<https://blog.csdn.net/freshui/article/details/54926111>

Binder驱动暴露给应用层的ioctl指令

|  |  |  |
| --- | --- | --- |
| BINDER\_WRITE\_READ | 读写数据使用 | struct binder\_write\_read |
| BINDER\_SET\_IDLE\_TIMEOUT |  |  |
| BINDER\_SET\_MAX\_THREADS | 该命令告知Binder驱动接收方（通常是Server端）线程池中最大的线程数 |  |
| BINDER\_SET\_IDLE\_PRIORITY |  |  |
| BINDER\_SET\_CONTEXT\_MGR | 将当前线程注册为service manager |  |
| BINDER\_THREAD\_EXIT |  |  |
| BINDER\_VERSION |  | struct binder\_version |
| BINDER\_GET\_NODE\_DEBUG\_INFO |  | struct binder\_node\_debug\_info |

读写操作通过一条指令和一个结构体就可以实现，当然读写指令的内部会分成许多小的指令

|  |  |  |
| --- | --- | --- |
| BC\_TRANSACTION | Client向server发送请求数据 | struct binder\_transaction\_data |
| BC\_REPLY | Server向client发送回复数据 | struct binder\_transaction\_data |
| BC\_ACQUIRE\_RESULT | 暂未实现 |  |
| BC\_FREE\_BUFFER | 释放一块映射的内存。Binder接收方通过mmap()映射一块较大的内存空间，Binder驱动基于这片内存采用最佳匹配算法实现接收数据缓存的动态分配和释放，满足并发请求对接收缓存区的需求。应用程序处理完这片数据后必须尽快使用该命令释放缓存区，否则会因为缓存区耗尽而无法接收新数据。 | binder\_uintptr\_t |
| BC\_INCREFS  BC\_ACQUIRE  BC\_RELEASE  BC\_DECREFS | 这组命令增加或减少Binder的引用计数，用以实现强指针或弱指针的功能 |  |
| BC\_INCREFS\_DONE  BC\_ACQUIRE\_DONE | 第一次增加Binder实体引用计数时，驱动向Binder实体所在的进程发送BR\_INCREFS， BR\_ACQUIRE消息；Binder实体所在的进程处理完毕回馈BC\_INCREFS\_DONE，BC\_ACQUIRE\_DONE | struct binder\_ptr\_cookie |
| BC\_REGISTER\_LOOPER  BC\_ENTER\_LOOPER  BC\_EXIT\_LOOPER | 这组命令同BINDER\_SET\_MAX\_THREADS一道实现Binder驱动对接收方线程池管理。BC\_REGISTER\_LOOPER通知驱动线程池中一个线程已经创建了；BC\_ENTER\_LOOPER通知驱动该线程已经进入主循环，可以接收数据；BC\_EXIT\_LOOPER通知驱动该线程退出主循环，不再接收数据。 |  |
| BC\_REQUEST\_DEATH\_NOTIFICATION | 获得Binder引用的进程通过该命令要求驱动在Binder实体销毁得到通知。虽说强指针可以确保只要有引用就不会销毁实体，但这毕竟是个跨进程的引用，谁也无法保证实体由于所在的Server关闭Binder驱动或异常退出而消失，引用者能做的是要求Server在此刻给出通知 | struct binder\_handle\_cookie |
| BC\_DEAD\_BINDER\_DONE | 收到实体死亡通知书的进程在删除引用后用本命令告知驱动 | binder\_uintptr\_t |
| BC\_CLEAR\_DEATH\_NOTIFICATION |  | struct binder\_handle\_cookie |
| BC\_TRANSACTION\_SG |  | struct binder\_transaction\_data\_sg |
| BC\_REPLY\_SG |  | struct binder\_transaction\_data\_sg |