# 基本概念

Java平台提供了一整套I/O，其抽象程度各有不同。然而，离冰冷的现实越远，要想搞清楚来龙去脉就越难——不管使用哪一种抽象，情况都是如此。JDK1.4的NIO软件包引入了一套新的抽象用于I/O处理。与以往不同的是，新的抽象把重点放在了如何缩短抽象与现实之间的距离上面。NIO抽象与现实中存在的实体有着非常真实直接的交互关系。要想最大限度地满足Java应用程序的密集I/O需求，理解这些新的抽象，以及与其他发生交互作用的I/O服务（其重要性不亚于抽象），正是关键所在。

本章假定您熟知基本的I/O概念，NIO类模拟I/O函数，因此，必须掌握操作系统层面的处理细节，才能理解新的I/O模型。

NIO是java New IO的简称，在JDK1.4里面提供的心api。Sun官方标榜的特性如下：

1、为所有的原始类型提供（Buffer）缓存支持。

2、字符集编码解码解决方案。

3、Channel：一个新的原始I/O抽象。

4、支持锁和内存映射文件的文件访问接口。

5、提供多路（non-bloking）非阻塞式的高伸缩性网络I/O。

**集成的I/O**

在JDK1.4中原来的I/O包和NIO已经很好地集成了。java.io.\*已经以NIO为基础重新实现了，所以现在它可以利用NIO的一些特性。

例如，java.io.\*包中的一些类包含以块的形式读写数据的方法，这使得即使在更面向流的系统中，处理速度也会更快。

也可以用NIO库实现标准I/O功能。例如，可以容易地使用块I/O一次一个字节地移动数据。但是正如您会看到的，NIO还提供了原I/O包中所没有的许多好处。

NIO创建目的是让Java程序员可以实现高速I/O而不需编写自定义的本机代码。NIO将最耗时的I/O操作（即填充和提取缓冲区）转移回操作系统，因而可以极大地提高速度。

流与块的比较

原来的I/O库（在java.io.\*中）与NIO最重要的区别是数据打包和传输方式。正如前面提到的，原来的I/O以流的方式处理数据，而NIO以块的方式处理数据。

面向流的I/O系统一次一个字节的处理数据。一个输入流产生一个字节的数据，一个输出流消费一个字节的数据。为流式数据创建过滤器非常容易。链接几个过滤器，以便每个过滤器只负责单个复杂处理机制的一部分，这样也是相对简单的。不利的一面是，面向流的I/O通常相当慢。

一个面向块的I/O系统以块的形式处理数据。每一个操作都在一步中产生或者消费一个数据块。按块处理数据比按（流式的）字节处理数据要快得多。但是面向I/O缺少一些面向流的I/O所具有的优雅性和简单性。