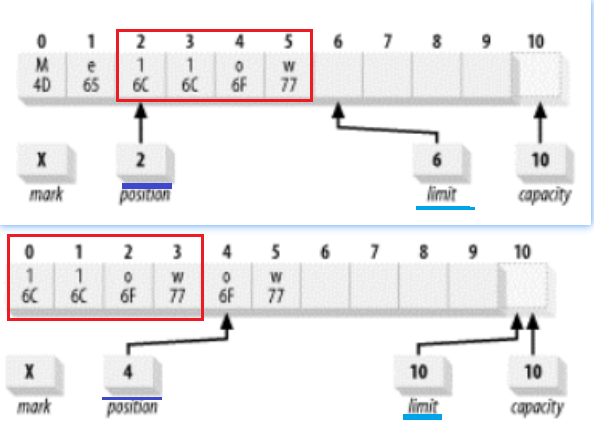
# Buffer-缓冲

## buffer.compact();

有时，您可能只想 从缓冲区中 释放 一部分数据，而不是全部，然后重新填充。



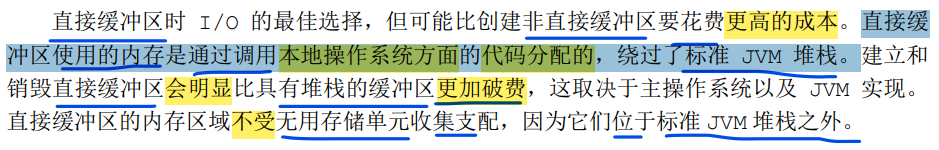
## reset( )和 clear( )

clear( )函数将清空缓冲区，而 reset( ) 位置返回到一个先前设定的标记

## buffer.hasRemaining( )

判断buffer是否有剩余数据

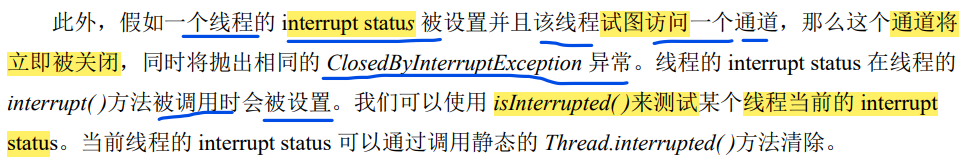
## 直接缓冲区



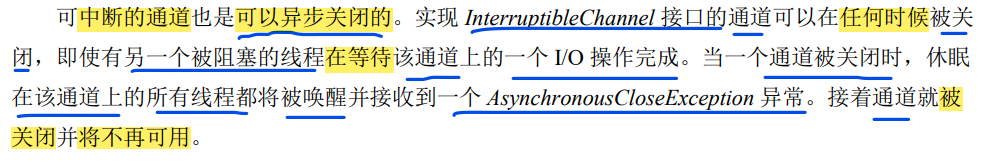
# Channel-通道

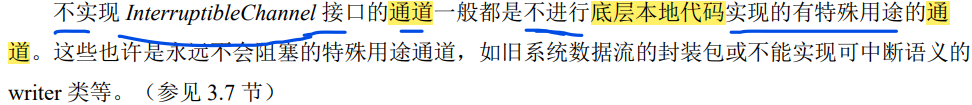
通道 可以以阻塞（blocking）或非阻塞（nonblocking）模式运行。非阻塞模式的通道永远不会 让调用的线程休眠。请求的操作要么立即完成，要么返回一个结果表明未进行任何操作。只有面向 流的（stream-oriented）的通道，如 sockets 和 pipes 才能使用非阻塞模式。

## 线程中断，通道会关闭

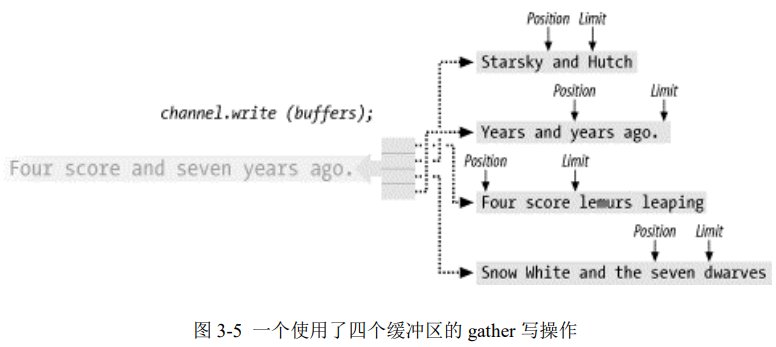


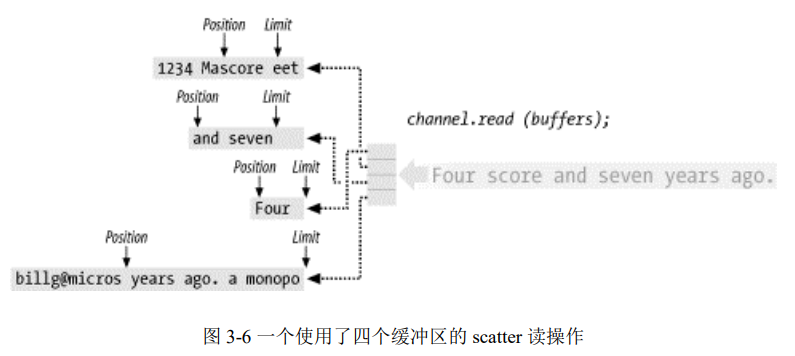
## 异步中断通道





## Gather/scatter





## FileChannel

public abstract long position( )

