

## 纸上得来终觉浅 绝知此事要躬行——<https://github.com/sgq0085/learn>



- [博客](#)
- [微博](#)
- [相册](#)
- [收藏](#)
- [留言](#)
- [关于我](#)

### (七) 新I/O

- 博客分类:
- [Java基础-流与文件](#)



Java SE 1.4引入大量用于改进输入/输出处理机制的特性，它们位于java.nio包中，合称"新I/O" 包含特性:字符集编码器和解码器，非阻塞的I/O，内存映射文件和文件加锁机制。

1.内存映射文件  
大多数操作系统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当作是内存数组一样地访问，比传统的文件操作要快的多。  
对于中等尺寸的文件的顺序读入没有必要使用内存映射。  
(1)从文件中获得一个通道(channel)，通道是用于磁盘对象的一种抽象，它使我们可以访问诸如内存映射、文件加锁机制以及文件快速数据传递等操作系统特性。  
通过getChannel方法获得通道，这个方法已经添加到了FileInputStream、FileOutputStream和RandomAccessFile类中

Java代码  



```
1. FileInputStream in = new FileInputStream(file);
2. FileChannel channel = in.getChannel();
```

(2)通过FileChannel类的 map 方法从这个通道中获得一个 MappedByteBuffer 。并可以指定三种文件区域与映射模式  
1)FileChannel.MapMode.READ\_ONLY:所产生的缓冲区是只读的，任何对该缓冲区写入的尝试都会导致ReadOnlyBufferException异常。  
2)FileChannel.MapMode.READ\_WRITE:所产生的缓冲区是可写的，任何修改都会在某时刻写回到文件中。  
注意:其他映射同一个文件的程序可能不能立即看到这些修改，多个程序同时进行文件映射的确切行为是依赖于操作系统的。  
3)FileChannel.MapMode.PRIVATE:所产生的缓冲区是可写的，但是任何修改对这个缓冲区来说都是私有的，不会传播到文件中。  
(3)通过ByteBuffer类和Buffer超类的方法对缓冲区(MappedByteBuffer 直接字节缓冲区)进行数据读写操作。  
1)缓冲区支持顺序和随机数据访问，它有一个可以通过get和put操作来推动的位置。  
2)顺序遍历缓冲区中的所有字节:

Java代码  

```
1. while(buffer.hasRemaining()){
2.     byte b = buffer.get();
3. }
```

3)进行随机访问

Java代码  

```
1. for(int i=0;i<buffer.limit();i++){
2.     byte b = buffer.get(i);
3. }
```



4)读写字符数组 get(byte[] bytes) 和 get(byte[] bytes, int offset, int length)  
5)其他方法 getInt,getLong,getShort,getChar,getFloat,getDouble，这些方法可以用来读入存储为二进制值的基本类型。  
6)Java对二进制数据使用高位在前的排序机制，如果需要处理的文件包含低位在前的二进制数字，使用方法 buffer.order(ByteOrder.LITTLE\_ENDIAN);  
7)查询缓冲区内当前的字节顺序，调用 ByteOrder b = buffer.order();

2.缓冲区数据结构  
缓冲区是由具有相同类型的数值构成的数组，Buffer是一个抽象类，它具有众多的具体子类，其中最长用的是ByteBuffer和CharBuffer,包括ByteBuffer,CharBuffer,DoubleBuffer,IntBuffer,LongBuffer和ShortBuffer。  
注意:StringBuffer类和缓冲区没有关系。

每个缓冲区具有：  
一个容量，它永远不能改变。  
一个读写位置，下一个值将在此进行读写。  
一个界限，超过它进行读写是没有意义的。  
一个可选的标记，用于重复一个读入或写出操作。  
这些值满足：0<=标记<=位置<=界限<=容量

(1)使用缓冲区的主要目的是执行"写，然后读入"循环。假设一个缓冲区，一开始位置为0，界限等于容量。不断调用put将值添加到这个缓冲区中，当耗尽所有的数据或写出的数据量达到容量大小时，就切换到读出操作。  
(2)这时调用flip方法将界限设置到当前位置，并把位置复位到0。现在在remaining方法返回正数时(返回值为“界限-位置”)，不断的调用get。  
(3)在将缓冲区中所有的值都读入之后，调用clear使缓冲区为下一次写循环做好准备。clear方法将位置复位到0，并将界限复位到容量。  
(4)重新读入缓冲区，使用rewind或mark/reset方法。

