```
 1 /*
 2  *  linux/fs/fcntl.c
 3  *
 4  *  (C) 1991  Linus Torvalds
 5  */
 6
 7 #include <string.h>       // 字符串头文件。主要定义了一些有关字符串操作的嵌入函数。
 8 #include <errno.h>        // 错误号头文件。包含系统中各种出错号。
 9 #include <linux/sched.h>  // 调度程序头文件，定义了任务结构 task_struct、任务 0 数据等。
10 #include <linux/kernel.h> // 内核头文件。含有一些内核常用函数的原形定义。
11 #include <asm/segment.h>  // 段操作头文件。定义了有关段寄存器操作的嵌入式汇编函数。
12
13 #include <fcntl.h>        // 文件控制头文件。定义文件及其描述符的操作控制常数符号。
14 #include <sys/stat.h>     // 文件状态头文件。含有文件状态结构 stat{}和常量。
15
16 extern int sys_close(int fd);     // 关闭文件系统调用。(fs/open.c, 192)
17
   //// 复制文件句柄（文件描述符）。
   // 参数 fd 是欲复制的文件句柄，arg 指定新文件句柄的最小数值。
   // 返回新文件句柄或出错码。
18 static int dupfd(unsigned int fd, unsigned int arg)
19 {
   // 首先检查函数参数的有效性。如果文件句柄值大于一个程序最多打开文件数 NR_OPEN，或者
   // 该句柄的文件结构不存在，则返回出错码并退出。如果指定的新句柄值 arg 大于最多打开文
   // 件数，也返回出错码并退出。注意，实际上文件句柄就是进程文件结构指针数组项索引号。
20         if (fd >= NR_OPEN || !current->filp[fd])
21                 return -EBADF;
22         if (arg >= NR_OPEN)
23                 return -EINVAL;
   // 然后在当前进程的文件结构指针数组中寻找索引号等于或大于 arg，但还没有使用的项。若
   // 找到的新句柄值 arg 大于最多打开文件数（即没有空闲项），则返回出错码并退出。
24         while (arg < NR_OPEN)
25                 if (current->filp[arg])
26                         arg++;
27                 else
28                         break;
29         if (arg >= NR_OPEN)
30                 return -EMFILE;
   // 否则针对找到的空闲项（句柄），在执行时关闭标志位图 close_on_exec 中复位该句柄位。
   // 即在运行 exec()类函数时，不会关闭用 dup()创建的句柄。并令该文件结构指针等于原句
   // 柄 fd 的指针，并且将文件引用计数增 1。最后返回新的文件句柄 arg。
31         current->close_on_exec &= ~(1<<arg);
32         (current->filp[arg] = current->filp[fd])->f_count++;
33         return arg;
34 }
35
   //// 复制文件句柄系统调用。
   // 复制指定文件句柄 oldfd，新文件句柄值等于 newfd。如果 newfd 已打开，则首先关闭之。
   // 参数：oldfd -- 原文件句柄；newfd - 新文件句柄。
   // 返回新文件句柄值。
36 int sys_dup2(unsigned int oldfd, unsigned int newfd)
37 {
```

```
38          sys_close(newfd);               // 若句柄 newfd 已经打开，则首先关闭之。
39          return dupfd(oldfd,newfd);      // 复制并返回新句柄。
40  }
41
    //// 复制文件句柄系统调用。
    // 复制指定文件句柄 oldfd，新句柄的值是当前最小的未用句柄值。
    // 参数：fildes -- 被复制的文件句柄。
    // 返回新文件句柄值。
42  int sys_dup(unsigned int fildes)
43  {
44          return dupfd(fildes,0);
45  }
46
    //// 文件控制系统调用函数。
    // 参数 fd 是文件句柄；cmd 是控制命令（参见 include/fcntl.h，23-30 行）；arg 则针对不
    // 同的命令有不同的含义。对于复制句柄命令 F_DUPFD，arg 是新文件句柄可取的最小值；对
    // 于设置文件操作和访问标志命令 F_SETFL，arg 是新的文件操作和访问模式。对于文件上锁
    // 命令 F_GETLK、F_SETLK 和 F_SETLKW，arg 是指向 flock 结构的指针。但本内核中没有实现
    // 文件上锁功能。
    // 返回：若出错，则所有操作都返回-1。若成功，那么 F_DUPFD 返回新文件句柄； F_GETFD
    // 返回文件句柄的当前执行时关闭标志 close_on_exec；F_GETFL 返回文件操作和访问标志。
47  int sys_fcntl(unsigned int fd, unsigned int cmd, unsigned long arg)
48  {
49          struct file * filp;
50
    // 首先检查给出的文件句柄的有效性。然后根据不同命令 cmd 进行分别处理。 如果文件句柄
    // 值大于一个进程最多打开文件数 NR_OPEN，或者该句柄的文件结构指针为空，则返回出错码
    // 并退出。
51          if (fd >= NR_OPEN || !(filp = current->filp[fd]))
52                  return -EBADF;
53          switch (cmd) {
54                  case F_DUPFD:       // 复制文件句柄。
55                          return dupfd(fd,arg);
56                  case F_GETFD:       // 取文件句柄的执行时关闭标志。
57                          return (current->close_on_exec>>fd)&1;
58                  case F_SETFD:  // 设置执行时关闭标志。arg 位 0 置位是设置，否则关闭。
59                          if (arg&1)
60                                  current->close_on_exec |= (1<<fd);
61                          else
62                                  current->close_on_exec &= ~(1<<fd);
63                          return 0;
64                  case F_GETFL:       // 取文件状态标志和访问模式。
65                          return filp->f_flags;
66                  case F_SETFL: // 设置文件状态和访问模式（根据 arg 设置添加、非阻塞标志）。
67                          filp->f_flags &= ~(O_APPEND | O_NONBLOCK);
68                          filp->f_flags |= arg & (O_APPEND | O_NONBLOCK);
69                          return 0;
70                  case F_GETLK:   case F_SETLK:   case F_SETLKW:    // 未实现。
71                          return -1;
72                  default:
73                          return -1;
74          }
75  }
```