

# BIO NIO AIO Netty

## 1. 什么 IO

- 中 以 为 入 出 , 串 化 串 化  
入 出 写 入 出 单 和 交互
- 在 中, 分 内 , 因为 及 : 准 入 出, 件  
作, 上 传 , 串 ,

IO

- 从 务 器 上 下 图 , 从 上 以 到 中, 在 到 中

## 2. 了 不 IO之前先了 : 与 , 与 区别

- 同 , 个 任 务 之 前 不 做 其 他 作, ( 于 在 )
- , 个 任 务 之 前, 可 以 其 他 作 ( 于 在 )
- , 于 , 前 , 不 做 其 他 作 只
- , 前 , 可 以 去 其 他 作

## 3. 什么 BIO

- 多线程：同一个线程，服务器一个线程一个线程，即服务器启动一个线程，之前不做其他工作（单线程情况下，线程件？），可以控制善于且固定，服务器，发生于中，以前唯一，但单

#### 4. 什么 NIO

- 多线程：同一个线程，服务器一个线程一个线程，即服务器启动一个线程，之前不做其他工作（单线程情况下，线程件？），可以控制善于且固定，服务器，发生于中，以前唯一，但单

#### 5. 什么 AIO

- 多线程：同一个线程，服务器一个线程一个线程，即服务器启动一个线程，之前不做其他工作（单线程情况下，线程件？），可以控制善于且固定，服务器，发生于中，以前唯一，但单
- 多线程：同一个线程，服务器一个线程一个线程，即服务器启动一个线程，之前不做其他工作（单线程情况下，线程件？），可以控制善于且固定，服务器，发生于中，以前唯一，但单

#### 6. 什么Netty

- 多线程：同一个线程，服务器一个线程一个线程，即服务器启动一个线程，之前不做其他工作（单线程情况下，线程件？），可以控制善于且固定，服务器，发生于中，以前唯一，但单
- 多线程：同一个线程，服务器一个线程一个线程，即服务器启动一个线程，之前不做其他工作（单线程情况下，线程件？），可以控制善于且固定，服务器，发生于中，以前唯一，但单



Netty

[https://blog.csdn.net/weixin\\_43122090](https://blog.csdn.net/weixin_43122090)

## 7. BIO NIO AIO 区别

- 向 , 只 单向 写, 向 冲 可以双向 写
- 使 做 , 于单向 写, , 会 前 , , 为 响其 , 为 个 , , 不 , 上下 切 , 从 , 因 使 , 使 个 听 , 使 其他
- 以 发 作 作 可以去做其他 作, 作 内 作 (不 可以 下 )

## 8. IO 分



### 写 单位 分:

- : 以 为单位, 入 出 位 其只 取 代 为 char
- : 以 为单位, 入 出 位 可以 任何 代 只 为 byte

### IO 作 分:

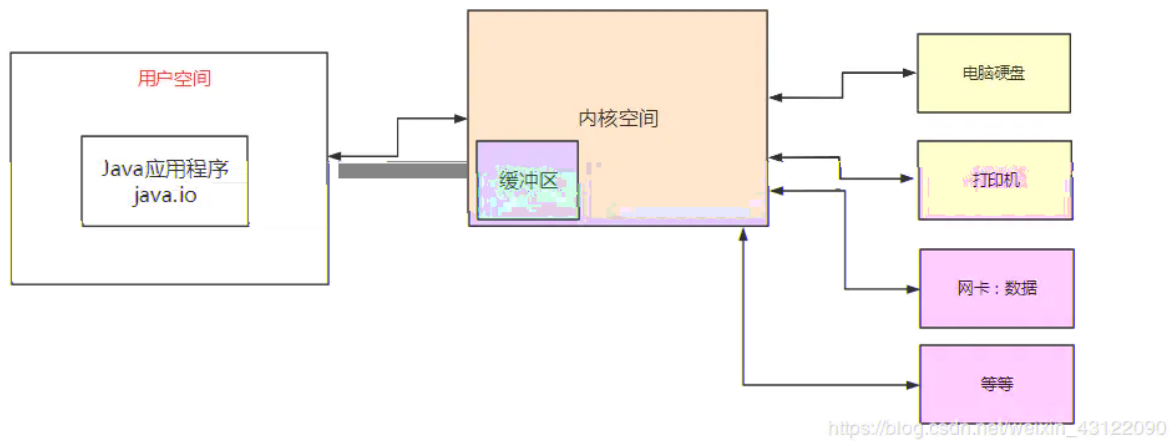
- 出 : 从内 出到 件 只 写 作
- 入 : 从 件 入到内 只 作
- : 出 可以 助 们创 件, 入 不会

