```
**********************************************************
   10118 Mon May 26 16:37:12 2014
new/usr/src/man/man3c/lockf.3c
4841 - lockf(3c): Minor formatting issues in man page
**********************************************************
'\" te
.\"  Copyright 1989 AT&T  Copyright (c) 2002, Sun Microsystems, Inc.  All Rights
.\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for permission
.\" http://www.opengroup.org/bookstore/.
.\" The Institute of Electrical and Electronics Engineers and The Open Group, ha
.\"  This notice shall appear on any product containing this material.
.\" 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 LOCKF 3C "Apr 10, 2002"
.SH NAME
lockf \- record locking on files
.SH SYNOPSIS
.LP
.nf
#include <unistd.h>

\fBint\fR \fBlockf\fR(\fBint\fR \fIfildes\fR, \fBint\fR \fIfunction\fR, \fBoff_t
.fi

.SH DESCRIPTION
.sp
.LP
The \fBlockf()\fR function allows sections of a file to be locked; advisory or
mandatory write locks depending  on the mode bits of the file (see
\fBchmod\fR(2)). Calls to \fBlockf()\fR from other threads that attempt to lock
the locked file section will either return an error value or be put to sleep
until the resource becomes unlocked. All the locks for a process are removed
when the process terminates. See \fBfcntl\fR(2) for more information about
record locking.
.sp
.LP
The \fIfildes\fR argument is an open file descriptor. The file descriptor must
have \fBO_WRONLY\fR or \fBO_RDWR\fR permission in order to establish locks with
this function call.
.sp
.LP
The \fIfunction\fR argument is a control value that specifies the action to be
taken. The permissible values for \fIfunction\fR are defined in
<\fBunistd.h\fR> as follows:
.sp
.in +2
.nf
#define   F_ULOCK   0   /* unlock previously locked section */
#define   F_LOCK    1   /* lock section for exclusive use */
#define   F_TLOCK   2   /* test & lock section for exclusive use */
#define   F_TEST    3   /* test section for other locks */
.fi
.in -2

.sp
.LP
All other values of \fIfunction\fR are reserved for future extensions and will
result in an error if not implemented.
.sp
.LP
\fBF_TEST\fR is used to detect if a lock by another process is present on the
specified section. \fBF_LOCK\fR and \fBF_TLOCK\fR both lock a section of a file
```



```
if the section is available. \fBF_ULOCK\fR removes locks from a section of the
file.
.sp
.LP
The \fIsize\fR argument is the number of contiguous bytes to be locked or
unlocked. The resource to be locked or unlocked starts at the current offset in
the file and extends forward for a positive \fIsize\fR and backward for a negati
\fIsize\fR (the preceding bytes up to but not including the current offset). If
\fIsize\fR is zero, the section from the current offset through the largest
file offset is locked (that is, from the current offset through the present or
any future end-of-file). An area need not be allocated to the file in order to
be locked as such locks may exist past the end-of-file.
.sp
.LP
The sections locked with \fBF_LOCK\fR or \fBF_TLOCK\fR may, in whole or in
part, contain or be contained by a previously locked section for the same
process.  Locked sections will be unlocked starting at the point of the offset
through \fIsize\fR bytes or to the end of file if \fIsize\fR is (\fBoff_t\fR)
0. When this situation occurs, or if this situation occurs in adjacent
sections, the sections are combined into a single section. If the request
requires that a new element be added to the table of active locks and this
table is already full, an error is returned, and the new section is not locked.
.sp
.LP
\fBF_LOCK\fR and \fBF_TLOCK\fR requests differ only by the action taken if the
resource is not available. \fBF_LOCK\fR blocks the calling thread until the
resource is available. \fBF_TLOCK\fR causes the function to return \(mi1 and
set \fBerrno\fR to \fBEAGAIN\fR if the section is already locked by another
process.
.sp
.LP
File locks are released on first close by the locking process of any file
descriptor for the file.
.sp
.LP
\fBF_ULOCK\fR requests may, in whole or in part, release one or more locked
sections controlled by the process. When sections are not fully released, the
remaining sections are still locked by the process. Releasing the center
section of a locked section requires an additional element in the table of
active locks. If this table is full, an \fBerrno\fR is set to \fBEDEADLK\fR and
the requested section is not released.
.sp
.LP
An \fBF_ULOCK\fR request in which \fIsize\fR is non-zero and the offset of the
last byte of the requested section is the maximum value for an object of type
\fBoff_t\fR, when the process has an existing lock in which \fIsize\fR is 0 and
which includes the last byte of the requested section, will be treated as a
request to unlock from the start of the requested section with a \fIsize\fR equa
0. Otherwise, an \fBF_ULOCK\fR request will attempt to unlock only the
requested section.
.sp
.LP
A potential for deadlock occurs if the threads of a process controlling a
locked resource is put to sleep by requesting another process's locked
resource. Thus calls to \fBlockf()\fR or \fBfcntl\fR(2) scan for a deadlock
prior to sleeping on a locked resource. An error return is made if sleeping on
the locked resource would cause a deadlock.
.sp
```

```
117 .LP
118 Sleeping on a resource is interrupted with any signal. The \fBalarm\fR(2)
119 function may be used to provide a timeout facility in applications that require
120 this facility.
121 .SH RETURN VALUES
122 .sp
123 .LP
124 Upon successful completion, \fB0\fR is returned.  Otherwise, \fB\(mi1\fR is
125 returned and \fBerrno\fR is set to indicate the error.
