程序 14-5　linux/include/fcntl.h

```
1  #ifndef _FCNTL_H
2  #define _FCNTL_H
3
4  #include <sys/types.h>    // 类型头文件。定义了基本的系统数据类型。
5
6  /* open/fcntl - NOCTTY, NDELAY isn't implemented yet */
   /* open/fcntl - NOCTTY 和 NDELAY 现在还没有实现 */
7  #define O_ACCMODE       00003                   // 文件访问模式屏蔽码。
   // 打开文件 open()和文件控制函数 fcntl()使用的文件访问模式。同时只能使用三者之一。
8  #define O_RDONLY          00                    // 以只读方式打开文件。
9  #define O_WRONLY          01                    // 以只写方式打开文件。
10 #define O_RDWR            02                    // 以读写方式打开文件。
   // 下面是文件创建和操作标志，用于 open()。可与上面访问模式用'位或'的方式一起使用。
11 #define O_CREAT         00100   /* not fcntl */ // 如果文件不存在就创建。fcntl 函数不用。
12 #define O_EXCL          00200   /* not fcntl */ // 独占使用文件标志。
13 #define O_NOCTTY        00400   /* not fcntl */ // 不分配控制终端。
14 #define O_TRUNC         01000   /* not fcntl */ // 若文件已存在且是写操作，则长度截为 0。
15 #define O_APPEND        02000                   // 以添加方式打开，文件指针置为文件尾。
16 #define O_NONBLOCK      04000   /* not fcntl */ // 非阻塞方式打开和操作文件。
17 #define O_NDELAY        O_NONBLOCK              // 非阻塞方式打开和操作文件。
18
19 /* Defines for fcntl-commands. Note that currently
20  * locking isn't supported, and other things aren't really
21  * tested.
22  */
   /* 下面定义了 fcntl 的命令。注意目前锁定命令还没有支持，而其他
    * 命令实际上还没有测试过。
    */
   // 文件句柄(描述符)操作函数 fcntl()的命令（cmd）。
23 #define F_DUPFD         0       /* dup */       // 拷贝文件句柄为最小数值的句柄。
24 #define F_GETFD         1       /* get f_flags */  // 取句柄标志。仅 1 个标志 FD_CLOEXEC。
25 #define F_SETFD         2       /* set f_flags */  // 设置文件句柄标志。
26 #define F_GETFL         3       /* more flags (cloexec) */  // 取文件状态标志和访问模式。
27 #define F_SETFL         4                                   // 设置文件状态标志和访问模式。
   // 下面是文件锁定命令。fcntl()的第三个参数 lock 是指向 flock 结构的指针。
28 #define F_GETLK         5       /* not implemented */  // 返回阻止锁定的 flock 结构。
29 #define F_SETLK         6       // 设置(F_RDLCK 或 F_WRLCK)或清除(F_UNLCK)锁定。
30 #define F_SETLKW        7       // 等待设置或清除锁定。
31
32 /* for F_[GET|SET]FL */
   /* 用于 F_GETFL 或 F_SETFL */
   // 在执行 exec()簇函数时需要关闭的文件句柄。(执行时关闭 - Close On EXECution)
33 #define FD_CLOEXEC      1       /* actually anything with low bit set goes */
                                   /* 实际上只要低位为 1 即可 */
34
35 /* Ok, these are locking features, and aren't implemented at any
36  * level. POSIX wants them.
37  */
   /* OK，以下是锁定类型，任何函数中都还没有实现。POSIX 标准要求这些类型。
    */
38 #define F_RDLCK         0       // 共享或读文件锁定。
39 #define F_WRLCK         1       // 独占或写文件锁定。
```

```
40 #define F_UNLCK         2       // 文件解锁。
41
42 /* Once again - not implemented, but ... */
   /* 同样 - 也还没有实现，但是... */
   // 文件锁定操作数据结构。描述了受影响文件段的类型(l_type)、开始偏移(l_whence)、
   // 相对偏移(l_start)、锁定长度(l_len)和实施锁定的进程 id。
43 struct flock {
44         short l_type;           // 锁定类型（F_RDLCK，F_WRLCK，F_UNLCK）。
45         short l_whence;         // 开始偏移(SEEK_SET，SEEK_CUR 或 SEEK_END)。
46         off_t l_start;          // 阻塞锁定的开始处。相对偏移（字节数）。
47         off_t l_len;            // 阻塞锁定的大小；如果是 0 则为到文件末尾。
48         pid_t l_pid;            // 加锁的进程 id。
49 };
50
   // 以下是使用上述标志或命令的函数原型。
   // 创建新文件或重写一个已存在文件。
   // 参数 filename 是欲创建文件的文件名，mode 是创建文件的属性（见 include/sys/stat.h）。
51 extern int creat(const char * filename,mode_t mode);
   // 文件句柄操作，会影响文件的打开。
   // 参数 fildes 是文件句柄，cmd 是操作命令，见上面 23--30 行。该函数可有以下几种形式：
   // int fcntl(int fildes, int cmd);
   // int fcntl(int fildes, int cmd, long arg);
   // int fcntl(int fildes, int cmd, struct flock *lock);
52 extern int fcntl(int fildes,int cmd, ...);
   // 打开文件。在文件与文件句柄之间建立联系。
   // 参数 filename 是欲打开文件的文件名，flags 是上面 7-17 行上的标志的组合。
53 extern int open(const char * filename, int flags, ...);
54
55 #endif
56
```