### 写 与 , 内 分:

- : 与 , 入 出
- : 也叫包 , 个 于 在 写 加个 冲区 ( 个 区, 于 件和 中 )
- : 为什么 ? 主 作 在 入 写出 , , 以减 , 以 便下 写 件, 了

## 9. 什么 内

- 们 不 , 们 , 但 作 ( ) 会 们 分 内 , 他叫内 , 和 们 区别



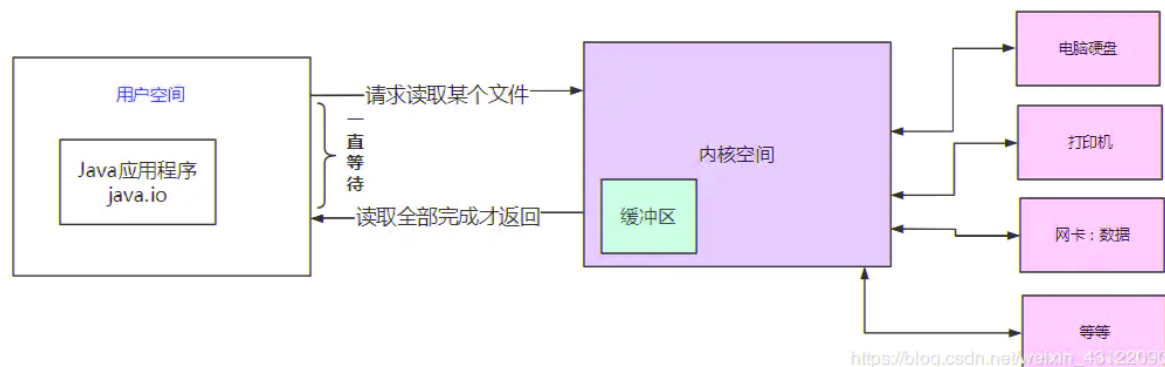
## 10. 五 IO

- :

### 1. BIO (blocking I/O)

- 在 , 且 在 前 , 在 候不做其他 事 , 十分专 只
- 上 , 动作, 上
- 在内 准 之前, 会 ,

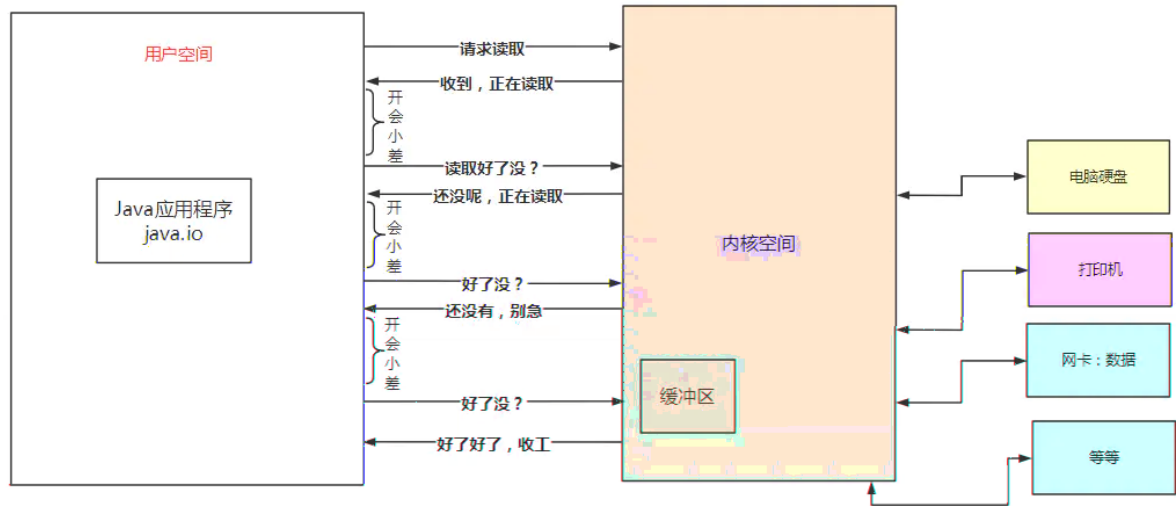
以线程为单位：期间正个过程是不能去操作其他任何事情



### 2. NIO (nonblocking I/O)

- 也在 , 但 不 在 上, 在 上 个 中, 也在 做其他 事 ( 会 书, 会 , 会又去 其他人 ), 但 在做 些事 候, 个固 否上 到 上 , 停下 中 事 , 上
- B , 个

