



时间 2017-03-03

栏目 Android (<http://www.voidcn.com/column/android>)

原文 <http://blog.csdn.net/hehehaha1123/article/details/60148211>

# Android系统--Binder系统具体框架分析（二）Binder驱动情景分析

## 1. Binder驱动情景分析

### 1.1 进程间通信三要素

- 源
- 目的：handle表示“服务”，即向实现该“服务”的进程发送数据；handle是“服务”的引用

```
int binder_call(struct binder_state *bs, struct binder_io *msg, struct binder_io *reply,    uint32_t target, uint32_t code)

// bs: 驱动信息

// msg: 含有服务的名字

// reply: 它会含有回复的数据

// target : 表示目的

// code : 调用的函数
```

- 数据

- Binder中的handle：进程A对进程B提供的服务S的引用

- 引用：通过struct binder\_ref结构体关联服务节点

```
struct binder_ref { //就是 refs_by_desc、refs_by_node 这两个引用树种的数据结构。

    int debug_id;

    struct rb_node rb_node_desc; //连接des的引用树。

    struct rb_node rb_node_node; //连接node的引用树。

    struct hlist_node node_entry;

    struct binder_proc *proc; //该binder引用所属的进程，

    struct binder_node *node; //和远程的binder实体binder_node关联的地方。相对应。

    uint32_t desc;

    int strong;

    int weak;

    struct binder_ref_death *death;

};
```

- 服务：创建binder\_node服务节点，指向进程B

```
struct binder_work work;

union {

    struct rb_node rb_node;

    struct hlist_node dead_node;

};

struct binder_proc *proc;

struct hlist_head refs;

int internal_strong_refs;

int local_weak_refs;

int local_strong_refs;

void __user *ptr;

void __user *cookie;

unsigned has_strong_ref : 1;

unsigned pending_strong_ref : 1;

unsigned has_weak_ref : 1;

unsigned pending_weak_ref : 1;

unsigned has_async_transaction : 1;
```

```
struct list_head async_cdbb,  
  
};
```

- 进程B：用结构体struct binder\_proc表示

```
struct rb_root threads; // 红黑树的节点，（不理解红黑树结构，暂时就当该存储数据的地方即可）
```

```
struct rb_root nodes;
```

```
struct rb_root refs_by_desc;
```

```
struct rb_root refs_by_node;
```

```
int pid; //进程的id.
```

```
struct vm_area_struct *vma;
```

```
struct mm_struct *vma_vm_mm;
```

```
struct task_struct *tsk;
```

```
struct files_struct *files;
```

```
struct hlist_node deferred_work_node;
```

```
int deferred_work;
```

```
void *buffer; //表示要映射的物理内存在内核空间中的起始位置
```

```
//内核使用的虚拟地址与进程使用的虚拟地址之间的差值，即如果某个物理页面在内核空间中对应的虚拟地址是addr的话，
```

```
//那么这个物理页面在进程空间对应的虚拟地址就为addr + user_buffer_offset
```

```
ptrdiff_t user_buffer_offset;
```

```
struct list_head buffers; //通过mmap映射的内存空间.
```

```
size_t free_async_space;

struct page **pages; // struct page 用来描述物理页面的数据结构

size_t buffer_size; //表示要映射的内存的大小.

uint32_t buffer_free;

struct list_head todo;

wait_queue_head_t wait;

struct binder_stats stats;

struct list_head delivered_death;

int max_threads;

int requested_threads;

int requested_threads_started;

int ready_threads;

long default_priority;

struct dentry *debugfs_entry;

};
```

- 真实场景中有多个客户端服务对进程B要求服务，进程B创建多个线程提供服务，用struct rb\_root结构体（红黑树）管理线程，线程用struct binder\_thread描述

```

struct rb_node rb_node; //来连入binder_proc的threads红黑树。

int pid;

int looper;//表示线程的状态 就是上面enum的类型。

struct binder_transaction *transaction_stack; //表示线程正在处理的事务

struct list_head todo; //表示发往该线程的数据列表待处理的一次通信事务。

uint32_t return_error; /* Write failed, return error code in read buf */

uint32_t return_error2; /* Write failed, return error code in read */

    /* buffer. Used when sending a reply to a dead process that */

    /* we are also waiting on */

wait_queue_head_t wait; //用来阻塞线程等待某个事件的发生

struct binder_stats stats; //用来保存一些统计信息

};

