- v-node 表(v-node table)。同文件表一样，所有的进程共享这张 v-node 表。每个表项包含 stat 结构中的大多数信息，包括 st_mode 和 st_size 成员。

图 10-12 展示了一个示例，其中描述符 1 和 4 通过不同的打开文件表表项来引用两个不同的文件。这是一种典型的情况，没有共享文件，并且每个描述符对应一个不同的文件。

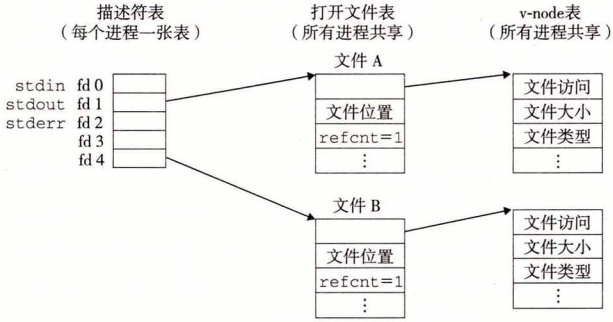

图 10-12　典型的打开文件的内核数据结构。在这个示例中，两个描述符引用不同的文件。没有共享

如图 10-13 所示，多个描述符也可以通过不同的文件表表项来引用同一个文件。例如，如果以同一个 filename 调用 open 函数两次，就会发生这种情况。关键思想是每个描述符都有它自己的文件位置，所以对不同描述符的读操作可以从文件的不同位置获取数据。

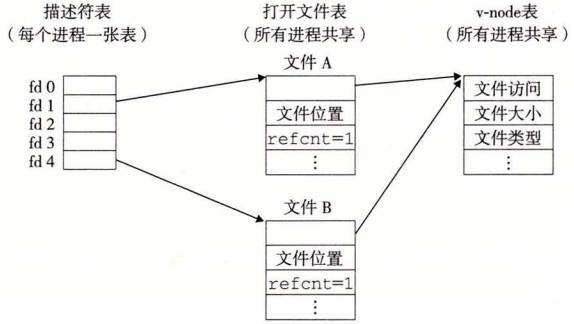

图 10-13　文件共享。这个例子展示了两个描述符通过两个打开文件表表项共享同一个磁盘文件

我们也能理解父子进程是如何共享文件的。假设在调用 fork 之前，父进程有如图 10-12 所示的打开文件。然后，图 10-14 展示了调用 fork 后的情况。子进程有一个父进程描述符表的副本。父子进程共享相同的打开文件表集合，因此共享相同的文件位置。一个很重要的结果就是，在内核删除相应文件表表项之前，父子进程必须都关闭了它们的描述符。