以线程为单位：期间整个过程中是可以去做别的操作



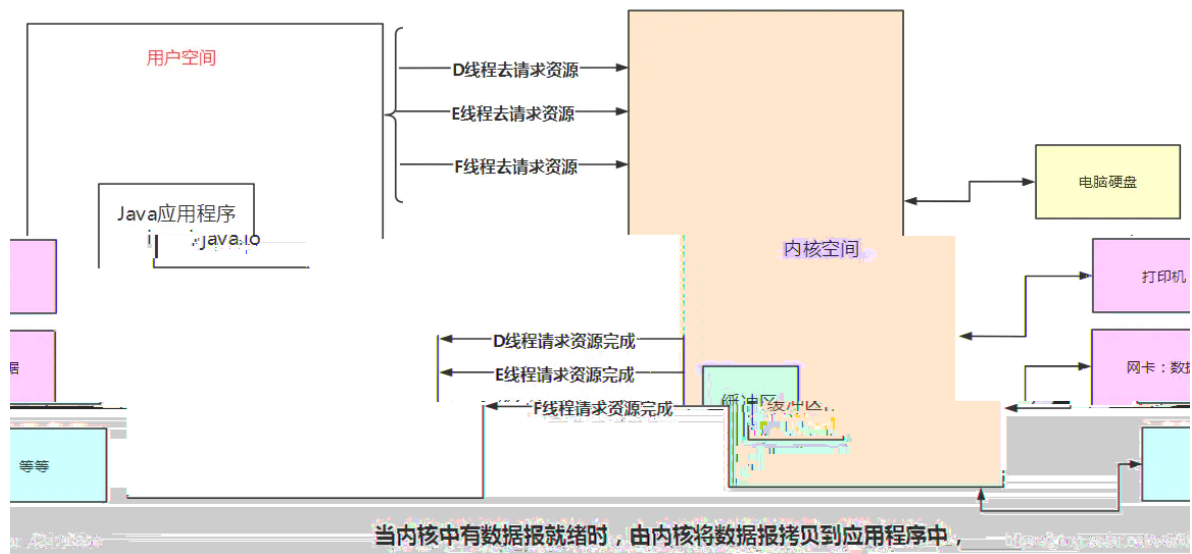
每次检查是否处理好就是在轮训的过程，这是BIO的操作

[https://blog.csdn.net/weixin\\_43122090](https://blog.csdn.net/weixin_43122090)