```

### 1.2.2 解析handle对Binder节点操作流程

- (1) server传入一个flat\_binder\_object结构体给驱动，在内核态驱动里为每一个服务创建binder\_node,biner\_node.proc关联描述进程的结构体
- (2) ServiceManager 在驱动中创建binder\_ref结构体，引用binder\_node服务节点
  - 在用户态创建服务链表 ( name,handle )
  - binder\_ref.desc关联handle
- (3) client向ServiceManager传入name查询对应服务
- (4) servicemanager返回hanle给驱动程序

---

## 1.3 client与server数据传输过程

- client端（先读后写）

- （1）client构造数据，调用ioctl发送数据
- （2）驱动根据handle找到server进程
- （3）将数据存入进程的binder\_proc.todo
- （4）等待唤醒（等待server传回数据）
- （5）被唤醒
- （6）从todo链表中取出数据，返回用户空间

- server端（先写后读）

- （1）等待数据传入，休眠
- （2）client有数据写入，唤醒
- （3）从binder\_proc.todo链表当中取出数据，返回用户空间
- （4）进行数据处理
- （5）将结果写给client，也是放入client中的binder\_proc.todo
- （6）唤醒client

## 1.4 数据复制详解

- 一般方法（需要两次）

- （1）client构造数据
- （2）client：copy\_from\_user
- （3）server：copy\_to\_user
- （4）用户态处理

- Binder方法(一次)



- 
- 描述一段通过mmap映射的内存空间--结构体binder\_buffer来描述

//空闲的binder\_buffer通过成员变量rb\_node连入到binder\_proc中的free\_buffers表示的红黑树中。

//正在使用的binder\_buffer通过成员变量rb\_node连入到binder\_proc中的allocated\_buffers表示的红黑树中去。

```
struct rb_node rb_node;
```

unsigned free:1; //每一个binder\_buffer又分为正在使用的和空闲的，通过free成员变量来区分。

```
unsigned allow_user_free:1;
```

```
unsigned async_transaction:1;
```

```
unsigned debug_id:29;
```

```
struct binder_transaction *transaction;
```

```
struct binder_node *target_node;
```

```
size_t data_size;
```

```
size_t offsets_size;
```

```
uint8_t data[0];
```

```
};
```

注：一次数据复制是针对传输数据，其binder\_write\_read还是需要两次复制。client调用ioctl发送数据，ioctl发送的是binder\_write\_read结构，其结构体中的成员指向传送数据。

## 1.5 改进binder分析一代码

```
//注册

if (ret) {

    fprintf(stderr, "failed to publish hello service\n");

    return -1;

}

ret = svcmgr_publish(bs, svcmgr, "goodbye", goodbye_service_handler);

if (ret) {

    fprintf(stderr, "failed to publish goodbye service\n");

}
```

- 将被调用函数指向handler , 返回handler函数

```
int (*handler)(struct binder_state *bs,struct binder_transaction_data *txn,struct binder_io *msg,struct binder_io *reply); //构造处理方法

handler = (int (*)(struct binder_state *bs,struct binder_transaction_data *txn,struct binder_io *msg,struct binder_io *reply))txn->target.ptr; //根据txn->target.ptr返回处理相应方法

return handler(bs, txn, msg, reply); //返回所调用的处理方法

}
```

#### 相关文章

- 1. Android系统--Binder系统具体框架分析（二）Binder驱动情景分析 (<http://www.voidcn.com/article/p-qchghybj-bhy.html>)
- 2. Android系统服务启动分析-binder (<http://www.voidcn.com/article/p-xlcrbpic-py.html>)
- 3. Binder 子系统之调试分析(二) (<http://www.voidcn.com/article/p-ftymypwm-sc.html>)
- 4. Android Binder 驱动分析 (<http://www.voidcn.com/article/p-oykndkok-na.html>)
- 5. Android系统的Binder机制分析 (<http://www.voidcn.com/article/p-esnyxuiq-bmh.html>)
- 6. Android系统中基于Binder的IPC流程框架分析 (<http://www.voidcn.com/article/p-upkxvuv-tsd.html>)
- 7. android binder驱动源码分析（二） (<http://www.voidcn.com/article/p-wjnciyao-y.html>)
- 8. Binder子系统之调试分析(一) (<http://www.voidcn.com/article/p-vobednfh-ep.html>)
- 9. Binder 子系统之调试分析(三) (<http://www.voidcn.com/article/p-boofyvfv-sc.html>)
- 10. Binder 子系统之调试分析(一) (<http://www.voidcn.com/article/p-suefnbqr-sc.html>)
- 更多相关文章... (<http://www.voidcn.com/relative/p-rctislhv-mz.html>)

