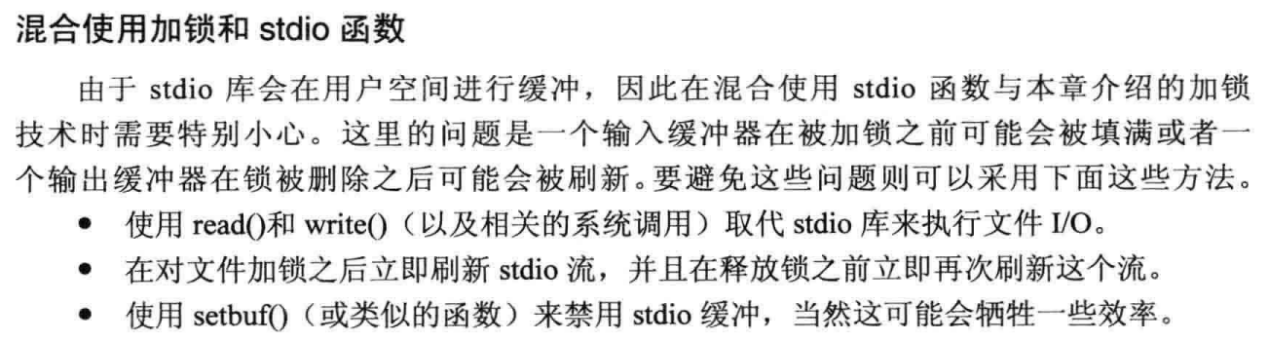
第五十五章

1. 文件加锁：
2. 概述：

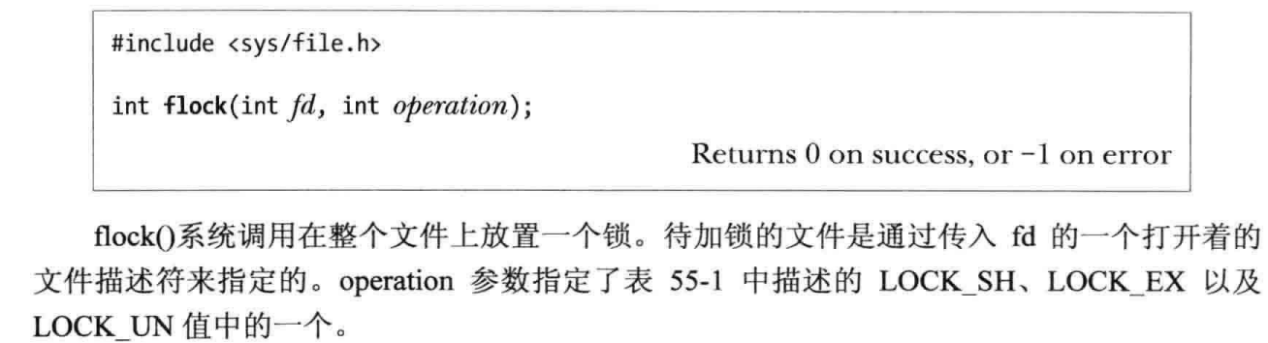
注意事项：



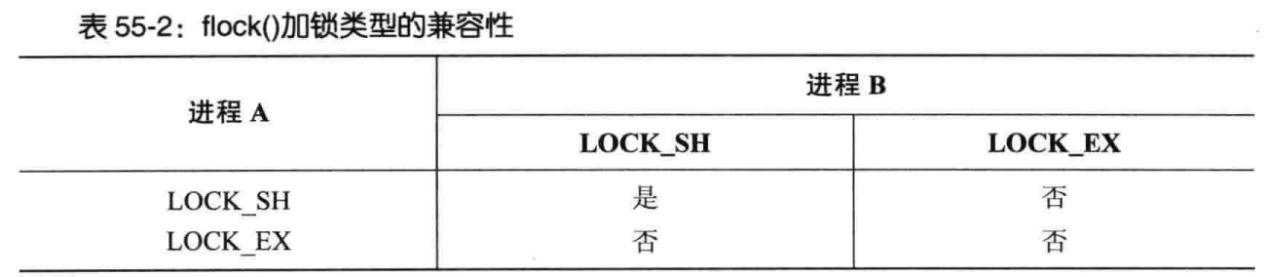
有两种类型的锁：

劝告锁和强制锁。

1. 使用flock给文件加锁：