### 3. AIO (asynchronous I/O)

- 也，但事，于他了，他们他上，上，  
，会上去

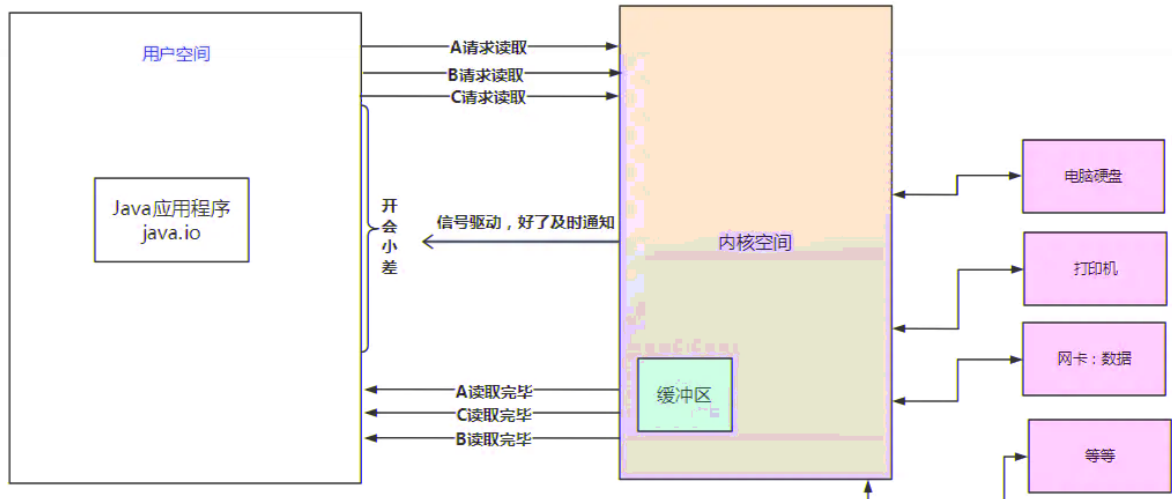
以线程为单位：期间整个过程中是可以去做别的操作



### 4. 信 动IO (signal blocking I/O)

- 也在，但与不同，他上个，上  
候，个会响，会上

以线程为单位：期间整个过程中是可以去做别的操作



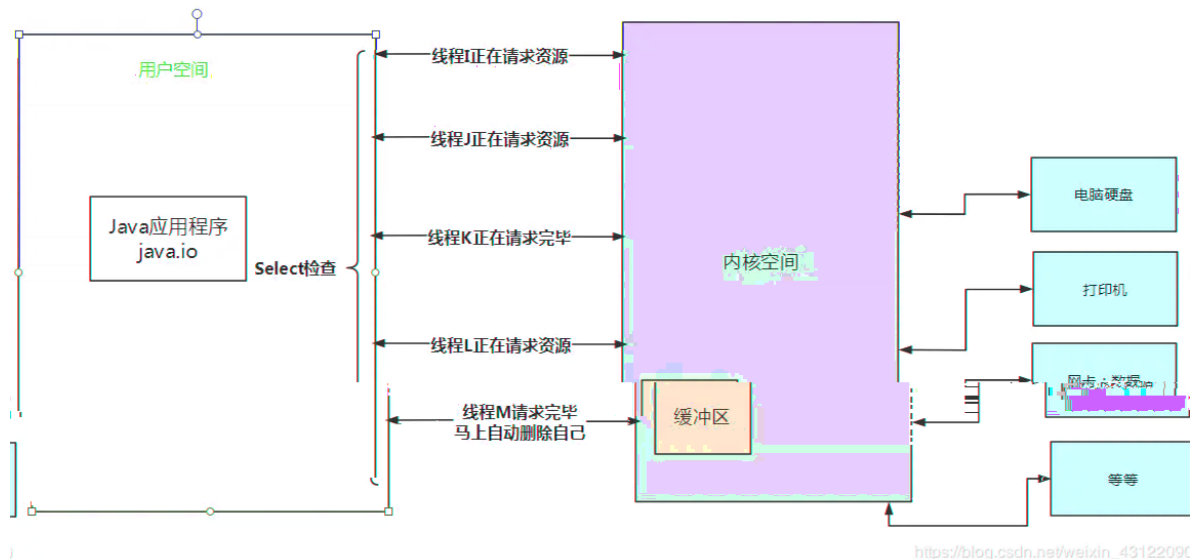
信号驱动IO模型，应用进程告诉内核：当数据报准备好的时候，给我发送一个信号，对SIGIO信号进行捕捉，并且调用我的信号处理函数来获取数据报。  
[https://blog.csdn.net/weixin\\_43122090](https://blog.csdn.net/weixin_43122090)

IO	SIGIO
----	-------

## 5.IO (I/O multiplexing)

- 同 也在 个 否 上 加了 ， 减 了 在 ， 不

以线程为单位：期间整个过程中是可以去做别的操作



IO	select	select
----	--------	--------

- 于 ， 但可以 个 件 听， 以

## 11. 什么 (Bit),什么 (Byte),什么 (Char), 们 , 什 么区别

- 二 制单位， 作 分取值

- 中 储 单元, 个 位 二 制 , ( 内 , 个 可 个 , 两个 可 个 ) 128 127
- 可 写 单位, 他只 义上 个 号 , 中, ¥ 在 位 取值 0 65535
- 单位 他只
- 个
- 到 东 个 二个

## 12. 什么 列化, 什么 列化, 列化 做 些 作

- 列化, 以二 制 保 在 上
- 反 列化; 二 制 件 化为 取
- 口, 不 在 上 加

## 13. 列化 候 个serialVersionUID , 做什么, 什么

- 个 口会 个
- 但 义 个 因为 于 , 反 列化 可 会 个
- ( 先 列化, 后在反 列化之前修 了 , 么 会 因为修 了 , 也变化了, 列化和反 列化 其 不匹 , 反 列化 功

## 14. 么 SerialVersionUID

- 可 列化 可以 名为 ( (static) (final) long ) 其
- 两 ( 你 个 了 口, 义 , 会 供 个 功 告 你去 义 在 中 击 中 图 下, 会 动 两

