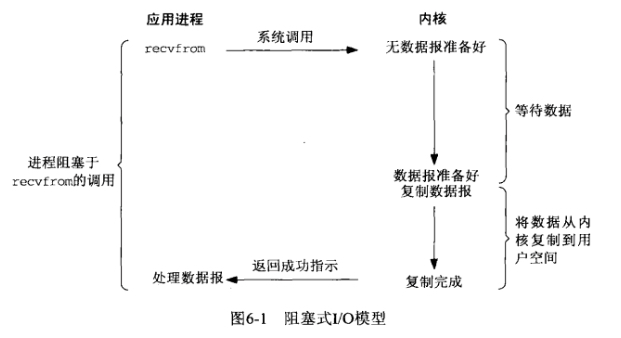
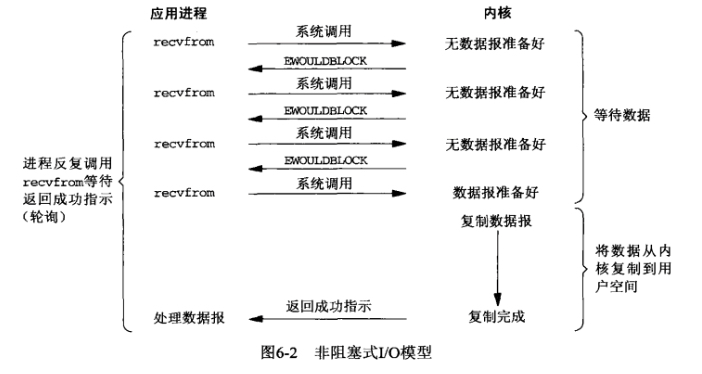
参考内容：<https://www.cnblogs.com/felixzh/p/10345929.html>

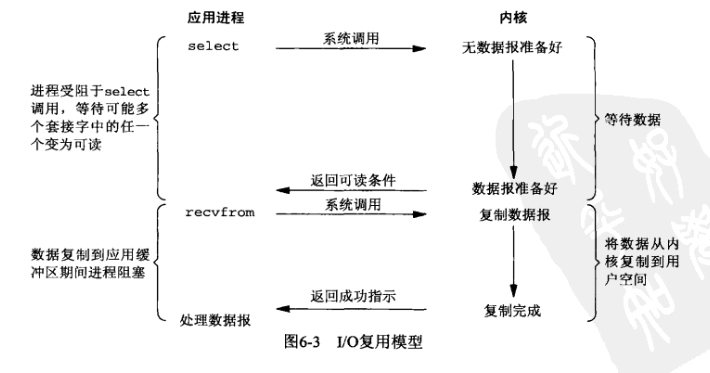
1）阻塞IO：当用户线程发出 IO 请求之后，内核会去查看数据是否就绪，如果没有就绪就会等待数据就绪，而用户线程就会处于阻塞状态，用户线程交出 CPU。当数据就绪之后，内核会将数据拷贝到用户线程，并返回结果给用户线程，用户线程才解除 block 状态。例子：data = socket.read()



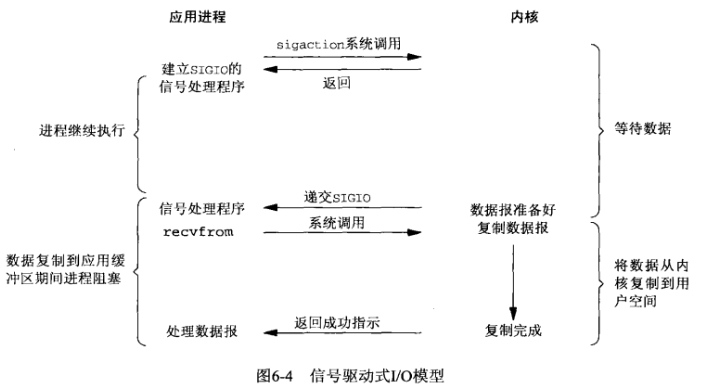
2）非阻塞IO：当用户线程发起一个 read 操作后，并不需要等待，而是马上就得到了一个结果。如果结果是一个error 时，它就知道数据还没有准备好，于是它可以再次发送 read 操作。一旦内核中的数据准备好了，并且又再次收到了用户线程的请求，那么它马上就将数据拷贝到了用户线程，然后返回。所以事实上，在非阻塞 IO 模型中，用户线程需要不断地询问内核数据是否就绪，也就说非阻塞 IO不会交出 CPU，而会一直占用 CPU。



3）多路复用IO：会有一个线程不断去轮询多个 socket 的状态，只有当 socket 真正有读写事件时，才真正调用实际的 IO 读写操作。相比于非阻塞IO，不会一直占用CPU。缺点：一旦事件响应体很大，那么就会导致后续的事件迟迟得不到处理。



4）信号驱动IO：当用户线程发起一个 IO 请求操作，会给对应的 socket 注册一个信号函数，然后用户线程会继续执行，当内核数据就绪时会发送一个信号给用户线程，用户线程接收到信号之后，便在信号函数中调用 IO 读写操作来进行实际的 IO 请求操作。



5）异步IO：只需要先发起一个请求，当接收内核返回的成功信号时表示 IO 操作已经完成，可以直接

去使用数据了。