注意，可以再次调用flock函数更换锁的类型，此方法不是原子的。

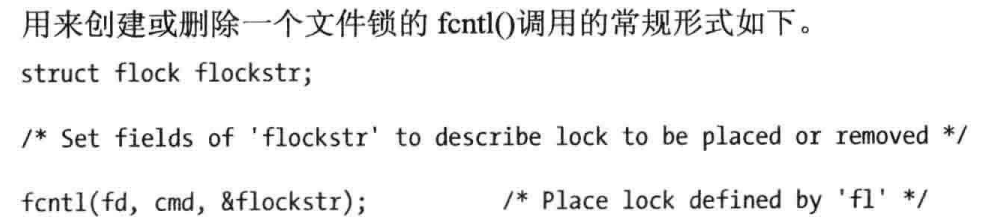
1. 锁继承与释放的语义：

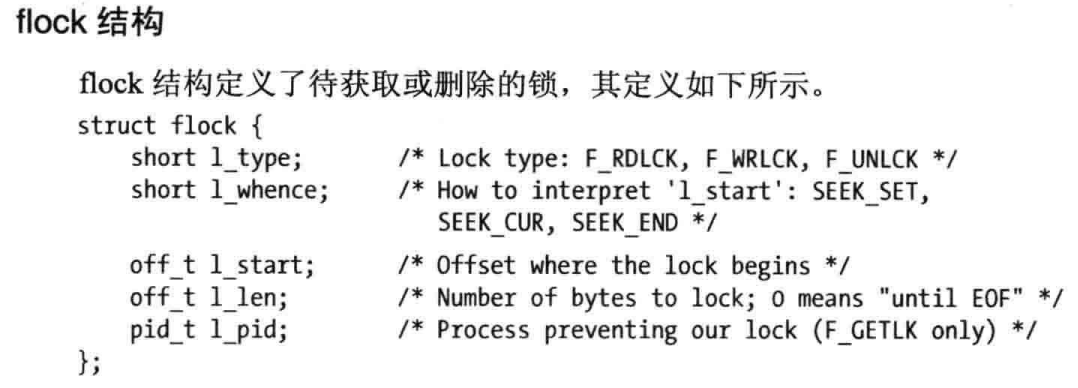
注意一点，通过flock获取的文件锁是与打开的文件描述关联的，而不是文件描述符和inode文件本身。所以继承和释放的语义所遵循的规律与上述描述有关。