126 .SH ERRORS
127 .sp
128 .LP
129 The \fBlockf()\fR function will fail if:
130 .sp
131 .ne 2
132 .na
133 \fB\fBEBADF\fR\fR
134 .ad
135 .RS 20n
136 The \fIfildes\fR argument is not a valid open file descriptor; or
137 \fBfunction\fR is \fBF_LOCK\fR or \fBF_TLOCK\fR and \fIfildes\fR is not a valid
137 \fIfunction\fR is \fBF_LOCK\fR or \fBF_TLOCK\fR and \fIfildes\fR is not a valid
138 file descriptor open for writing.
139 .RE

141 .sp
142 .ne 2
143 .na
144 \fB\fBEACCES\fR or \fBEAGAIN\fR\fR
145 .ad
146 .RS 20n
147 The \fIfunction\fR argument is \fBF_TLOCK\fR or \fBF_TEST\fR and the section is
147 The \fBfunction\fR argument is \fBF_TLOCK\fR or \fBF_TEST\fR and the section is
148 already locked by another process.
149 .RE

151 .sp
152 .ne 2
153 .na
154 \fB\fBEDEADLK\fR\fR
155 .ad
156 .RS 20n
157 The \fIfunction\fR argument is \fBF_LOCK\fR and a deadlock is detected.
157 The \fBfunction\fR argument is \fBF_LOCK\fR and a deadlock is detected.
158 .RE

160 .sp
161 .ne 2
162 .na
163 \fB\fBEINTR\fR\fR
164 .ad
165 .RS 20n
166 A signal was caught during execution of the function.
167 .RE

169 .sp
170 .ne 2
171 .na
172 \fB\fBECOMM\fR\fR
173 .ad
174 .RS 20n
175 The \fIfildes\fR argument is on a remote machine and the link to that machine
176 is no longer active.
177 .RE

179 .sp
```



```
180 .ne 2
181 .na
182 \fB\fBEINVAL\fR\fR
183 .ad
184 .RS 20n
185 The \fIfunction\fR argument is not one of \fBF_LOCK\fR, \fBF_TLOCK\fR,
186 \fBF_TEST\fR, or \fBF_ULOCK\fR; or \fIsize\fR plus the current file offset is
185 The \fBfunction\fR argument is not one of \fBF_LOCK\fR, \fBF_TLOCK\fR,
186 \fBF_TEST\fR, or \fBF_ULOCK\fR; or \fBsize\fR plus the current file offset is
187 less than 0.
188 .RE

190 .sp
191 .ne 2
192 .na
193 \fB\fBEOVERFLOW\fR\fR
194 .ad
195 .RS 20n
196 The offset of the first, or if \fIsize\fR is not 0 then the last, byte in the
196 The offset of the first, or if \fBsize\fR is not 0 then the last, byte in the
197 requested section cannot be represented correctly in an object of type
198 \fBoff_t\fR.
199 .RE

201 .sp
202 .LP
203 The \fBlockf()\fR function may fail if:
204 .sp
205 .ne 2
206 .na
207 \fB\fBEAGAIN\fR\fR
208 .ad
209 .RS 24n
210 The \fIfunction\fR argument is \fBF_LOCK\fR or \fBF_TLOCK\fR and the file is
210 The \fBfunction\fR argument is \fBF_LOCK\fR or \fBF_TLOCK\fR and the file is
211 mapped with \fBmmap\fR(2).
212 .RE

214 .sp
215 .ne 2
216 .na
217 \fB\fBEDEADLK\fR or \fBENOLCK\fR\fR
218 .ad
219 .RS 24n
220 The \fIfunction\fR argument is \fBF_LOCK\fR, \fBF_TLOCK\fR, or \fBF_ULOCK\fR
220 The \fBfunction\fR argument is \fBF_LOCK\fR, \fBF_TLOCK\fR, or \fBF_ULOCK\fR
221 and the request would cause the number of locks to exceed a system-imposed
222 limit.
223 .RE

225 .sp
226 .ne 2
227 .na
228 \fB\fBEOPNOTSUPP\fR or \fBEINVAL\fR\fR
229 .ad
230 .RS 24n
231 The locking of files of the type indicated by the \fIfildes\fR argument is not
232 supported.
233 .RE

235 .SH USAGE
236 .sp
237 .LP
238 Record-locking should not be used in combination with the \fBfopen\fR(3C),
239 \fBfread\fR(3C), \fBfwrite\fR(3C) and other \fBstdio\fR functions.  Instead,
240 the more primitive, non-buffered functions (such as \fBopen\fR(2)) should be
```

```
241 used.  Unexpected results may occur in processes that do buffering in the user
242 address space.  The process may later read/write data which is/was locked.  The
243 \fBstdio\fR functions are the most common source of unexpected buffering.
244 .sp
245 .LP
246 The \fBalarm\fR(2) function may be used to provide a timeout facility in
247 applications requiring it.
248 .sp
249 .LP
250 The \fBlockf()\fR function has a transitional interface for 64-bit file
251 offsets.  See \fBlf64\fR(5).
252 .SH ATTRIBUTES
253 .sp
254 .LP
255 See \fBattributes\fR(5) for descriptions of the following attributes:
256 .sp

258 .sp
259 .TS
260 box;
261 c | c
262 l | l .
263 ATTRIBUTE TYPE  ATTRIBUTE VALUE
264 _
265 Interface Stability     Standard
266 _
267 MT-Level        MT-Safe
268 .TE

270 .SH SEE ALSO
271 .sp
272 .LP
273 \fBIntro\fR(2), \fBalarm\fR(2), \fBchmod\fR(2), \fBclose\fR(2), \fBcreat\fR(2),
274 \fBfcntl\fR(2), \fBmmap\fR(2), \fBopen\fR(2), \fBread\fR(2), \fBwrite\fR(2),
275 \fBattributes\fR(5), \fBlf64\fR(5), \fBstandards\fR(5)
```