结束时的位置。

实现这一点的方法如图10-34所示。实现的技巧是在文件描述符表和i节点表之间引入一个新的表，叫做打开文件描述表，并将文件读写位置（以及读/写位）放到里面。在这个图中，父进程是shell而子进程首先是p1然后是p2。当shell生成p1时，p1的用户结构（包括文件描述符表）是shell的用户结构的一个副本，因此两者都指向相同的打开文件描述表的表项。当p1结束时，shell的文件描述符仍然指向包含p1的文件位置的打开文件描述。当shell生成p2时，新的子进程自动继承文件读写位置，甚至p2和shell都不需要知道文件读写位置到底是在哪里。

然而，当不相关的进程打开该文件时，它将得到自己的打开文件描述表项，以及自己的文件读写位置，而这正是我们所需要的。因此，打开文件描述表的重点是允许父进程和子进程共享一个文件读写位置，而给不相关的进程提供各自私有的值。

再来看读操作，我们已经说明了如何定位文件读写位置和i节点。i节点包含文件前12个数据块的磁盘地址。如果文件位置是在前12个块，那么这个块被读入并且其中的数据被复制给用户。对于长度大于12个数据块的文件，i节点中有一个域包含一个一级间接块的磁盘地址，如图10-34所示。这个块含有更多的磁盘块的磁盘地址。例如，如果一个磁盘块大小为1KB而磁盘地址长度是4字节，那么这个一级间接块可以保存256个磁盘地址。因此这个方案对于总长度在268KB以内的文件适用。

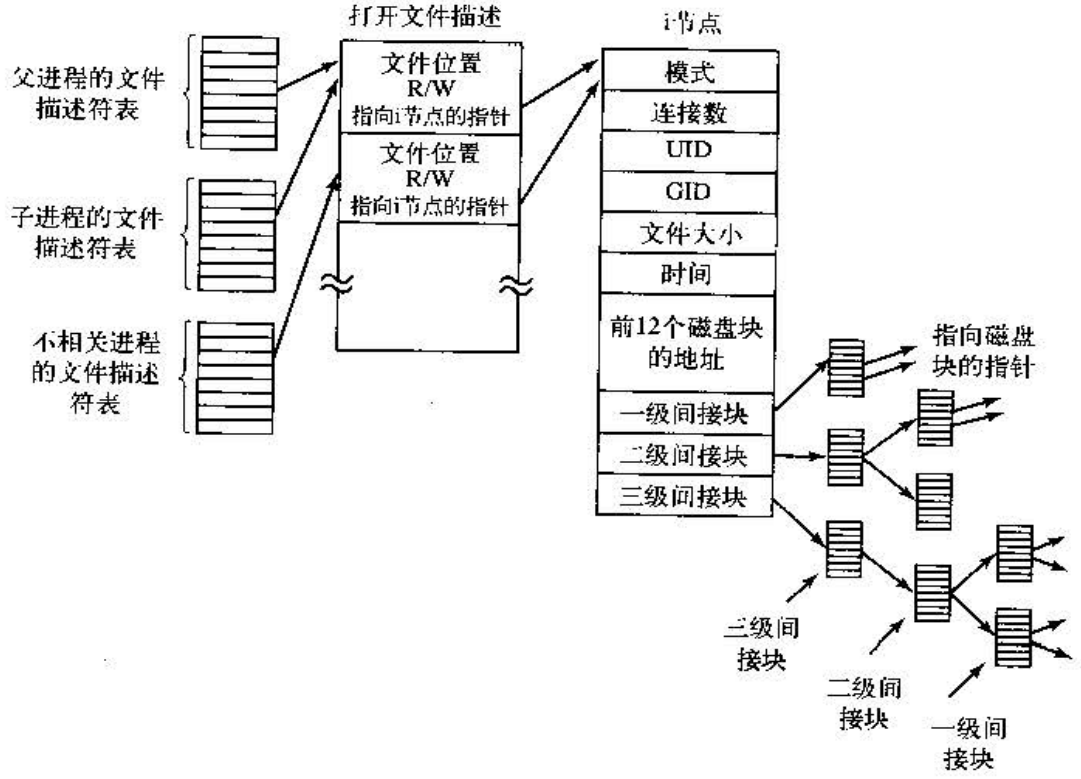

图10-34 文件描述符表、打开文件描述表和i节点表之间的关系

除此之外，还使用一个二级间接块。它包含256个一级间接块的地址，每个一级间接块保存256个数据块的地址。这个机制能够处理$10+2^{16}$个块（67 119 104字节）。如果这样仍然不够，那么i节点为三级间接块留下了空间，三级间接块的指针指向许多二级间接块。这个寻址方案能够处理大小为2^{24}个1KB块（16GB）的文件。对于块大小是8KB的情况，这个寻址方案能够支持最大64TB的文件。

3. Linux Ext3文件系统

为了防止由系统崩溃和电源故障造成的数据丢失，ext2文件系统必须在每个数据块创建之后立即将其写出到磁盘上。必需的磁盘磁头寻道操作导致的延迟是如此之长以至于性能差得无法让人接受。因此，写操作被延迟，对文件的改动可能在30秒内都不会提交给磁盘，而相对于现代的计算机硬件来说，这是一段相当长的时间间隔。