1. Flock的限制：

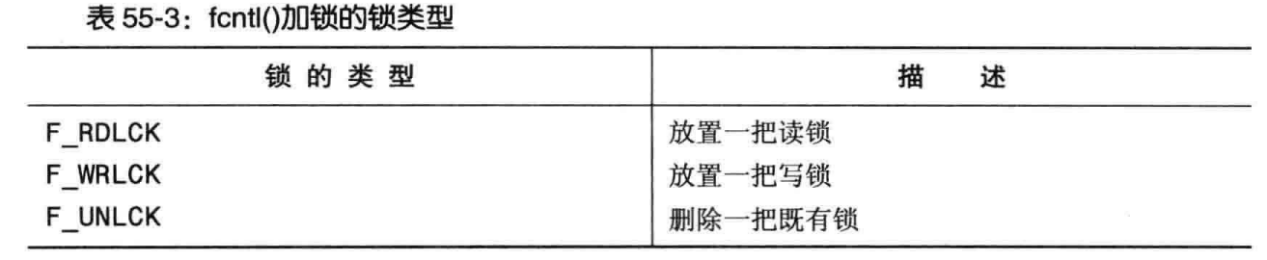


1. 使用fcntl给记录加锁：





l\_type字段锁类型：

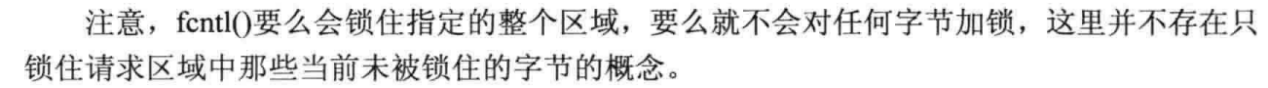


注意，放置一个文件的读锁需要以读取的方式打开文件。写锁类似。

其他的查看书本和手册。

Cmd参数：

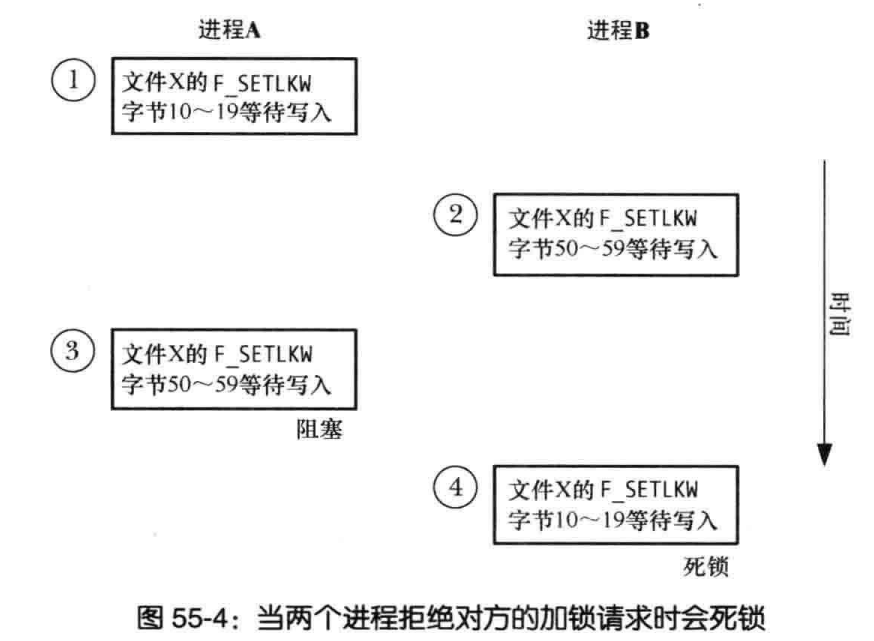
