1.MappedByteBuffer

内存映射文件，磁盘文件内容已经被映射到内存中，直接操作内存，修改的数据会同步到磁盘文件中，这个同步操作是操作系统来做的，简单来说直接操作内存（**堆外内存**）就可以操作磁盘文件数据。

```
1  public class NioTest9 {
2   public static void main(String[] args) throws Exception{
3   RandomAccessFile randomAccessFile = new RandomAccessFile("src/main/java/
com/zhangtianyi/nio/resource/NioTest9.txt","rw");
4   FileChannel fileChannel = randomAccessFile.getChannel();
5   //映射模式，起始位置，映射长度
6   MappedByteBuffer mappedByteBuffer =
fileChannel.map(FileChannel.MapMode.READ_WRITE, 0, 5);
7   mappedByteBuffer.put(0, (byte)'a');
8   mappedByteBuffer.put(3, (byte)'b');
9   randomAccessFile.close();
10   }
11  }
```

2.文件锁(共享锁和排他锁)

```
1  public class NioTest10 {
2   public static void main(String[] args) throws Exception {
3   RandomAccessFile randomAccessFile = new RandomAccessFile("src/main/java/
com/zhangtianyi/nio/resource/NioTest9.txt","rw");
4   FileChannel fileChannel = randomAccessFile.getChannel();
5   //起始位置，锁长度，true表示共享锁，flase表示排他锁
6   FileLock fileLock = fileChannel.lock(3, 6, true);
7   System.out.println("valid:"+fileLock.isValid());
8   System.out.println("lock type:"+fileLock.isShared());
9
10   fileLock.release();//释放锁
11   fileChannel.close();
12   }
13  }
```

3.关于Buffer的Scattering和Gathering

Scattering总是把按顺序把channel的字节数据读到多个buffer，读满一个buffer在读到下一个buffer

Gathering总是按顺序把多个buffer的字节数据写入到channel，写完一个buffer在写下一个buffer

服务端

```java
public class NioTest11 {
 public static void main(String[] args) throws Exception {
 ServerSocketChannel serverSocketChannel = ServerSocketChannel.open();
 InetSocketAddress address = new InetSocketAddress(8899);
 serverSocketChannel.socket().bind(address);

 int messageLength = 2 + 3 + 4;
 ByteBuffer[] buffers = new ByteBuffer[3];
 buffers[0] = ByteBuffer.allocate(2);
 buffers[1] = ByteBuffer.allocate(3);
 buffers[2] = ByteBuffer.allocate(4);
 SocketChannel socketChannel = serverSocketChannel.accept();

 while (true){

 int bytesRead = 0;
 while (bytesRead < messageLength){
 long r = socketChannel.read(buffers);
 bytesRead += r;

 System.out.println("bytesRead:"+bytesRead);

 Arrays.asList(buffers).stream()
 .map(buffer -> "position:" + buffer.position() + ",limit:" + buffer.lim
it())
 .forEach(System.out::println);
 }

 Arrays.asList(buffers).forEach(buffer -> {
 buffer.flip();
 });

 long bytesWritten = 0;
 while (bytesWritten < messageLength){
 long r = socketChannel.write(buffers);
 bytesWritten += r;
 }

 Arrays.asList(buffers).forEach(buffer -> {
 buffer.clear();
 });
```

```
41   System.out.println("bytesRead:" + bytesRead + ", bytesWritten:" + bytes
Written + ", messageLength:" + messageLength);
42   }
43   }
44  }
```

客户端

```
1  public class NioTest11_Client {
2   public static void main(String[] args) throws Exception {
3
4   SocketChannel socketChannel = SocketChannel.open();
5   socketChannel.connect(new InetSocketAddress("localhost", 8899));
6   System.out.println("连接是否已经建立: "+socketChannel.isConnected());
7
8   ByteBuffer buffer = ByteBuffer.allocate(9);
9   buffer.put("welcomeXX".getBytes(Charset.defaultCharset()));
10  // System.out.println("position:"+buffer.position()+",limit"+buffer.limi
t());
11   buffer.flip();
12  // System.out.println("position:"+buffer.position()+",limit"+buffer.limi
t());
13   socketChannel.write(buffer);
14  // System.out.println("position:"+buffer.position()+",limit"+buffer.limi
t());
15   buffer.flip();
16  // System.out.println("position:"+buffer.position()+",limit"+buffer.limi
t());
17   socketChannel.read(buffer);
18  // System.out.println("position:"+buffer.position()+",limit"+buffer.limi
t());
19   buffer.flip();
20   byte[] bytes = new byte[9];
21   buffer.get(bytes);
22   System.out.println(new String(bytes, Charset.defaultCharset()));
23
24   }
25  }
```