1. 内存映射文件
2. 概述

大多数操作系统都可以利用虚拟内存实现将一个文件或文件的一部分映射到内存中，然后，这个文件就可以被当做内存数组一样访问，这比传统的文件操作要快很多，从效率来看，内存映射 > 带缓冲的输入流 > 普通输入流 > 随机访问文件。

1. 映射过程

首先，从文件中获得一个通道，通道是用于磁盘文件的一种抽象，它可以使我们访问诸如 内存映射，文件加锁机制 以及 文件间快速数据传递等操作系统特性。

FileChannel channel = FileChannel.open(path, options);

然后，通过调用FileChannel类的map方法从这个通道中获得一个ByteBuffer，可以指定想要映射的文件区域与映射模式，支持的模式有三种：

FileChannel.MapMode.READ\_ONLY : 只读缓冲区

FileChannel.MapMode.READ\_WRITE: 可读写缓冲区，任何修改都会在某个时刻写回到恩建中

FileChannel,MapMode.PRIVATE: 可写缓冲区，但是不会传播到文件中

MappedByteBuffer buffer = channel.map(FileChannel.MapMode.READ\_ONLY, 0, length);

1. 缓冲区
2. 概述

使用内存映射时，我们创建了单一的缓冲区横跨整个文件或者感兴趣的文件区域，我们还可以使用更多的缓冲区来读写大小适度的信息块。缓冲区是由相同类型的数值构成的数组，Buffer类是一个抽象类，它有众多的具体子类，如ByteBuffer, CharBuffer, DoubleBuffer, IntBuffer, LongBuffer 和 ShortBuffer。

1. 缓冲区的常用配置

容量，永远不变，表示缓冲区的最大值；

读写位置， 下一个值将在这里读写；

界限，超过它进行读写没有意义；

可选的标记， 用于重复一个读入或写出操作；

0 < 标记 < 读写位置 < 界限 < 容量

使用缓冲区的目的是执行“写入，然后读取”的循环，即首先写入缓冲区，然后从缓冲区中读取。

1. 常用方法
2. clear()

将位置复位到0，界限设置到容量，为写出做好准备。

1. flip()

界限设置到位置，位置复位到0，为读入做好准备。

1. remaining()

返回剩余可读入或可写出的值的数量，即界限与位置之间的差异。

1. 文件加锁机制
2. 概述

当多个并发的程序需要修改同一个文件时，会有线程安全问题， 因此采用文件锁控制对文件或文件中某个范围的字节访问。

1. 加锁

要锁定一个文件，可以调用FileChannel类的lock或tryLock方法：

FileChannel channel = FileChannel.open(path);

FileLock lock = channel.lock();

FileLock lock = channel.lock();

lock()会阻塞直至可获得锁，tryLock()会立即返回锁或者null。这个文件将会保持锁定状态，直到通道关闭，或者在锁上调用了release()方法释放锁。

还可以只锁定文件的一部分，并制定是否共享锁，如果共享，则允许多个线程从文件中读入，并组织任何线程获得独占的锁。