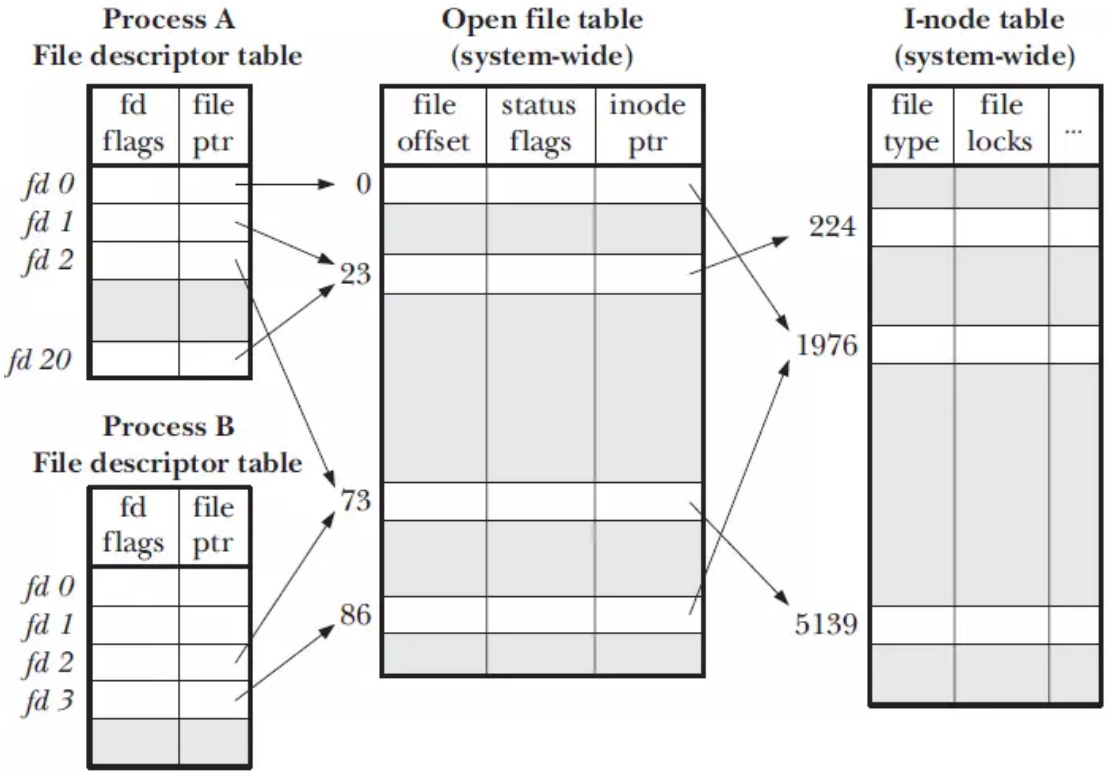
Linux 进程文件

在 Linux 通用 I/O 模型中，I/O 操作系列函数(系统调用)都是围绕一个叫做文件描述符的整数展开。

理解具体情况，需要了解由内核维护的 3 个数据结构:

1. 进程级文件描述符表(file descriptor table)
2. 系统级打开文件表(open file table)
3. 文件系统 i-node 表(i-node table)



文件描述符表

内核为**每个进程**维护一个文件描述符表，该表每一条目都记录了单个文件描述符的相关信息，包括:

1． 控制标志(flags)，目前内核仅定义了一个，即 close-on-exec

2． 打开文件描述体指针

打开文件表

内核**对所有打开的文件维护一个系统级别**的打开文件描述表(open file description table)，简称打开文件表。表中条目称为打开文件描述体(open file description)，存储了与一个打开文件相关的全部信息，包括:

1. 文件偏移量(file offset)，调用 read()和 write()更新，调用 lseek()直接修改
2. 访问模式，由 open()调用设置，例如：只读、只写或读写等
3. i-node 对象指针

i-node 表

每个文件系统会为存储于其上的所有文件(包括目录)维护一个 i-node 表，单个 i-node

包含以下信息：

1. 文件类型(file type)，可以是常规文件、目录、套接字或 FIFO
2. 访问权限
3. 文件锁列表(file locks)
4. 文件大小

等等

需要注意 Fork 进程带来的影响

在调用 Fork 之后，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副本。

造成的一个重要结果就是，在内核删除相应文件表表项之前，父子进程必须都关闭他们的描述符。