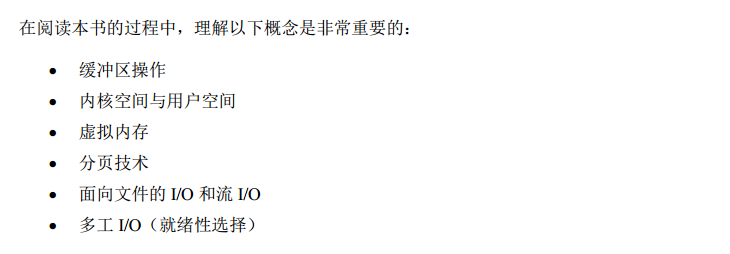
Selector:

我的理解为一层抽象，最开始是一个线程负责自己的IO事件，数据双向传输

现在是专门找一个人负责IO事件（Selector），一个人负责数据的传输（Channel），Channel和开发人员通过buffer将字节流与java对象进行转换。只有转换成了java对象，才有可能对数据进行处理。

K：JVM在追求统一的时候，也是在抛弃操作系统提供的特性，而很多特性是我们需要的。比如并发，IO等等。nio基于Linux epoll的特性，zero copy也是基于Linux系统的特性。



缓冲区：所谓输入输出无非是把数据移进或移除缓冲区，首先为什么需要缓冲区，我的理解：

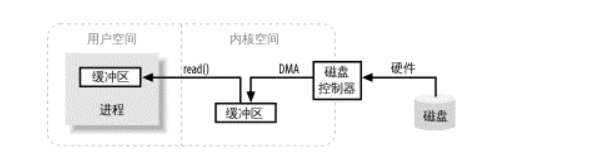
1. 程序直接处理数据，对于IO密集型会有问题，当处理速度小于IO速度，怎么办呢，就需要缓冲区先将数据缓冲起来
2. Q：如果数据填满了缓冲区怎么办，丢掉么

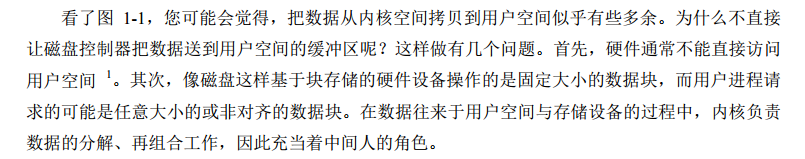
Q:用户空间

A：用户空间是常规进程所在区域，JVM就是常规进程，用户空间的代码不能直接访问硬件设备，内核空间是操作系统的所在区域，内核代码具有特别的权利，他能与设备控制器通讯，控制着用户区域进程的运行状态，最重要的是，所有的IO连接都直接通过内核空间。

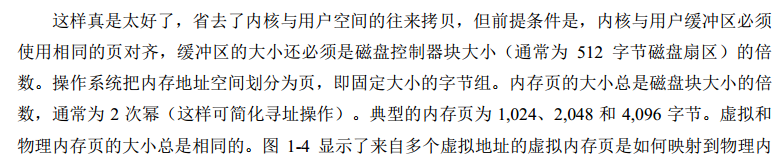
关于内核空间和用户空间确实要清楚：

对于程序员而言，你只需要知道，你所编写的代码基本上都是运行在用户空间的，而用户空间是无法访问硬件设备包括磁盘，网卡等，须有内核空间通过DMA通过磁盘控制器从磁盘读取放入缓存区内在刷到用户空间的缓存区，中间会进行数据的复制（缓冲区到缓冲区传递是复制）。read（）操作并不是调用的时候就有数据，而是要求其缓冲区被填满，这个时候，内核再去取数据，取到通过DMA，DMA通过磁盘控制器。（使用DMA的好处就是不需要通过CPU）





为什么不让DMA直接写到用户空间缓冲区的原因



这里给了提示，在设计缓冲大小的时候为什么总是1024字节的原因，讲到这里，应该很清楚了吧

其实如果没有虚拟内存，多大的内存空间似乎就只能有多大的硬盘

Q:块IO和流IO的区别

作者这么说的话，NIO应该从设计上主要倾向于流，而对于流，就少不了通道这个角色

CH2 缓冲区