Reactor和preactor都是IO多路复用模式，一般地,I/O多路复用机制都依赖于一个事件多路分离器(Event Demultiplexer)。分离器对象可将来自事件源的I/O事件分离出来，并分发到对应的read/write事件处理器(Event Handler)。开发人员预先注册需要处理的事件及其事件处理器（或回调函数）。

Reactor模式采用同步IO，而Proactor采用异步IO。同步和异步是针对应用程序和内核的交互而言的，同步指的是用户进程触发IO操作并等待或者轮询的去查看IO操作是否就绪，而异步是指用户进程触发IO操作以后便开始做自己的事情，而当IO操作已经完成的时候会得到IO完成的通知（异步的特点就是通知）。

而阻塞和非阻塞是针对于进程在访问数据的时候，根据IO操作的就绪状态来采取的不同方式，说白了是一种读取或者写入操作函数的实现方式，阻塞方式下读取或者写入函数将一直等待，而非阻塞方式下，读取或者写入函数会立即返回一个状态值。

同步阻塞： 在此种方式下，用户进程在发起一个IO操作以后，必须等待IO操作的完成，只有当真正完成了IO操作以后，用户进程才能运行。	**同步非阻塞：** 在此种方式下，用户进程发起一个I 但是用户进程需要时不时的询问IO 不停的去询问，从而引入不必要的C
异步阻塞： 此种方式下是指应用发起一个IO操作以后，不等待内核IO操作的完成，等内核完成IO操作以后会通知应用程序，这其实就是同步和异步最关键的区别，同步必须等待或者主动的去询问IO是否完成，那么为什么说是阻塞的呢？因为此时（通知）是通过select系统调用来完成的，而select函数本身的实现方式是阻塞的，而采用select函数有个好处就是它可以同时监听多个文件句柄（就绪的没有就绪的都有监听，epoll是select的替代方式，只监听就绪的文件句柄），从而提高系统的并发性！	**异步非阻塞：** 在此种模式下，用户进程只需要发起 IO操作真正的完成以后，应用程序会 用户进程只需要对数据进行处理就好 作，因为真正的IO读取或者写入操作

reactor	**proactor**
1. 应用程序注册读/写就绪事件和相关联的事件处理器 2. 事件分离器等待事件的发生 (Reactor负责) 3. 当发生读就绪事件的时候，事件分离器调用第一步注册的事件处理器(Reactor负责) 4. 事件处理器首先执行实际的读取操作，然后根据读取到的内容进行进一步的处理(用户处理器负责)	1. 应用程序初始化一个异步读取操作 此时事件处理器不关注读取就绪事件 区别于Reactor的关键。 2. 事件分离器等待读取操作完成事件 3. 在事件分离器等待读取操作完成的 成读取操作（异步IO都是操作系统负 缓冲区供应用程序操作，操作系统拷 容放入用户传递过来的缓存区中。这 Proactor中，应用程序需要传递缓存 4. 事件分离器捕获到读取完成事件后 器，事件处理器直接从缓存区读取数 作。

与reactor相比，proactor显然系统调用更少。

从上面可以看出，Reactor和Proactor模式的主要区别就是真正的读取和写入操作是有谁来完成的，Reactor中需要应用程序自己读取或者写入数据，而Proactor模式中，应用程序不需要进行实际的读写过程，它只需要从缓存区读取或者写入即可，操作系统会读取缓存区或者写入缓存区到真正的IO设备.

综上所述，同步和异步是相对于应用和内核的交互方式而言的，同步需要主动去询问，而异步的时候内核在IO事件发生的时候通知应用程序，而阻塞和非阻塞仅仅是系统在调用系统调用的时候函数的实现方式而已。