返回值是一个长整型（long），表示文件中的当前字节位置

public abstract void position (long newPosition)

并将通道的 position 设置为指定值

public abstract void truncate (long size)

砍掉您所指定的新 size 值之外的所有数据

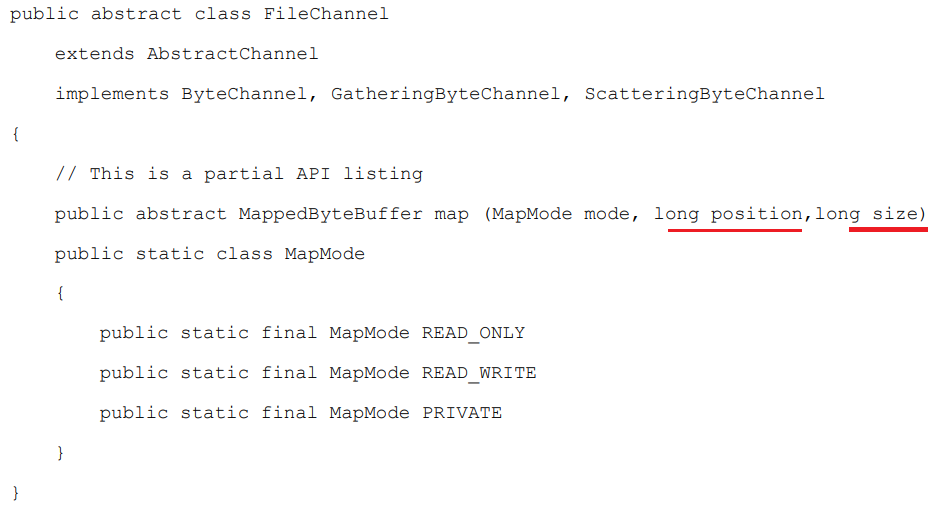
public abstract void force (boolean metaData)

强制将 全部待定的修改 都应用到磁盘的文件上。

参数：metaData的布尔型参数表示:在方法返回值前 文件的元数据（metadata）是否也要被同步更新到磁盘。

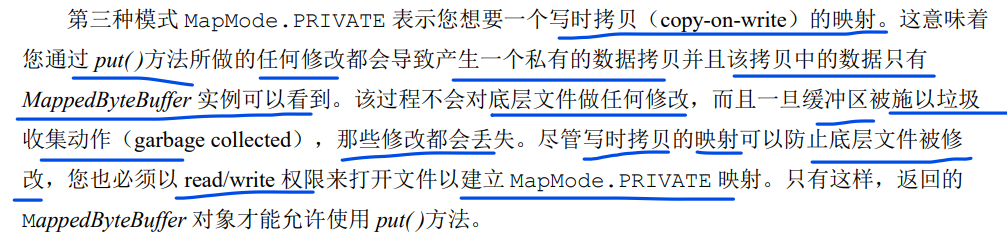
元数据指文件所有者、访问权限、最后一次修改时间等信息。

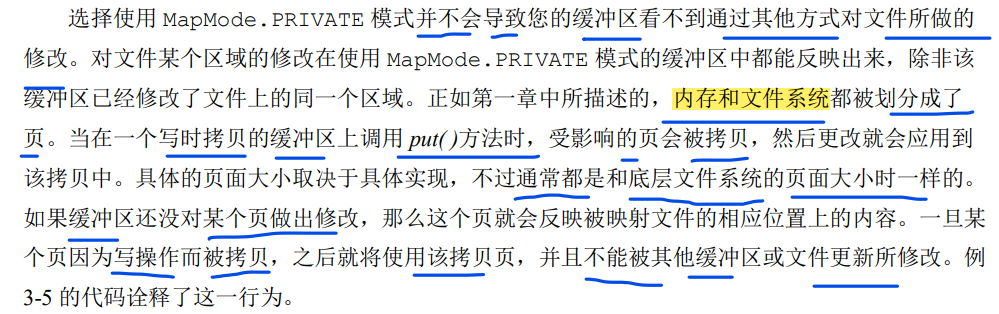
## 内存映射文件

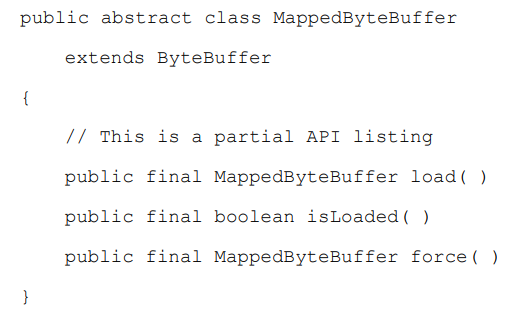


**map( )方法用于创建一个文件映射。**

**MappedByteBuffer buffer = fileChannel.map(FileChannel.MapMode.READ\_ONLY, 0, fileChannel.size());**







**load( )方法**是一个代价高的操作，并不能保证文件就会完全常驻内存，这是由于请求页面调入（demand paging）是动态的。

**isLoaded( )方法**来判断一个被映射的文件是否完全常驻内存了,返回值不保真，只是一个尽力而为。

**force( )方法**会强制将映射缓冲区上的更改应用到永久磁盘存储器上

# Selector-选择器