```
*********************************************************
   10118 Mon May 26 16:40:22 2014
new/usr/src/man/man3c/lockf.3c
4841 - lockf(3c): Minor formatting issues in man page
*********************************************************
   1 '\" te
   2 .\"  Copyright 1989 AT&T  Copyright (c) 2002, Sun Microsystems, Inc.  All Rights
   3 .\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for permission
   4 .\" http://www.opengroup.org/bookstore/.
   5 .\" The Institute of Electrical and Electronics Engineers and The Open Group, ha
   6 .\"  This notice shall appear on any product containing this material.
   7 .\" The contents of this file are subject to the terms of the Common Development
   8 .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
   9 .\" When distributing Covered Code, include this CDDL HEADER in each file and in
  10 .TH LOCKF 3C "Apr 10, 2002"
  11 .SH NAME
  12 lockf \- record locking on files
  13 .SH SYNOPSIS
  14 .LP
  15 .nf
  16 #include <unistd.h>

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

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

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

```
  59 if the section is available. \fBF_ULOCK\fR removes locks from a section of the
  60 file.
  61 .sp
  62 .LP
  63 The \fIsize\fR argument is the number of contiguous bytes to be locked or
  63 The \fBsize\fR argument is the number of contiguous bytes to be locked or
  64 unlocked. The resource to be locked or unlocked starts at the current offset in
  65 the file and extends forward for a positive \fIsize\fR and backward for a negati
  66 \fIsize\fR (the preceding bytes up to but not including the current offset). If
  67 \fIsize\fR is zero, the section from the current offset through the largest
  65 the file and extends forward for a positive size and backward for a negative
  66 size (the preceding bytes up to but not including the current offset). If
  67 \fBsize\fR is zero, the section from the current offset through the largest
  68 file offset is locked (that is, from the current offset through the present or
  69 any future end-of-file). An area need not be allocated to the file in order to
  70 be locked as such locks may exist past the end-of-file.
  71 .sp
  72 .LP
  73 The sections locked with \fBF_LOCK\fR or \fBF_TLOCK\fR may, in whole or in
  74 part, contain or be contained by a previously locked section for the same
  75 process.  Locked sections will be unlocked starting at the point of the offset
  76 through \fIsize\fR bytes or to the end of file if \fIsize\fR is (\fBoff_t\fR)
  76 through \fBsize\fR bytes or to the end of file if \fBsize\fR is (\fBoff_t\fR)
  77 0. When this situation occurs, or if this situation occurs in adjacent
  78 sections, the sections are combined into a single section. If the request
  79 requires that a new element be added to the table of active locks and this
  80 table is already full, an error is returned, and the new section is not locked.
  81 .sp
  82 .LP
  83 \fBF_LOCK\fR and \fBF_TLOCK\fR requests differ only by the action taken if the
  84 resource is not available. \fBF_LOCK\fR blocks the calling thread until the
  85 resource is available. \fBF_TLOCK\fR causes the function to return \(mil and
  86 set \fBerrno\fR to \fBEAGAIN\fR if the section is already locked by another
  87 process.
  88 .sp
  89 .LP
  90 File locks are released on first close by the locking process of any file
  91 descriptor for the file.
  92 .sp
  93 .LP
  94 \fBF_ULOCK\fR requests may, in whole or in part, release one or more locked
  95 sections controlled by the process. When sections are not fully released, the
  96 remaining sections are still locked by the process. Releasing the center
  97 section of a locked section requires an additional element in the table of
  98 active locks. If this table is full, an \fBerrno\fR is set to \fBEDEADLK\fR and
  99 the requested section is not released.
 100 .sp
 101 .LP
 102 An \fBF_ULOCK\fR request in which \fIsize\fR is non-zero and the offset of the
 102 An \fBF_ULOCK\fR request in which \fBsize\fR is non-zero and the offset of the
 103 last byte of the requested section is the maximum value for an object of type
 104 \fBoff_t\fR, when the process has an existing lock in which \fIsize\fR is 0 and
 104 \fBoff_t\fR, when the process has an existing lock in which \fBsize\fR is 0 and
 105 which includes the last byte of the requested section, will be treated as a
 106 request to unlock from the start of the requested section with a \fIsize\fR equa
 106 request to unlock from the start of the requested section with a size equal to
 107 0. Otherwise, an \fBF_ULOCK\fR request will attempt to unlock only the
 108 requested section.
 109 .sp
 110 .LP
 111 A potential for deadlock occurs if the threads of a process controlling a
 112 locked resource is put to sleep by requesting another process's locked
 113 resource. Thus calls to \fBlockf()\fR or \fBfcntl\fR(2) scan for a deadlock
 114 prior to sleeping on a locked resource. An error return is made if sleeping on
 115 the locked resource would cause a deadlock.
 116 .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 **\fIfunction\fR is \fBF_LOCK\fR or \fBF_TLOCK\fR and \fIfildes\fR is not a valid**
 137 *\fBfunction\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)
```