# Unix的5种I/O模型

1. 阻塞I/O模型：最常用的I/O模型，如果当前请求的资源不可用，I/O会一直阻塞，直到得到反馈结果。
2. 非阻塞式I/O:如果当前请求的资源不可用，返回一个资源不可用标识，Unix利用轮询机制来查询当前的资源是否被准备好。
3. I/O复用模型：linux提供select/poll，把一个或多个fd（文件描述符）传递给select/poll系统，阻塞在select操作，select会用轮询机制来检查是否有fd处于就绪状态，如果有就立即调用回调函数。
4. 信号I/O驱动模型：通过系统调用一个信号处理函数，进程继续进行，当数据准备好的时候，向该进程返回一个SIGIO信号。进程收到信号调用数据。
5. 异步I/O:进程告知内核启动某个操作，并在操作完成后通知我们。与信号I/O的区别是，一个是信号I/O告知我们，一个是内核告知我们。

# I/O多路复用

把多个请求都送到selector，然后通过轮询的机制来查询哪个I/O准备好了，就完成这个I/O请求。

# NIO

1. buffer

缓冲区buffer就是一个对象，在NIO库中，所有的数据操作都是跟缓冲区打交道的，读，写数据都是读入缓冲区和写入缓冲区。每个java基本类型都有一个channel类型。最常用的是ByteBuffer。其实质是一个数组。

1. Channel

网络数据都是通过Channel读取和写入的。它和流的区别是它是全双工的，流是单向的。它的优点是更好的映射了底层操作系统的通道，因为它也是全双工的。

Channel分为两类，一个是读写网络数据的SelectableChannel,另一个是读写文件数据的FileChannel.

1. 多路复用器Selector

多路复用器提供选择已经就绪的任务的能力，selector会不断的轮询注册其上的Channel,如果某个Channel上面发生读或者写的事件，这个Channel就会处于就绪状态，会被selector轮询出来，然后通过SelectionKey可以获取就绪Channel的集合，进行后续的IO操作。

## NIO的优点

1.客户端发起的连接操作是异步的，可以通过多路复用注册OP\_CONNECT等待后续结果，不需要像之前的客户端那样被阻塞。

2.SocketChannel的读写操作都是异步的，如果没有读写的数据，它不会同步等待，直接返回，这样IO通信线程就可以处理其他的链路，不需要同步等待这个链路可用。

3.线程模型的优化：由于JDK的selector在linux等主流操作系统上通过epoll实现，它没有连接句柄数的限制，直意味着一个Selector线程可以处理成千上万个客户端连接，而且性能不会随着客户端的增加而下降，因此，它非常的适合做高性能，高负载的网络服务器。

# AIO

NIO2.0引入了新的异步通道的概念，并提供了异步文件通道和异步套接字通道的实现。

异步通道提供以下两种方式获取操作结果。

1. 通过java.util.concurrent.Futrue类来表示异步操作的结果。
2. 在执行异步操作的时候传入一个java.nio.channels。

CompletionHandler接口的实现类作为操作完成的回调。

它与NIO的区别，NIO有一个专门的selector线程来轮询所有的就绪请求，但是AIO没有，如果某个IO请求的数据准备好了，它是通过JDk底层的ThreadLocal来通知的，而不是某个线程来通知的。所以不需要单独开辟线程来管理这些请求。

# I/O的底层原理