## 15. BufferedReader 于 , 主 做什么 , 些 典

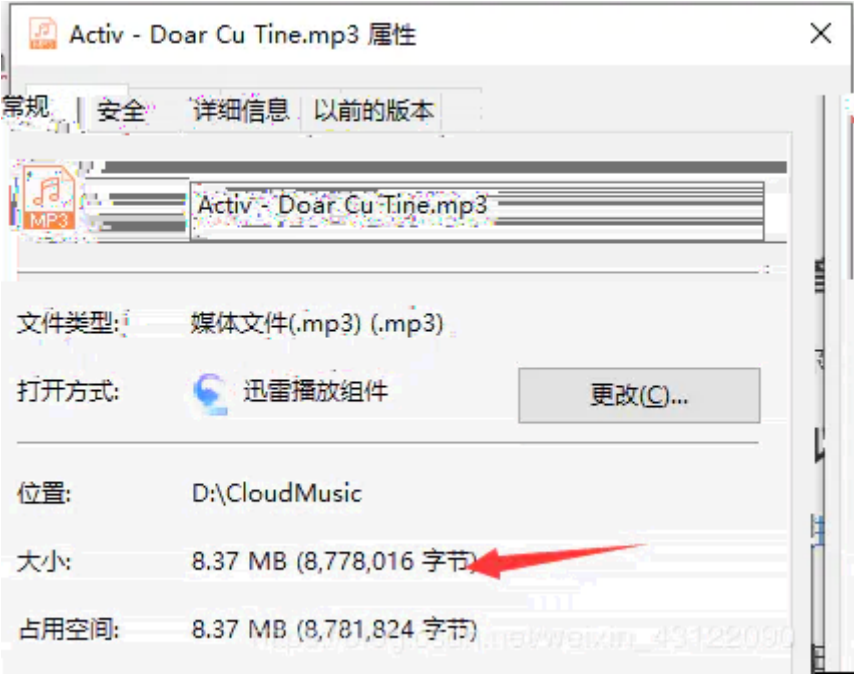
- 于 中 冲 , 可以 取 内 在内 , ()

