通过NIO读取文件三个步骤：

1.从FileInputStream中读取FileChannel对象。

2.创建Buffer

3.将数据从FileChannel读取到Buffer中

绝对方法和相对方法：

1.相对方法：limit和position的值在操作时会被考虑到。

2.绝对方法：完全忽略掉limit和position的值。

```
/**
 * @ClassName: NioTest4
 * @Description: nio文件通道
 * @author zhangtainyi
 * @date 2019/6/24 8:38
 *
 */
public class NioTest4 {
 public static void main(String[] args) {
  try(
  FileInputStream inputStream = new FileInputStream("src/main/java/com/zhangtianyi/nio/resource/input.txt");
  FileOutputStream outputStream = new FileOutputStream("src/main/java/com/zhangtianyi/nio/resource/output.txt");
  FileChannel inputChannel = inputStream.getChannel();
  FileChannel outputChannel = outputStream.getChannel();
  ){
  ByteBuffer buffer = ByteBuffer.allocate(1024);
  while (true){
  buffer.clear();//必须清空,把position置为0
  int read = inputChannel.read(buffer);
  System.out.println("read:"+read);
  if(-1 == read){
  break;
  }
  buffer.flip();
  outputChannel.write(buffer);
  }
  }catch (Exception e){
  e.printStackTrace();
```

```
29    }
30    }
31  }
```

根据上述代码，流程就是循环读写，先读再写，必须调用**buffer.clear()**方法将limit和position恢复到初始值,不然有以下流程：

buffer的capacity为1024，inputChannel可读为30；

第一次read后：position=30，limit=1024，capacity=1024

此时read=30

第一次调用flip：position=0，limit=30，capacity=1024

第一次write后：position=30，limit=30，capacity=1024

没有调用buffer.clear()

第二次read后：position=30，limit=30，capacity=1024

此时read=0（因为position=limit，position不能大于limit，无法再读取了），不满足read==-1

第二次调用flip：position=0，limit=30，capacity=1024

第二次调用write：position=30，limit=30，capacity=1024

.....后续都和第二次一样