F\_SETLK，F\_SETLKW，F\_GETLK。



锁获取和释放的细节查看书本，比较有用。

1. 死锁：

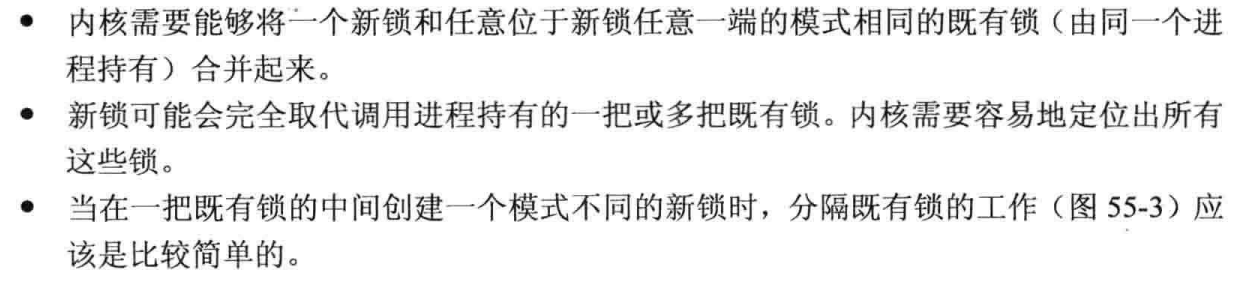
使用F\_SETLKW的时候，会出现死锁的情况，举例如下：



1. 锁的限制和性能：

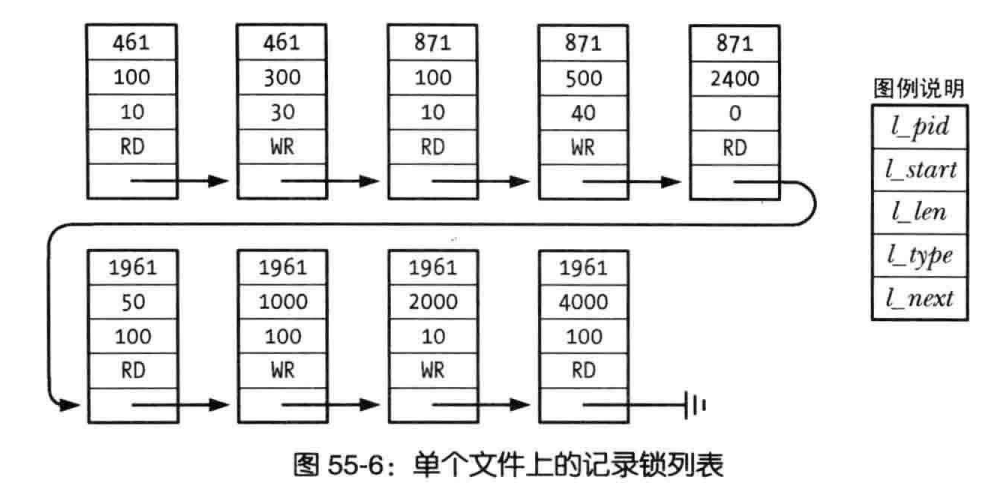
Linux并没有为所获取记录锁的上限的数量设置一个固定的上限，具体数量取决于可用的内存数量。