## 16. Java中 主 些?

- 代 ( )
- 
- 
- 
- 
- 

## 17. 为什么 乐 件

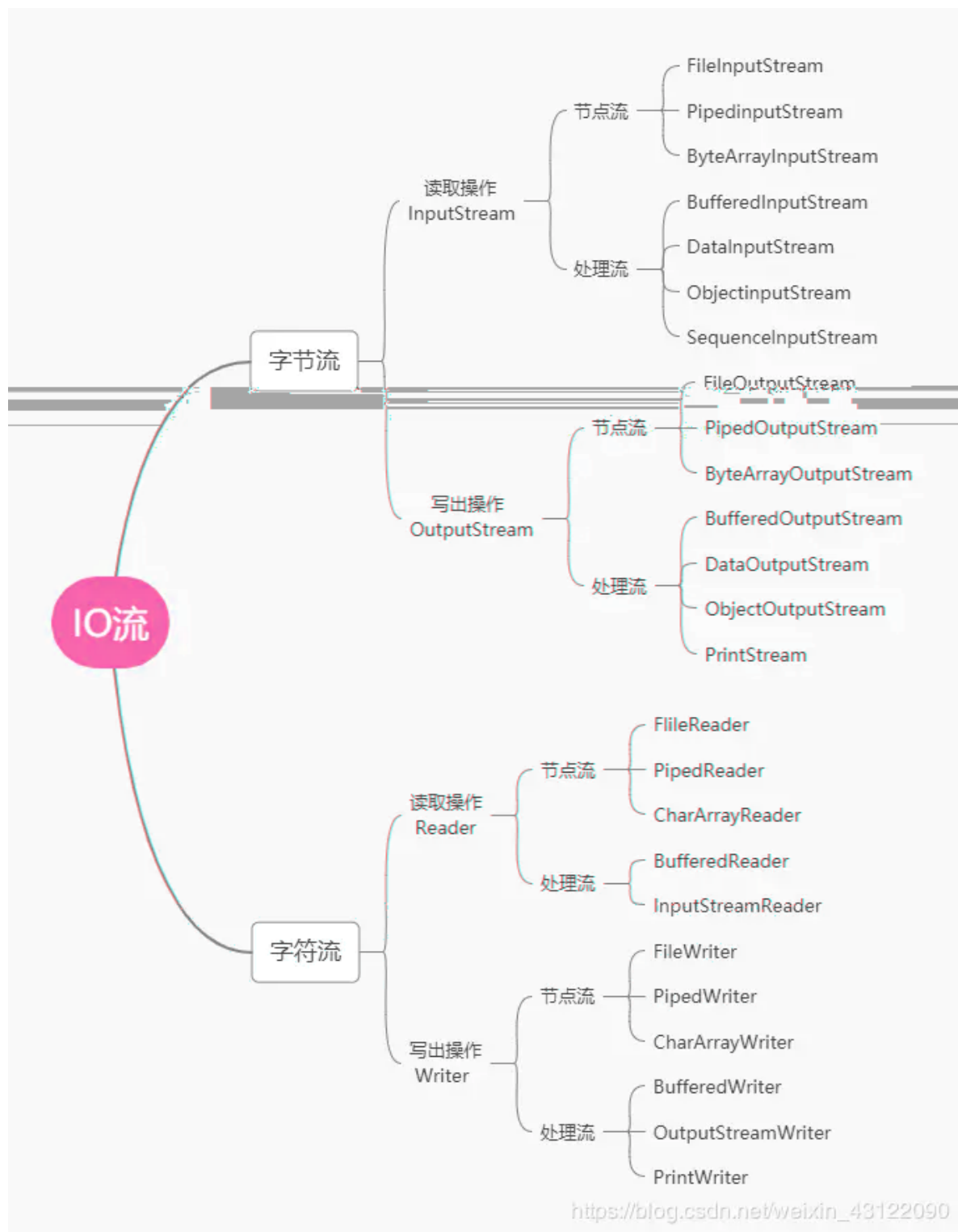
- 个，你你件了，了储件



18. IO，以及何使

前了么，在们入主，后，从件作到后IO作会  
例：





## 19. IO 作

- IO IO

### 1 字符 件

#### 1.1

- 我们取件单位，么（）取件合：
- ：

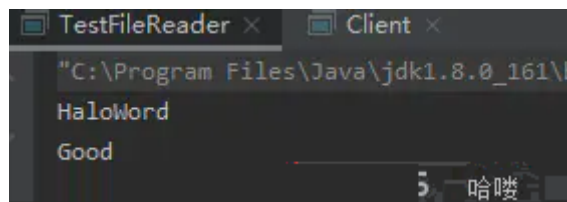
- `FileReader`：以 为 单位， 入 出 位 其只 取 代 为 `char`
- `FileWriter`：以 为 单位， 入 出 位 可以 任何 ， 图 件 乐 代 只 为 `byte`
- **FileReader**：（ 入 ）：

```
package com.test.io;

import java.io.FileReader;
import java.io.IOException;

public class TestFileReader {
    public static void main(String[] args) throws IOException {
        int num=0;
        // char
        char[] buf=new char[1024];
        //
        FileReader fr = new FileReader("D:\\test.txt");//
        //FileReader.read() buf , -1
        //FileReader.read() java char int
        // A java 97
        char
        while((num=fr.read(buf))!=-1) { }
        //
        for(int i=0;i<buf.length;i++) {
            System.out.print(buf[i]);
        }
    }
}
```

- :



电脑 > 本地磁盘 (D:)

名称	HaloWord
	Good
test.txt	哈喽

## 1.2

- , 但 了 们 不同 作 了 个 , 加 冲功 , 免 写 只 单 , 其 作
- **BufferedReader**： 入 使 , 加 冲功 , 免 写

```
package com.test.io;

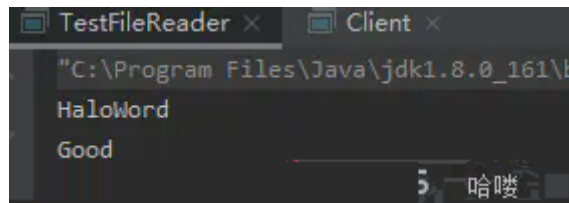
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
```

```

public class TestBufferedReader {
    public static void main(String[] args) throws IOException {
        int num=0;
        //          String
        String[] bufstring=new String[1024];
        //
        FileReader fr = new FileReader("D:\\test.txt");//
        //
        BufferedReader br = new BufferedReader(fr);
        //
        String line=null;
        //BufferedReader.readLine()          null
        while((line=br.readLine())!=null) {
            bufstring[num]=line;
            num++;
        }
        br.close();//
        for(int i=0;i<num;i++) {
            system.out.println(bufstring[i]);
        }
    }
}

```

•



## 2 字符 写出 件

### 2.1 写出

- 写出 , 使 ( ) 写出 件 合 : 出内 加到 件
- **FileWriter** : ( 出 ), 写出 件不 在会 动创 个 件 使  
写出 件 原件, 在 件 加内 不 , 参  
加 参 : 例了

```

package com.test.io;

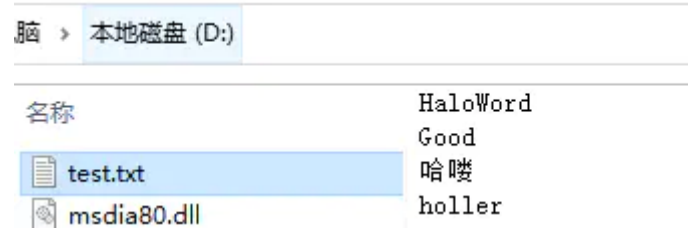
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

public class TestFileWriter {
    public static void main(String[] args) throws IOException {
        //File
        File file = new File("D:\\test.txt");//
        //
        //new FileWriter(file,true)      true
        FileWriter out=new FileWriter(file,true);
        //          ,\n
        String str="\nholler";
    }
}

```

```
//
out.write(str);
out.close();
}
}
```

- :



