fR \fBrw_tryrdlock\fR(\fBrwlock_t *\fR\fIrwlp\fR);
.fi

.LP
.nf
\fBint\fR \fBrw_trywrlock\fR(\fBrwlock_t *\fR\fIrwlp\fR);
.fi

.SH DESCRIPTION
.sp
.LP
Many threads can have simultaneous read-only access to data, while only one
thread can have write access at any given time. Multiple read access with
single write access is controlled by locks, which are generally used to protect
data that is frequently searched.
.sp
.LP
Readers/writer locks can synchronize threads in this process and other
```



```
processes if they are allocated in writable memory  and shared among
cooperating processes (see \fBmmap\fR(2)), and are initialized for this
purpose.
.sp
.LP
Additionally, readers/writer locks must be initialized prior to use.
The readers/writer lock pointed to by \fIrwlp\fR is
\fBrwlock_init()\fR The readers/writer lock pointed to by \fIrwlp\fR is
initialized by  \fBrwlock_init()\fR. A readers/writer lock is capable of having
several types of behavior, which is specified by \fItype\fR. \fIarg\fR is
several types of behavior, which is specified by \fBtype\fR. \fIarg\fR is
currently not used, although a future type may define  new behavior parameters
by way of  \fIarg\fR.
.sp
.LP
The \fItype\fR argument can be one of the following:
.sp
.ne 2
.na
\fB\fBUSYNC_PROCESS\fR \fR
.ad
.RS 18n
The readers/writer lock can synchronize threads in this process and other
processes. The readers/writer lock should be initialized by only one process.
\fIarg\fR is ignored. A readers/writer lock initialized with this type, must be
allocated in memory shared between processses, i.e. either in Sys V shared
memory (see \fBshmop\fR(2)) or in memory mapped to a file (see \fBmmap\fR(2)).
It is illegal to initialize the object this way and to not allocate it in such
shared memory.
.RE

.sp
.ne 2
.na
\fB\fBUSYNC_THREAD\fR \fR
.ad
.RS 18n
The readers/writer lock can synchronize  threads in this process, only.
\fIarg\fR is ignored.
.RE

.sp
.LP
Additionally, readers/writer locks can be initialized by allocation  in zeroed
memory. A \fItype\fR of \fBUSYNC_THREAD\fR is assumed in this case. Multiple
threads must not simultaneously initialize the same  readers/writer lock. And a
readers/writer lock must not be re-initialized while in use by other threads.
.sp
.LP
The following are default readers/writer lock initialization (intra-process):
.sp
.in +2
.nf
rwlock_t rwlp;
rwlock_init(&rwlp, NULL, NULL);

.fi
.in -2

.sp
.LP
or
.sp
.in +2
.nf
```

```
rwlock_init(&rwlp, USYNC_THREAD, NULL);
.fi
.in -2

.sp
.LP
or
.sp
.in +2
.nf
rwlock_t  rwlp  =  DEFAULTRWLOCK;
.fi
.in -2

.sp
.LP
The following is a customized readers/writer lock  initialization
(inter-process):
.sp
.in +2
.nf
rwlock_init(&rwlp, USYNC_PROCESS, NULL);
.fi
.in -2

.sp
.LP
Any state associated with the readers/writer lock pointed to by \fIrwlp\fR are
destroyed by  \fBrwlock_destroy()\fR and the readers/writer lock storage space
is not released.
.sp
.LP
\fBrw_rdlock()\fR gets a read lock on the readers/writer lock pointed to by
\fIrwlp\fR. If the readers/writer lock is currently locked for writing, the
calling thread blocks until the write lock is freed. Multiple threads may
simultaneously hold a read lock on a  readers/writer lock.
.sp
.LP
\fBrw_tryrdlock()\fR tries to get a read lock on the readers/writer lock pointed
\fIrw_tryrdlock()\fR trys to get a read lock on the readers/writer lock pointed
to by \fIrwlp\fR. If the readers/writer lock is locked for writing, it returns
an error; otherwise, the read lock is acquired.
.sp
.LP
\fBrw_wrlock()\fR gets a write lock on the readers/writer lock pointed to by
\fIrwlp\fR. If the readers/writer lock is currently locked for reading or
writing, the calling thread blocks until all the read and write locks are
freed. At any given time, only one thread may have a write lock on a
readers/writer lock.
.sp
.LP
\fBrw_trywrlock()\fR tries to get a write lock on the readers/writer lock
\fIrw_trywrlock()\fR trys to get a write lock on the readers/writer lock
pointed to by \fIrwlp\fR. If the readers/writer lock is currently locked for
reading or writing, it returns an error.
.sp
.LP
\fBrw_unlock()\fR unlocks a readers/writer lock pointed to by  \fIrwlp\fR, if
the readers/writer lock is locked and the calling thread holds the lock for
either reading or writing. One of the other threads that is waiting for the
readers/writer  lock to be freed will be unblocked, provided there are other
readers/writer  lock to be freed will be unblocked, provided there is other
waiting threads. If the calling thread does not hold the lock for either
reading or writing, no error status is returned, and the program’s behavior is
unknown.
.SH RETURN VALUES
```



```
.sp
.LP
If successful, these functions return \fB0\fR. Otherwise, a non-zero value is
returned to indicate the error.
.SH ERRORS
.sp
.LP
The \fBrwlock_init()\fR function will fail if:
.sp
.ne 2
.na
\fB\fBEINVAL\fR \fR
.ad
.RS 11n
\fItype\fR is invalid.
\fBtype\fR is invalid.
.RE

.sp
.LP
The  \fBrw_tryrdlock()\fR or \fBrw_trywrlock()\fR functions will fail if:
.sp
.ne 2
.na
\fB\fBEBUSY\fR \fR
.ad
.RS 10n
The readers/writer lock pointed to by \fIrwlp\fR was already locked.
The reader or writer lock pointed to by \fIrwlp\fR was already locked.
.RE

.sp
.LP
These functions may fail if:
.sp
.ne 2
.na
\fB\fBEFAULT\fR \fR
.ad
.RS 11n
\fIrwlp\fR or \fIarg\fR points to an illegal address.
.RE

.SH ATTRIBUTES
.sp
.LP
See \fBattributes\fR(5) for descriptions of the following attributes:
.sp

.sp
.TS
box;
c | c
l | l .
ATTRIBUTE TYPE  ATTRIBUTE VALUE
_
MT-Level        MT-Safe
.TE

.SH SEE ALSO
.sp
.LP
\fBmmap\fR(2), \fBattributes\fR(5)
.SH NOTES
.sp
.LP
```

```
250 These interfaces also available by way of:
251 .sp
252 .LP
253 \fB#include\fR \fB<thread.h>\fR
254 .sp
255 .LP
256 If multiple threads are waiting for a readers/writer lock, the acquisition
257 order is random by default. However, some implementations may bias acquisition
258 order to avoid depriving writers. The current implementation favors writers
259 over readers.
```