获取和释放记录锁的性能有多快，取决于用来维护记录锁的数据结构和具体某一把锁在数据结构的位置。这个数据结构需要快速解决下述需求：



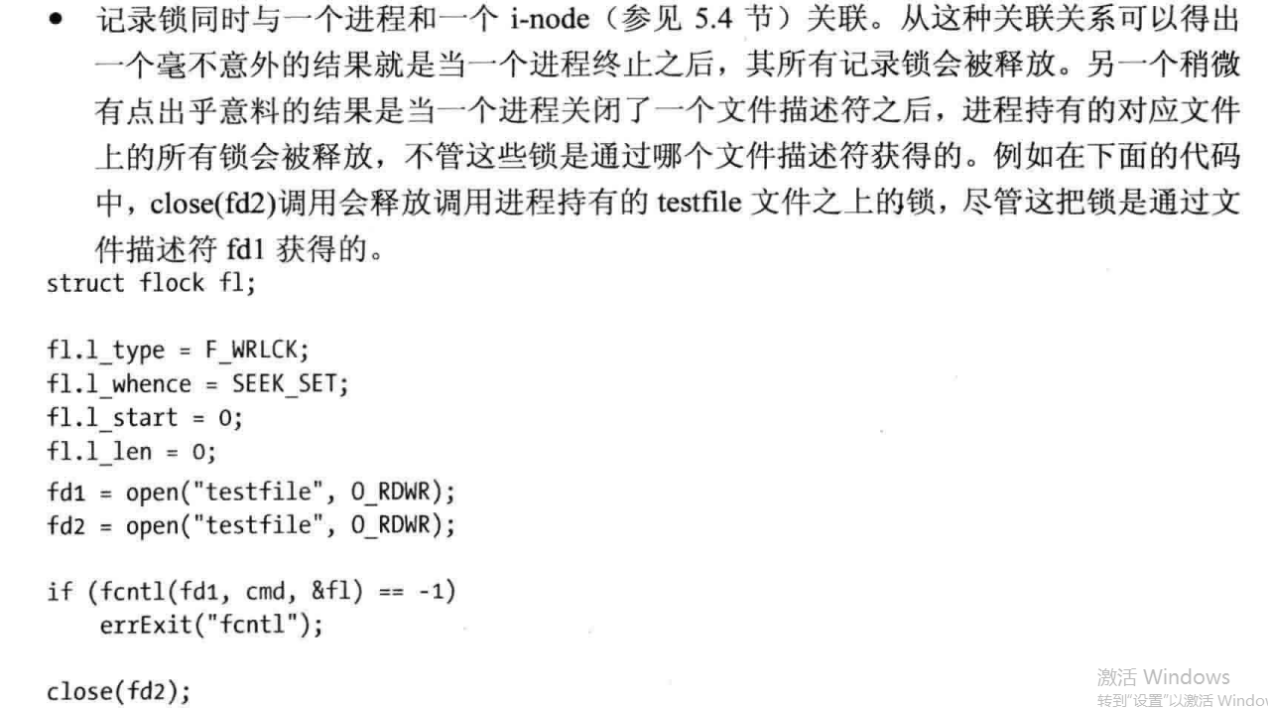
具体结构如下：

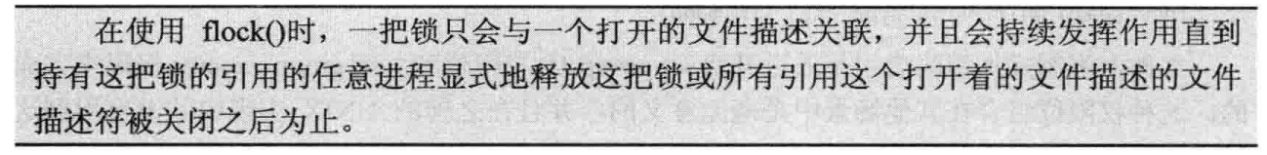




1. 锁继承和释放的语义：

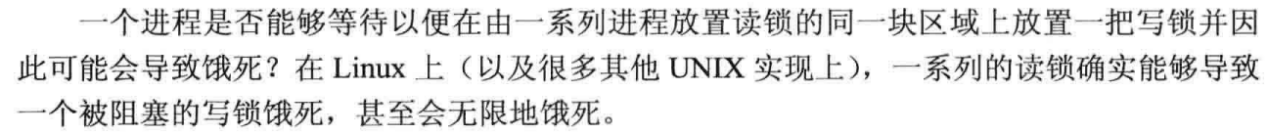
记录锁同时和一个进程和一个i-node关联。



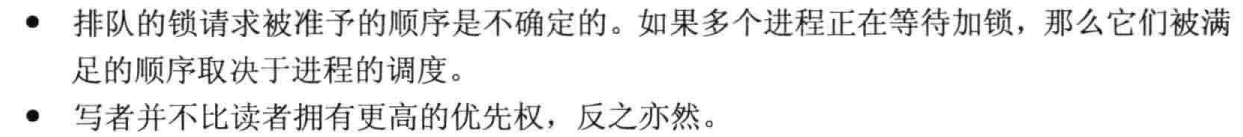