## 2.2 写出

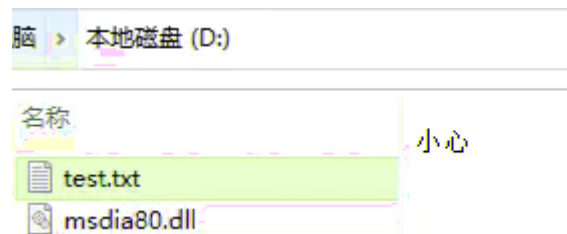
- **BufferedWriter** : 加 冲功 , 免 写 : , 只 了他 件位 , 加 代 件

```
package com.test.io;

import java.io.*;

public class TestBufferedWriter {
    public static void main(String[] args) throws IOException {
        //File
        File file = new File("D:\\test.txt");//
        //
        //new FileWriter(file) true
        writer writer = new FileWriter(file);
        ////
        BufferedWriter bw = new BufferedWriter(writer);
        bw.write("\n ");
        bw.close();
        writer.close();
    }
}
```

- :



## 3 字节 写入写出 件

### 3.1 写入写出 件

- 们 取 图 件 乐 , 使 取写出 :

- `FileInputStream`: 以 为 单位, 入 出 位 其 只 取 代 为 `char`
- `FileOutputStream`: 以 为 单位, 入 出 位 可 以 任 何 代 只 为 `byte`
- `FileInputStream`: ( 入 )
- `FileOutputStream`: ( 出 )





```
package com.test.io;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;



public class TestFileOutputStream {
    public static void main(String[] args) throws IOException {
        //
        FileInputStream fis = new FileInputStream("D:\\Akie - Lemon Cover.mp3");
        //
        FileOutputStream fos = new FileOutputStream("D:\\copy.mp3");

        //
        byte[] arr = new byte[fis.available()];
        //
        fis.read(arr);
        //
        fos.write(arr);
        fis.close();
        fos.close();
    }
}
```

- 之前:

	Akie秋绘 - Lemon (Cover: 米津玄師) .m...	2020/2/27 12:18	媒体
	msdia80.dll	2006/12/1 23:37	应用
	llq	2020/4/2 13:02	文件
	boke	2020/4/2 0:08	文件

\*\* \*\*

电脑 > 本地磁盘 (D:) >		
名称	修改日期	类型
	copy.mp3	2020/4/2 17:27
	Akie秋绘 - Lemon (Cover: 米津玄師) .m...	2020/2/27 12:18

### 3.2 写入写出 件

- `FileInputStream`: ( 入 )
- `FileOutputStream`: ( 出 )
- `BufferedInputStream`: ( 缓冲区 入 )

- **BufferedOutputStream**: ( 缓冲区 入 ) 缓冲区 , 缓冲区 作 主 : 免 和 交 ,

```
package com.test.io;

import java.io.*;

public class TestBufferedOutputStream {
    //      ,      .mp3
    public static void main(String[] args) throws IOException {
        FileInputStream fis = new FileInputStream("D:\\copy.mp3");
        //      fis
        BufferedInputStream bis = new BufferedInputStream(fis);
        //      ,      copy.mp3
        FileOutputStream fos = new FileOutputStream("D:\\copy2.mp3");
        //      fos
        BufferedOutputStream bos = new BufferedOutputStream(fos);
        //
        int i;
        while((i = bis.read()) != -1) {
            bos.write(i);
        }
        bis.close();
        bos.close();
    }
}
```

- 之前:

名称	修改日期	类型
copy.mp3	2020/4/2 17:27	媒体
Akie秋绘 - Lemon (Cover: 米津玄師) .m...	2020/2/27 12:18	媒体

之 :

名称	修改日期	类型
copy2.mp3	2020/4/2 17:33	媒体
copy.mp3	2020/4/2 17:27	媒体
Akie秋绘 - Lemon (Cover: 米津玄師) .m...	2020/2/27 12:18	媒体