#### 相关标签/搜索

android binder 分析 (<http://www.voidcn.com/tag/android+binder+%E5%88%86%E6%9E%90>) 系统分析  
(<http://www.voidcn.com/tag/%E7%B3%BB%E7%BB%9F%E5%88%86%E6%9E%90>) 分析系统  
(<http://www.voidcn.com/tag/%E5%88%86%E6%9E%90%E7%B3%BB%E7%BB%9F>) Android Binder驱动  
(<http://www.voidcn.com/tag/Android+Binder%E9%A9%B1%E5%8A%A8>) Android系统源代码情景分析  
(<http://www.voidcn.com/tag/Android%E7%B3%BB%E7%BB%9F%E6%BA%90%E4%BB%A3%E7%A0%81%E6%83%85%E6%99%AF%E5%88%86%E6%9E%90>)  
Android源码分析 Binder (<http://www.voidcn.com/tag/Android%E6%BA%90%E7%A0%81%E5%88%86%E6%9E%90+Binder>) AWStats分析系统  
(<http://www.voidcn.com/tag/AWStats%E5%88%86%E6%9E%90%E7%B3%BB%E7%BB%9F>) 系统分析员  
(<http://www.voidcn.com/tag/%E7%B3%BB%E7%BB%9F%E5%88%86%E6%9E%90%E5%91%98>) 系统分析师  
(<http://www.voidcn.com/tag/%E7%B3%BB%E7%BB%9F%E5%88%86%E6%9E%90%E5%B8%88>) 系统架构分析  
(<http://www.voidcn.com/tag/%E7%B3%BB%E7%BB%9F%E6%9E%B6%E6%9E%84%E5%88%86%E6%9E%90>) Binder分析 (<http://www.voidcn.com/cata/2607473>)

---

android系统启动死机分析 (<http://www.voidcn.com/search/pzpxme/>) go 类型系统分析 (<http://www.voidcn.com/search/siotve/>) logstash 分析系统messages (<http://www.voidcn.com/search/vgleit/>) NuttX\_编译系统分析 (<http://www.voidcn.com/search/pukuac/>) python股票分析系统 (<http://www.voidcn.com/search/olkxdd/>) android 蓝牙系统框架 (<http://www.voidcn.com/search/hcrwhy/>) android 图形系统 框架 (<http://www.voidcn.com/search/rssawq/>) XUtils框架分析 (<http://www.voidcn.com/search/trscok/>)

0

分享到微博

分享到微信

分享到QQ

每日一句

每一个你不满意的现在，都有一个你没有努力的曾经。

#### 最新文章

- 1. virt-install参数详解 (<http://www.voidcn.com/article/p-stcmwgs-bsd.html>)
- 2. MySQL集群搭建--多主模式 (<http://www.voidcn.com/article/p-wsxwzbau-bsd.html>)
- 3. linux 环境下安装mysql---ubuntu (<http://www.voidcn.com/article/p-tbthnvuj-bsd.html>)
- 4. 持续集成与持续部署宝典Part 1：将构建环境容器化 (<http://www.voidcn.com/article/p-mfganztd-bsd.html>)
- 5. Exchange Server 2019 公共预览版发布 (<http://www.voidcn.com/article/p-yhtxnnjz-bsd.html>)
- 6. Skype for Business Server 2019公开预览版发布 (<http://www.voidcn.com/article/p-zkpkbtj-bsd.html>)
- 7. 《11招玩转网络安全》之第五招：DVWA命令注入 (<http://www.voidcn.com/article/p-uonapr-bsd.html>)
- 8. 剖析LNMP架构 (<http://www.voidcn.com/article/p-xvukeond-bsd.html>)
- 9. 为Django网站添加favicon.ico图标 (<http://www.voidcn.com/article/p-qhuaoumg-bsd.html>)
- 10. 跟着动画来学习TCP三次握手和四次挥手 (<http://www.voidcn.com/article/p-vnqdkxta-bsd.html>)

公众号推荐 (</contact>)



本站公众号 (/contact)

欢迎关注本站公众号,获取更多程序园信息



相关文章