是的，完全正确！

你的理解非常准确。文件描述符的编号是相对于每个进程而言的，是进程本地的。

详细解释

让我们通过一个具体的例子来说明：

假设有两个进程：进程A（PID: 100）和 进程B（PID: 200），它们都打开了同一个文件 `/home/user/data.txt`。

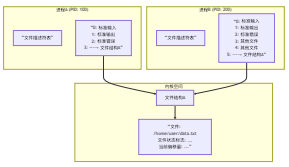
在进程A中（PID: 100）：

- 它调用 `fd1 = open("/home/user/data.txt", O_RDONLY);`
- 内核为进程A分配一个空闲的文件描述符，假设是 3。
- 在进程A的文件描述符表中，条目3指向内核中为 `/home/user/data.txt` 创建的文件结构。
- 所以，在进程A看来，这个文件的文件描述符是 3。

在进程B中（PID: 200）：

- 它调用 `fd2 = open("/home/user/data.txt", O_RDWR);`
- 内核为进程B分配一个空闲的文件描述符，假设是 5（因为0-4可能已经被进程B的其他操作占用了）。
- 在进程B的文件描述符表中，条目5指向内核中为 `/home/user/data.txt` 创建的文件结构（可能与进程A是同一个，也可能是另一个）。
- 所以，在进程B看来，这个文件的文件描述符是 5。

这个过程如下图所示：



关键要点

1. 文件描述符表是进程私有的：
   - 每个进程都有自己独立的文件描述符表。
   - 进程A的文件描述符3和进程B的文件描述符5是完全独立的，它们只是各自进程表中的索引。
2. 内核中的文件结构：
   - 虽然两个进程使用不同的文件描述符编号，但它们可能指向同一个内核文件结构。

    - 这意味着两个进程可能共享文件的读写位置等信息，也可能不共享，这取决于打开文件的方式。
3. 文件描述符的分配规则：
    - 在每个进程中，文件描述符总是分配当前可用的最小编号。
    - 这就是为什么新打开的文件描述符编号会递增，但如果有描述符被关闭后再打开，可能会重用较小的编号。
4. 继承的情况：
    - 在父进程fork()创建子进程时，子进程会继承父进程的文件描述符表。
    - 在这种情况下，同一个文件描述符编号在两个进程中指向同一个内核文件结构。

总结

文件描述符就像是每个进程自己独立的"电话号码簿"。不同进程的"电话簿"中，同一个人（文件）可能被记录在不同的页码（文件描述符编号）下。页码是每个电话簿私有的，但最终都能联系到正确的人。

这就是UNIX/Linux系统中文件描述符设计的巧妙之处——它提供了进程间的隔离，同时又通过内核的文件结构实现了必要的共享机制。