内核线程：

对于一切的进程，无论是系统进程还是用户进程，进程的创建和撤销，以及I/O操作都是利用系统调用进入到内核，由内核处理完成，所以说在KST下，所有进程都是在操作系统内核的支持下运行的，是与内核紧密相关的。内核空间实现还为每个内核支持线程设置了一个线程控制快，内核是根据该控制快而感知某个线程是否存在，并加以控制。

优点：

在多处理器上，内核可以调用同一进程中的多个线程同时工作；

如果一个进程中的一个线程阻塞了，其他线程仍然可以得到运行；

缺点：

对于用户线程的切换代价太大，在同一个线程中，从一个线程切换到另一个线程时，需要从用户态，进入到内核态并且由内核切换。因为线程调度和管理在内核实现。

用户线程：

用户进程ULT仅存在于用户空间中。对于这种线程的创建、撤销、线程之间的同步和通信等功能，都无需系统调用来实现。对于同一进程的线程之间切换仍然是不需要内核支持的。所以呢，内核也会是完全不会知道用户级线程的存在。

但是有一点必须注意：设置了用户级线程的系统，其调度荏苒是以进程为单位进行的哦。

优点：

线程切换不需要转换到内核空间，节省了宝贵的内核空间；

调度算法可以是进程专用，由用户程序进行指定；

用户级线程实现和操作系统无关；

缺点:

系统调用阻塞，同一进程中一个线程阻塞和整个进程都阻塞了。

一个进程只能在一个cpu上获得执行。

同步IO和异步IO模型是针对用户线程和内核的交互来说的：

　　对于同步IO：当用户发出IO请求操作之后，如果数据没有就绪，需要通过用户线程或者内核不断地去轮询数据是否就绪，当数据就绪时，再将数据从内核拷贝到用户线程；

　　而异步IO：只有IO请求操作的发出是由用户线程来进行的，IO操作的两个阶段都是由内核自动完成，然后发送通知告知用户线程IO操作已经完成。也就是说在异步IO中，不会对用户线程产生任何阻塞。

事实上，同步和异步是一个非常广的概念，它们的重点在于多个任务和事件发生时，一个事件的发生或执行是否会导致整个流程的暂时等待。

阻塞就是：当某个事件或者任务在执行过程中，它发出一个请求操作，但是由于该请求操作需要的条件不满足，那么就会一直在那等待，直至条件满足；

　　非阻塞就是：当某个事件或者任务在执行过程中，它发出一个请求操作，如果该请求操作需要的条件不满足，会立即返回一个标志信息告知条件不满足，不会一直在那等待。

　　这就是阻塞和非阻塞的区别。也就是说阻塞和非阻塞的区别关键在于当发出请求一个操作时，如果条件不满足，是会一直等待还是返回一个标志信息。

这个博客有的地方讲得挺清楚：http://www.cnblogs.com/dolphin0520/p/3916526.html