http://wiki.jikexueyuan.com/project/java-nio-zh/java-nio-filechannel.html

Java NIO中的FileChannel是用于连接文件的通道。通过文件通道可以读、写文件的数据。Java NIO的FileChannel是相对标准Java IO API的可选接口。

FileChannel不可以设置为非阻塞模式，他只能在阻塞模式下运行。

打开文件通道（Opening a FileChannel）

在使用FileChannel前必须打开通道，打开一个文件通道需要通过输入/输出流或者RandomAccessFile，下面是通过RandomAccessFile打开文件通道的案例：

```
RandomAccessFile aFile = new RandomAccessFile("data/nio-data.txt",
"rw");
FileChannel inChannel = aFile.getChannel();
```

从文件通道内读取数据（Reading Data from a FileChannel）

读取文件通道的数据可以通过read方法：

```
ByteBuffer buf = ByteBuffer.allocate(48);
int bytesRead = inChannel.read(buf);
```

首先开辟一个Buffer，从通道中读取的数据会写入Buffer内。接着就可以调用read方法，read的返回值代表有多少字节被写入了Buffer，返回-1则表示已经读取到文件结尾了。

向文件通道写入数据（Writing Data to a FileChannel）

写数据用write方法，入参是Buffer：

```
String newData = "New String to write to file..." +
System.currentTimeMillis();
ByteBuffer buf = ByteBuffer.allocate(48);
buf.clear();
buf.put(newData.getBytes());
buf.flip();
while(buf.hasRemaining()) {
    channel.write(buf);
}
```

注意这里的write调用写在了wihle循环汇总，这是因为write不能保证有多少数据真

实被写入，因此需要循环写入直到没有更多数据。

关闭通道（Closing a FileChannel）

操作完毕后，需要把通道关闭：

```
channel.close();
```

FileChannel Position

当操作FileChannel的时候读和写都是基于特定起始位置的（position），获取当前的位置可以用FileChannel的position()方法，设置当前位置可以用带参数的position(long pos)方法。

```
long pos channel.position();
channel.position(pos +123);
```

假设我们把当前位置设置为文件结尾之后，那么当我们试图从通道中读取数据时就会发现返回值是-1，表示已经到达文件结尾了。 如果把当前位置设置为文件结尾之后，在想通道中写入数据，文件会自动扩展以便写入数据，但是这样会导致文件中出现类似空洞，即文件的一些位置是没有数据的。

FileChannel Size

size()方法可以返回FileChannel对应的文件的文件大小：

```
long fileSize = channel.size();
```

FileChannel Truncate

利用truncate方法可以截取指定长度的文件：

```
channel.truncate(1024);
```

FileChannel Force

force方法会把所有未写磁盘的数据都强制写入磁盘。这是因为在操作系统中出于性能考虑会把数据放入缓冲区，所以不能保证数据在调用write写入文件通道后就及时写到磁盘上了，除非手动调用force方法。 force方法需要一个布尔参数，代表是否把meta data也一并强制写入。

```
channel.force(true);
```