**Parcel**

参考：<http://blog.csdn.net/u010794180/article/details/43791691>

https://www.jianshu.com/p/82e3090e00e0

Parcel是andoid中特有的不同于java中Serialize的序列化机制。由于移动设备上内存的限制，所以在序列化和反序列化机制的性能上就要求的更高。而android推荐的Parcel就是轻量级并且高效的序列化和反序列化机制。在android系统的IPC通信中就是使用Parcel类进行序列化和反序列化来完成客户端和服务端传递通信。

**API使用**

首先介绍它的里面的方法的作用（参考网上资料）：

obtain() 获得一个新的parcel对象，相当于java中new一个对象

dataSize() 得到当前parcel对象的实际存储空间

dataCapacity() 得到当前parcel对象的已分配的存储空间,该值大于或等于dataSize()返回值 。

dataPostion() 获得当前parcel对象的偏移量(类似于文件流指针的偏移量)

setDataPosition() 设置偏移量（类似于移动指针到特定位置）

recyle() 清空并回收parcel对象所占内存

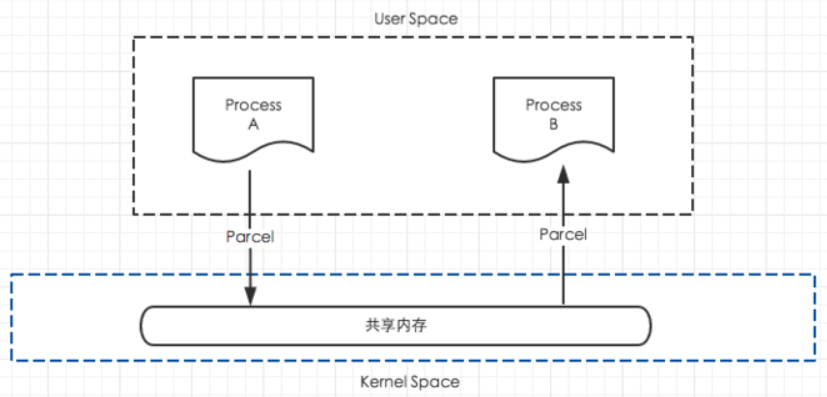
writeXXX（）方法表示写于一个XXX类型的数

readXXX（）方法表示读取一个XXX类型的数

比如：writeInt(int) 写入一个整数；readInt（）读取一个整数。

**机制**

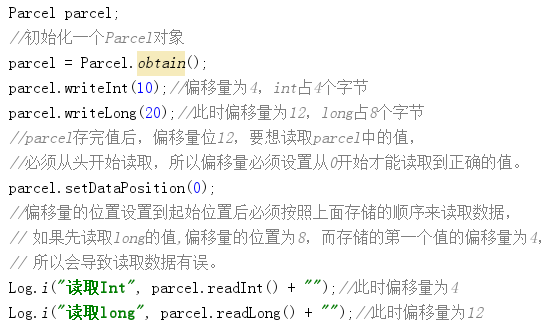
Parcel将序列化后的数据写入到一个共享内存中，然后其他进程通过Parcel可以在共享内存中反序列化出这些数据生成一个对象。注意这块共享内存是存在于内核空间的，而进程是存在于用户空间的。这就是与Serializable机制不同的地方，Parcel是将数据序列化到内存中，而Serializable将数据序列化到本地硬盘文件中，所以Parcel的效率更高。借用网络上一个图：



那么它到底是怎么读取和存入的呢？

Parcel将数据序列化到内核空间的内存中，读取数据的最小单位是32bit（位），即4个字节。所以当我们写入一个字节时它的偏移量最少为4byte(32bit)

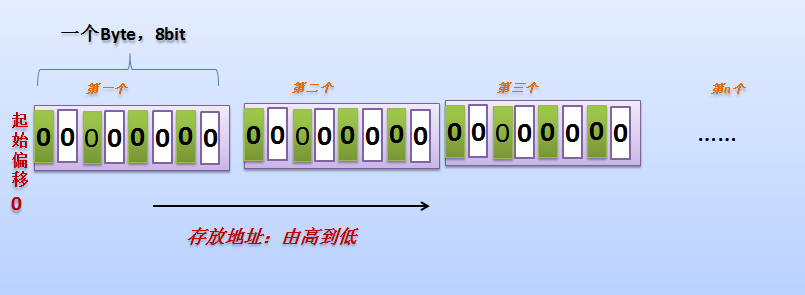
另外，可以通过setDataPostion来直接设置偏移量，但是如果设置的不对则读取数据也会有误，所以Parcel必须按照存储数据的顺序来读取数据。另外对于同一个对象它的偏移量是共用的，即writeXXX()和readXXX()时会导致偏移量的变化：



读取结果：

1515135804(1)

下面引用网上的一张Parcel内部存放值的图：



上面只是对java的基本数据类型进行序列化和反序列化，那么对于一个对象也肯定能进行序列化和反序列化，那就是实现Parcelable接口即可，它最终也是序列化对象里的基本数据类型。