## 认识NIO

#### 一．Linux IO模型

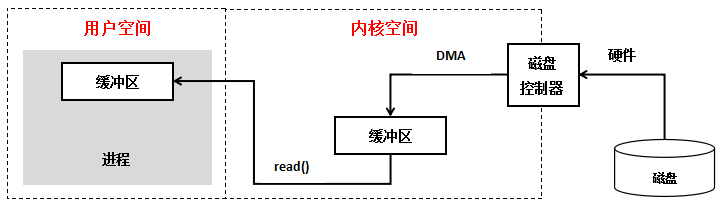
在《Unix网络编程》这本书中将IO模型划分为以下五种：

1. 阻塞式IO模型Blocking IO
2. 非阻塞式IO模型Non-Blocking IO/New IO
3. IO复用
4. 信号驱动式IO模型
5. 异步IO模型Asynchronous IO

其中前四种都是同步模式。

#### 了解Linux IO流程

在Linux中运行的应用程序如果需要进行IO操作，需要涉及到两个空间的概念，用户空间和内核空间。



上图表示的是一个数据读取的过程：

1. DMA先从将磁盘数据拷贝到内核空间，该过程也称为数据准备过程；
2. 应用程序拷贝内核空间数据到用户空间。

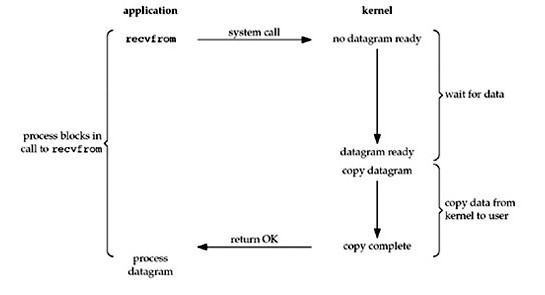
从应用程序角度来看，分为两步：

1. 等待内核将数据准备好（Waiting for the data to be ready）
2. 从内核向进程复制数据（Copying the data from the kernel to the process）

#### 各IO模型执行过程

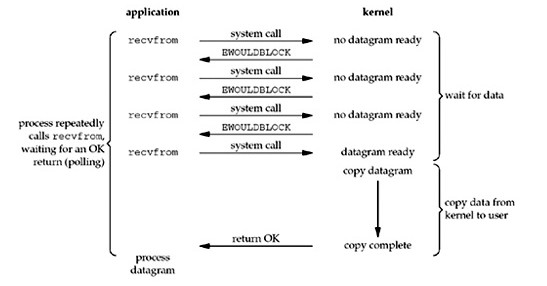
1. 阻塞式IO模型

应用程序发起一次系统调用，即IO请求，询问内核数据是否准备好，发现没有，则进程一直等待数据准备好为止；在接收到系统调用后内核开始准备数据报，准备好后进行内核态到用户态的数据拷贝，拷贝完成后，返回准备就绪信息给调用进程。



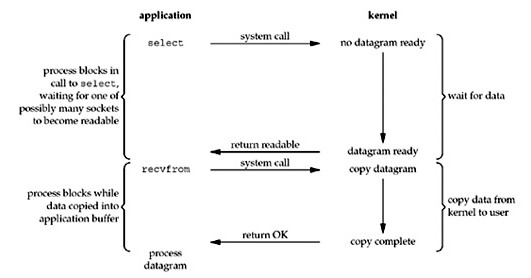
1. 非阻塞式IO模型

应用进程轮询询问内核数据是否准备好。当有数据报准备好时就进行数据报的拷贝操作，当没有准备好时，内核直接返回未准备就绪的信号，不让进程阻塞，并且开始准备数据，等待进程下一次询问。



