Source to sys/fcntl.h

Search Enter a symbol's name here to quickly find it.

```
/*-
 * Copyright (c) 1983, 1990 The Regents of the University of California.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 *      @(#)fcntl.h     5.14 (Berkeley) 7/1/91
 */

#ifndef _FCNTL_H_
#define _FCNTL_H_

/*
 * This file includes the definitions for open and fcntl
 * described by POSIX for <fcntl.h>; it also includes
 * related kernel definitions.
 */

#ifndef KERNEL
#include <sys/types.h>
#endif

/*
 * File status flags: these are used by open(2), fcntl(2).
 * They are also used (indirectly) in the kernel file structure f_flags,
 * which is a superset of the open/fcntl flags.  Open flags and f_flags
 * are inter-convertible using OFLAGS(fflags) and FFLAGS(oflags).
 * Open/fcntl flags begin with O_; kernel-internal flags begin with F.
 */
/* open-only flags */
#define O_RDONLY        0x0000          /* open for reading only */
#define O_WRONLY        0x0001          /* open for writing only */
#define O_RDWR          0x0002          /* open for reading and writing */
#define O_ACCMODE       0x0003          /* mask for above modes */
```

```
#ifdef KERNEL
/*
 * Kernel encoding of open mode; separate read and write bits
 * that are independently testable: 1 greater than the above.
 */
#define FREAD           0x0001
#define FWRITE          0x0002
#endif
#define O_NONBLOCK      0x0004          /* no delay */
#define O_APPEND        0x0008          /* set append mode */
#ifndef _POSIX_SOURCE
#define O_SHLOCK        0x0010          /* open with shared file lock */
#define O_EXLOCK        0x0020          /* open with exclusive file lock */
#define O_ASYNC         0x0040          /* signal pgrp when data ready */
#define O_FSYNC         0x0080          /* synchronous writes */
#endif
#define O_CREAT         0x0200          /* create if nonexistant */
#define O_TRUNC         0x0400          /* truncate to zero length */
#define O_EXCL          0x0800          /* error if already exists */
#ifdef KERNEL
#define FMARK           0x1000          /* mark during gc() */
#define FDEFER          0x2000          /* defer for next gc pass */
#define FHASLOCK        0x4000          /* descriptor holds advisory lock */
#endif

/* defined by POSIX 1003.1; BSD default, so no bit required */
#define O_NOCTTY        0               /* don't assign controlling terminal */

#ifdef KERNEL
/* convert from open() flags to/from fflags; convert O_RD/WR to FREAD/FWRITE */
#define FFLAGS(oflags)  ((oflags) + 1)
#define OFLAGS(fflags)  ((fflags) - 1)

/* bits to save after open */
#define FMASK           (FREAD|FWRITE|FAPPEND|FASYNC|FFSYNC|FNONBLOCK)
/* bits settable by fcntl(F_SETFL, ...) */
#define FCNTLFLAGS      (FAPPEND|FASYNC|FFSYNC|FNONBLOCK)
#endif

/*
 * The O_* flags used to have only F* names, which were used in the kernel
 * and by fcntl.  We retain the F* names for the kernel f_flags field
 * and for backward compatibility for fcntl.
 */
#ifndef _POSIX_SOURCE
#define FAPPEND         O_APPEND        /* kernel/compat */
#define FASYNC          O_ASYNC         /* kernel/compat */
#define FFSYNC          O_FSYNC         /* kernel */
#define FNONBLOCK       O_NONBLOCK      /* kernel */
#define FNDELAY         O_NONBLOCK      /* compat */
#define O_NDELAY        O_NONBLOCK      /* compat */
#endif

/*
 * Constants used for fcntl(2)
 */

/* command values */
#define F_DUPFD         0               /* duplicate file descriptor */
#define F_GETFD         1               /* get file descriptor flags */
#define F_SETFD         2               /* set file descriptor flags */
#define F_GETFL         3               /* get file status flags */
#define F_SETFL         4               /* set file status flags */
#ifndef _POSIX_SOURCE
#define F_GETOWN        5               /* get SIGIO/SIGURG proc/pgrp */
#define F_SETOWN        6               /* set SIGIO/SIGURG proc/pgrp */
#endif
#define F_GETLK         7               /* get record locking information */
#define F_SETLK         8               /* set record locking information */
```

```
#define F_SETLKW        9               /* F_SETLK; wait if blocked */

/* file descriptor flags (F_GETFD, F_SETFD) */
#define FD_CLOEXEC      1               /* close-on-exec flag */

/* record locking flags (F_GETLK, F_SETLK, F_SETLKW) */
#define F_RDLCK         1               /* shared or read lock */
#define F_UNLCK         2               /* unlock */
#define F_WRLCK         3               /* exclusive or write lock */
#ifdef KERNEL
#define F_WAIT          0x010           /* Wait until lock is granted */
#define F_FLOCK         0x020           /* Use flock(2) semantics for lock */
#define F_POSIX         0x040           /* Use POSIX semantics for lock */
#endif

/*
 * Advisory file segment locking data type -
 * information passed to system by user
 */
struct flock {
        short   l_type;         /* lock type: read/write, etc. */
        short   l_whence;       /* type of l_start */
        off_t   l_start;        /* starting offset */
        off_t   l_len;          /* len = 0 means until end of file */
        pid_t   l_pid;          /* lock owner */
};


#ifndef _POSIX_SOURCE
/* lock operations for flock(2) */
#define LOCK_SH         0x01            /* shared file lock */
#define LOCK_EX         0x02            /* exclusive file lock */
#define LOCK_NB         0x04            /* don't block when locking */
#define LOCK_UN         0x08            /* unlock file */
#endif


#ifndef KERNEL
#include <sys/cdefs.h>

__BEGIN_DECLS
int     open __P((const char *, int, ...));
int     creat __P((const char *, mode_t));
int     fcntl __P((int, int, ...));
#ifndef _POSIX_SOURCE
int     flock __P((int, int));
#endif /* !_POSIX_SOURCE */
__END_DECLS
#endif

#endif /* !_FCNTL_H_ */
```