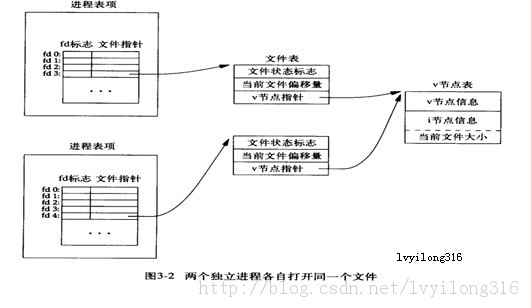
Linux两个独立进程打开同一个文件（转）

http://simg.sinajs.cn/blog7style/images/common/sg_trans.gif (2015-09-20 19:58:14)

[[http://simg.sinajs.cn/blog7style/images/common/sg_trans.gif](javascript:;)转载▼](javascript:;)

|  |  |
| --- | --- |
| 标签：  [佛学](http://search.sina.com.cn/?c=blog&q=%B7%F0%D1%A7&by=tag) | 分类： [linux](http://blog.sina.com.cn/s/articlelist_2115595863_2_1.html) |

如果两个独立进程各自打开了同一文件，则有图3-2中所示的安排。我们假定第一个进程使该文件在文件描述符3上打开，而另一个进程则使此文件在文件描述符4上打开。打开此文件的每一个进程都得到一个文件对象，但对一个给定的文件只有一个v节点表项。每个进程都有自己的文件对象的一个理由：这种安排使每个进程都有它自己对该文件的当前位移量。这种情况不会增加对应的打开文件引用计数，而会增加dentry的引用。



       给出了这些数据结构后，现在对前面所述的操作作进一步说明。

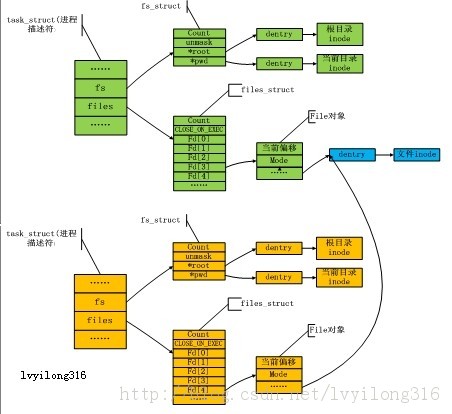
（1） 在完成每一个write后，在文件表项中的当前文件位移量即增加所写的字节数。如果这使当前文件位移量超过了当前文件长度，则在i节点表象中的当前文件长度被设置为当前文件位移量(也就是该文件加长了)。

（2）   如果用O\_APPEND标志打开一个文件，则相应标志也被设置到文件表项（file对象）的文件状态标志中。每次对这种具有填写标志的文件执行写操作时，在文件表项中的当前文件位移量首先被设置为i节点表项中的文件长度。这就使得每次写的数据都添加到文件的当前尾端处。

（3）  lseek值修改文件表项中的当前文件位移量，没有进行任何I/O操作。（不影响i节点，只影响file对象，详细分析请见lvyilong316博客：空洞文件）

（4）  若一个文件用lseek被定位到文件当前的尾端，则文件表项中的当前文件位移量被设置为i节点表项中的当前文件长度。

将图3-2转化为linux下的具体实现，如下图所示。



注：绿色部分为进程1的私有资源，黄色部分为进程2的私有资源，蓝色部分为进程1、进程2的共享资源。

扩展：

（1）用leek定位到当前文件尾端，在向文件写入（write）与使用O\_APPEND打开（open）文件再写入（write）的区别：

前者是“非原子”操作，假如两个进程都使用前者的方式向文件结尾写入数据，那么有可能产生这样的调度序列：

进程A：leek  进程B：leek 进程A：write 进程B：write

第一个进程写入后，文件（i节点）的偏移已经改变，第二个进程再写会覆盖第一个进程刚写的内容。而是用O\_APPEND的open，会使内核每次对文件写之前，都将进程的当前偏移量（file对象中的）设置到文件的尾端处（i节点的当前文件长度）。

注意：对于多个进程打开同一文件的情况，每个进程都有它自己的文件表项（file对象），其中有它自己的文件位移量，所以对于多个进程读同一文件都能正确工作。但是，当多个进程写同一文件时，则可能产生预期不到的结果。（可以使用pread，pwrite）。

总结：两个独立进程打开同一文件，对应不同的file对象，每个进程调用close只影响本进程的“打开文件计数”（file对象的引用计数）。