3.文件加锁机制  
多个同时执行的文件需要修改同一个文件的情况下，这个文件很容易被破坏。  
文件锁可以控制对文件或文件中某个范围的字节的访问，但是文件加锁机制在不同的操作系统之间变化很大  
(1)锁定一个文件可以调用FileChannel类的lock或tryLock方法 FileLock lock = channel.lock(); 或 FileLock lock = channel.tryLock();  
其中lock调用会阻塞直至可获得锁；tryLock将立刻返回，锁不可用的情况下返回null。这个文件将保持锁定状态，直至这个通道关闭，或者在锁上再调用了release方法。  
(2)还可以调用锁定文件的一部分 FileLock lock(long start, long size, boolean exclusive) 或 FileLock trylock(long start, long size, boolean exclusive)  
如果exclusive标志为true，则锁定文件的目的读写，而如果为false，则这是一个共享锁，允许多个进程从文件中读入，并阻止任何进程获得独占的锁。并非所有的操作系统都支持共享锁，可能请求共享锁返回独占锁，调用FileLock类的isShared方法确认持有锁的类型。  
如果锁定了文件的尾部，但文件的长度随后增长，那么增长出来的额外区域是不锁定的，可以使用Long.MAX\_VALUE来锁定所有字节。  
注意：因为文件加锁机制是依赖于操作系统的所以  
(1)在某些系统中，文件加锁仅仅是建议性的，如果一个应用未能得到锁，它仍旧可以被另一个应用并发锁定的文件执行写操作。  
(2)在某些系统中，不能在锁定一个文件的同时将其映射到内存中。  
(3)文件锁是由整个JAVA虚拟机持有的，如果有两个程序是由同一个虚拟机启动的，那么它们不可能每一个都获得一个在同一个文件上的锁。如果对虚拟机一个文件已经持有锁，再调用lock和tryLock时，会抛出OverlappingFileLockException。  
(4)在一些系统中，关闭一个通道会释放由Java虚拟机持有的潜在文件上所有的锁，因此同一个加锁文件应避免使用多个锁。  
(5)在网络化的文件系统上锁定文件是高度依赖于系统的。应避免使用。

分享到： 

[\(八\) 正则表达式](#) | [\(六\) 文件管理](#)

- 2011-12-20 22:00
- 浏览 1173
- [评论\(0\)](#)

- 分类:[编程语言](#)
- [查看更多](#)

评论

发表评论



[您还没有登录,请您登录后再发表评论](#)

相关资源推荐

- [I/O流理解](#)
- [Socket编程： I/O复用](#)
- [Linux信号驱动I/O 学习记录](#)
- [\[NOIp2017 Day1 T2\] 时间复杂度complexity（栈，模拟）](#)
- [硬盘出现“I/O设备错误”的解决方法](#)
- [codesys编程语言源程序](#)
- [你应该知道5个新的编程语言](#)
- [实验七 C++的I/O流（验证性）](#)
- [kafka 问题](#)
- [网络编程中常见的5种I/O模型](#)



sgq0085

- 浏览: 279708 次
- 性别:
- 来自: 吉林→上海
- 我现在离綫

最近访客 [更多访客>>](#)

- ITEye
- [farwind](#)
- ITEye
- [linkyhu](#)
- ITEye
- [nucleus](#)
- ITEye
- [aaron198](#)

文章分类

- [全部博客 \(174\)](#)
- [Java基础-接口与内部类 \(5\)](#)
- [Java基础-流与文件 \(10\)](#)
- [Java基础-JDBC \(12\)](#)
- [Java基础-XML解析 \(7\)](#)
- [Java基础-多线程 \(11\)](#)
- [Java基础-网络 \(6\)](#)
- [Java基础-注解 \(5\)](#)
- [Hibernate 研究记录 \(7\)](#)
- [JavaScript 研究记录 \(6\)](#)
- [ECMAScript 研究记录 \(7\)](#)
- [CSS 研究记录 \(9\)](#)
- [Maven 研究记录 \(8\)](#)
- [SQL 随笔 \(5\)](#)
- [权限控制和单点登陆 \(8\)](#)
- [Hadoop 研究记录 \(6\)](#)
- [随想杂谈 \(33\)](#)
- [JAVA EE \(4\)](#)
- [测试 \(3\)](#)
- [Redis \(10\)](#)
- [Memcached \(2\)](#)
- [MongoDB \(6\)](#)
- [ElasticSearch \(3\)](#)

社区版块

- [我的资讯](#) (0)
- [我的论坛](#) (77)
- [我的问答](#) (25)

存档分类

- [2018-05](#) (3)
- [2018-03](#) (1)
- [2018-02](#) (1)
- [更多存档...](#)

最新评论

- [sgq0085](#): 无尘... 写道楼主，在吗？可以加你qq咨询一下问题吗？公司禁用Q ...  
[Shiro通过Redis管理会话实现集群](#)
- [无尘...](#): 楼主，在吗？可以加你qq咨询一下问题吗？  
[Shiro通过Redis管理会话实现集群](#)
- [zhouminsen](#): 感谢楼主的无私奉献  
[Shiro通过Redis管理会话实现集群](#)
- [tonny1228](#): 经测试还是运行在local  
[远程调用执行Hadoop Map/Reduce](#)
- [asdhobby](#): 楼主，个人感觉每次调用SessionDAO的doUpdate方 ...  
[Shiro通过Redis管理会话实现集群](#)