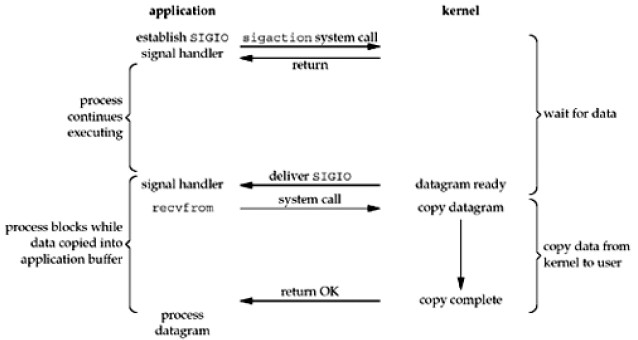
1. IO多路复用模型

IO多路复用模型会通过一个select函数对多个文件描述符（集合）进行循环监听，当某个文件描述符就绪时，就对这个文件描述符的数据进行处理。



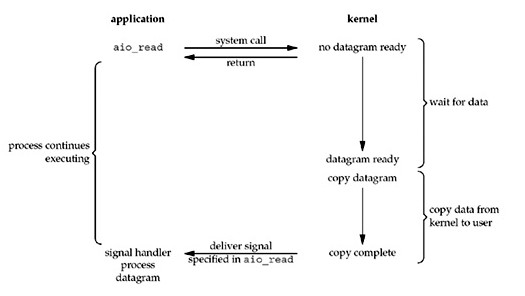
1. 信号驱动式IO模型

应用程序通知内核，当数据准备就绪时给它发送一个SIGIO信号，应用程序会对这个信号进行捕捉，并且调用相应的信号处理函数执行数据拷贝的过程。

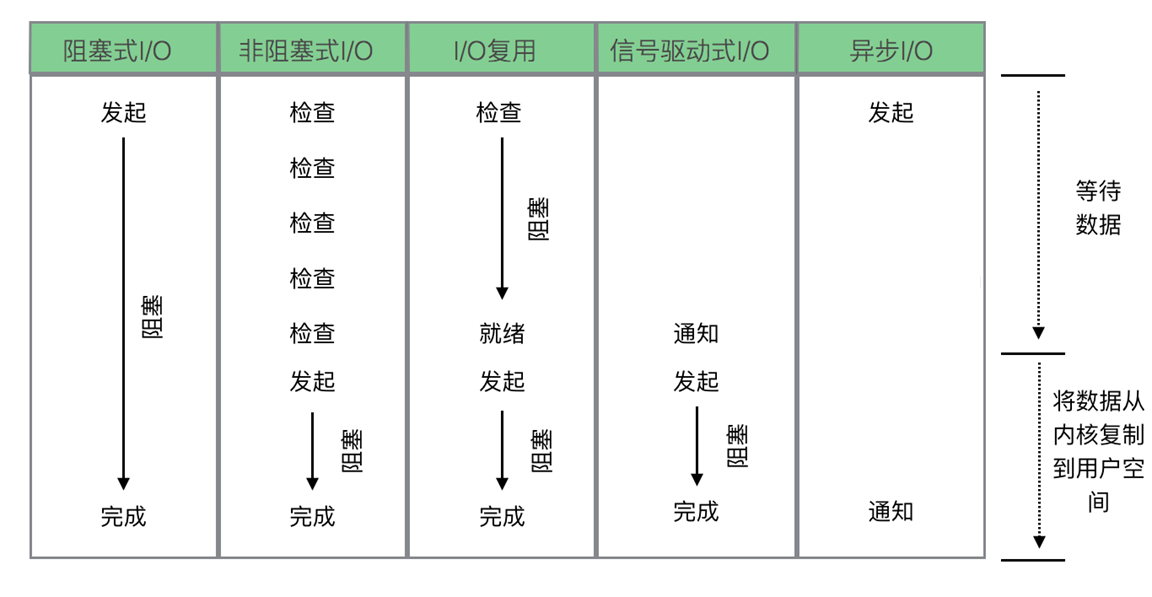


1. 异步IO模型

当应用程序调用aio\_read时，内核一边开始准备数据，另一边将程序控制权返回给应用进程，让应用进程处理其他事情；当内核中有数据报准备就绪时，由内核将数据报拷贝到用户空间，这也是与其他四种模型的最大区别，拷贝完成后返回aio\_read中定义好的函数处理程序。



1. 各IO模型对比



从图中可以看出阻塞程度是：阻塞式IO >非阻塞式IO > IO复用>信号驱动式IO >异步IO。