注意，fcntl锁的继承和释放语义是一个架构上的缺陷。在使用库函数包的时候库函数无法阻止调用者关闭相应文件的文件描述符。

1. 锁的饿死和排队加锁的优先级：

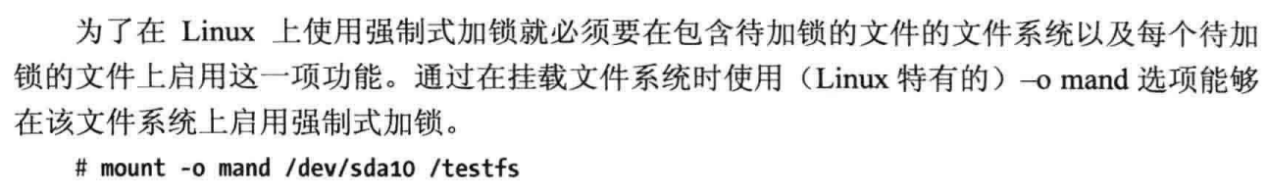


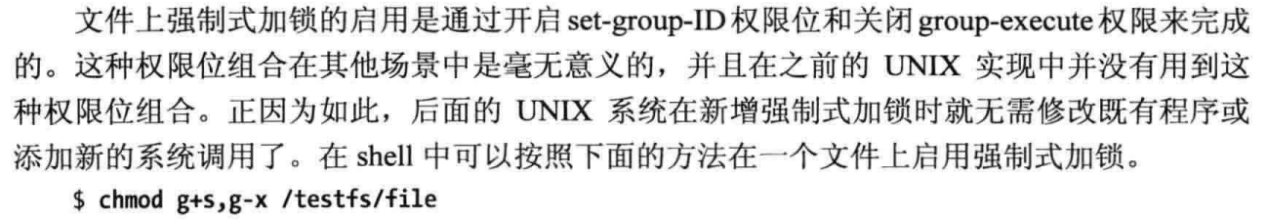
在Linux上排队的规则如下：



1. 强制加锁：

开启：

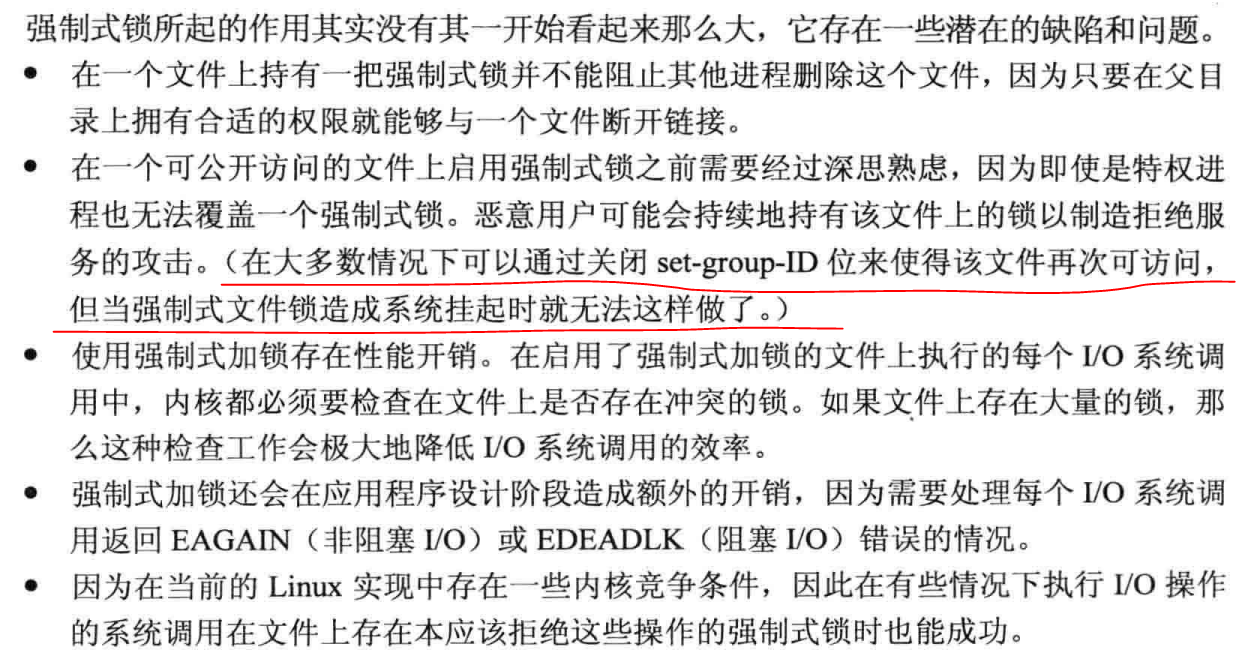




强制加锁对文件IO的影响：

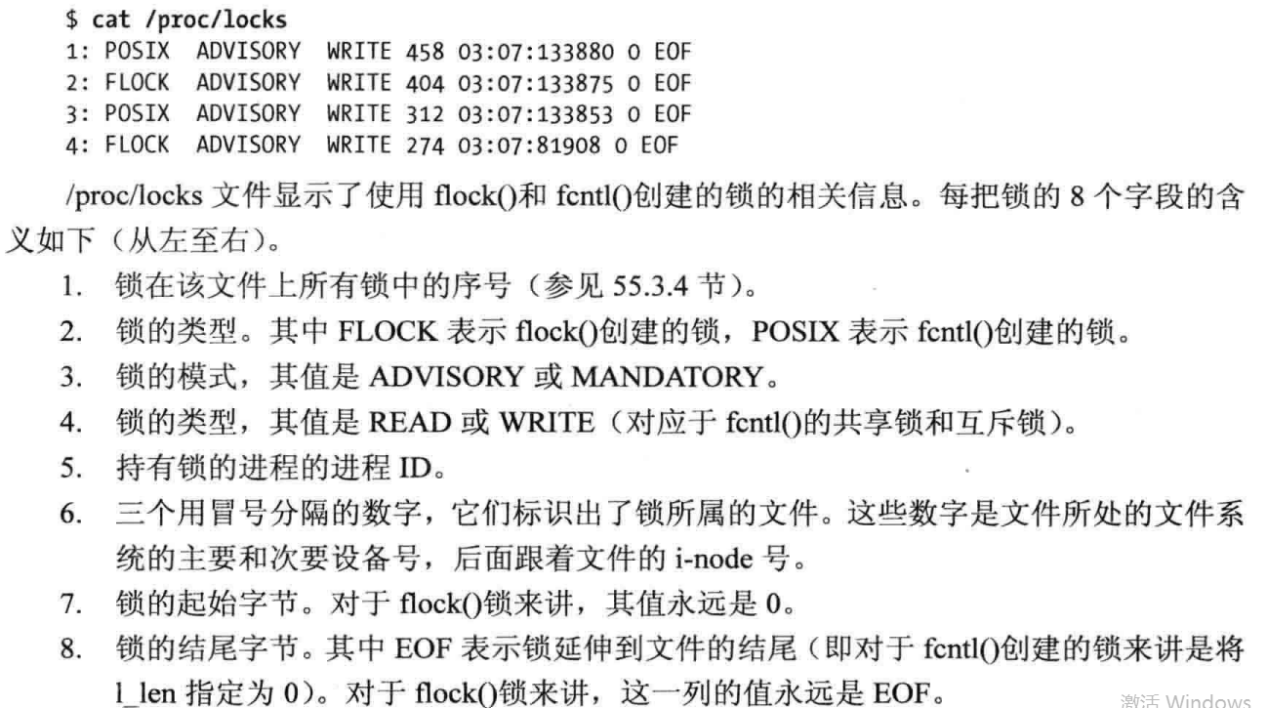
数据传输系统调用与锁冲突时会发生阻塞或EAGAIN错误。同时也会出现死锁。对于共享内存文件映射，这种情况与强加锁时矛盾的，只要其中一方存在，另一方会出错。

强制锁的缺陷：



上述红线部分不理解表达的意思。

1. /proc/locks文件：



书本给出了如何根据主设备号和从设备号以及inum查找文件，具体查看书本。

此外，这个文件还可以查看到阻塞的情况。

1. 仅运行一个程序的实例：

利用对文件加上写锁，可实现。

1. 老式的加锁技术：

有三种：







但是它们都有缺陷：