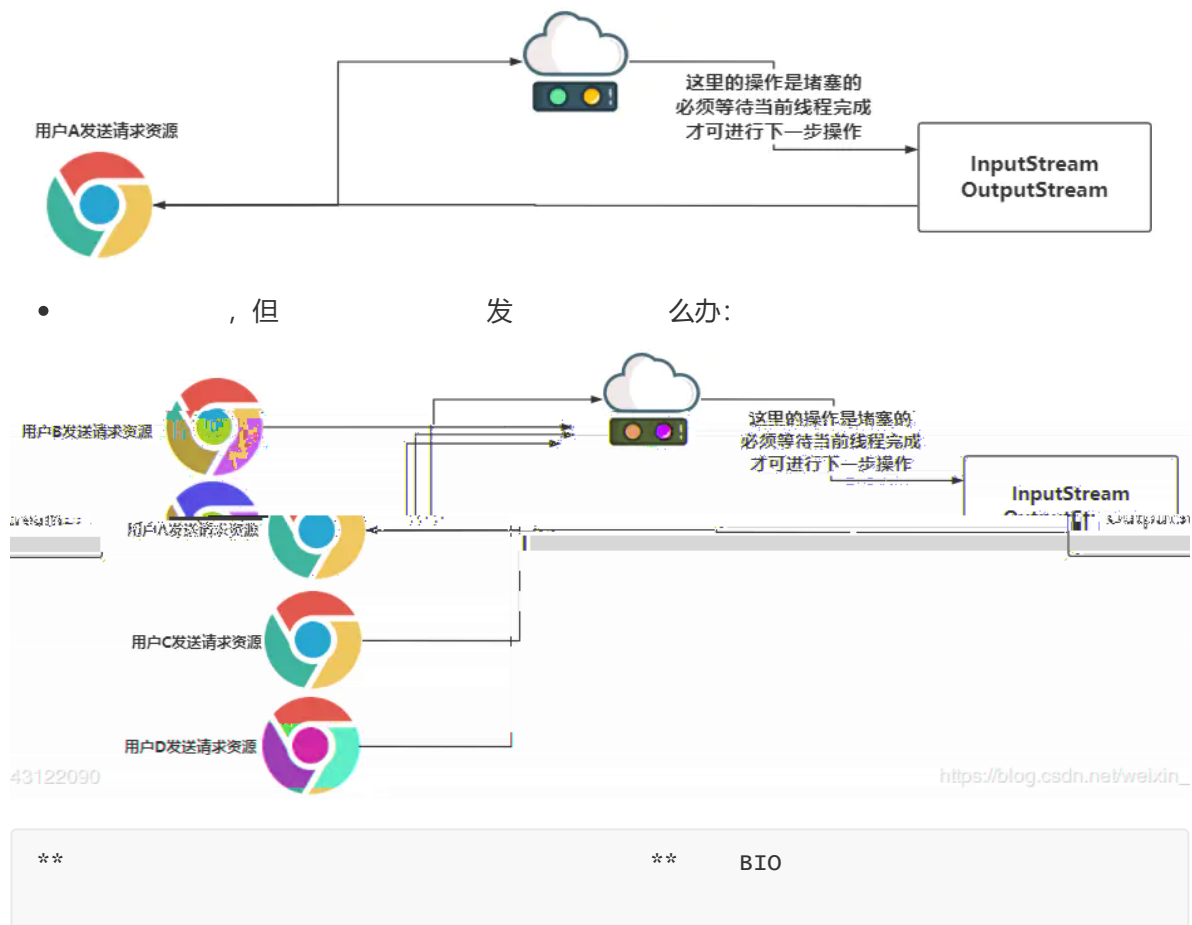
## 20. 作IO

- 使 单 会
- 不 可以 之前 , 个东 , 于 信,

## 21. 作IO 历

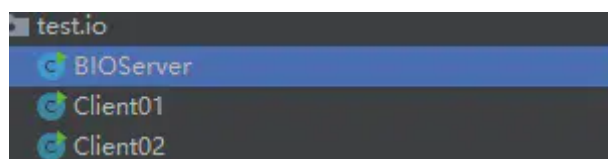
### 1 BIO 会出 什么 ？

- 
- 例：（ ） 在 ， 且 在 前 ， 在 候不  
做其他 事 ， 十分专 只 上 ， 动作， 上

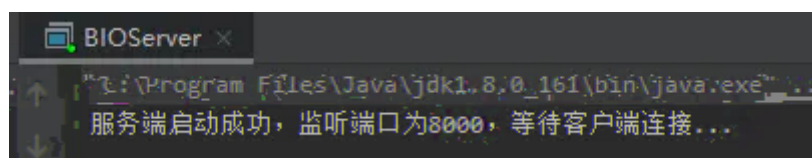


### BIO代 例：（ ）

- 三个 ， 启动 务 ， 后启动 ， 作 候启动 二个



### 启动 务 （ ）



### 启动 个 ， 发 务器 功

```
BIOServer x Client01 x
"C:\Program Files\Java\jdk1.8.0_161\bin\java.exe" ...
TCP连接成功
请输入:
```

.

```
BIOServer x Client01 x
"C:\Program Files\Java\jdk1.8.0_161\bin\java.exe" ...
服务端启动成功，监听端口为8000，等待客户端连接...
客户连接成功，客户信息为: /127.0.0.1:52861
```

启动二个，功 (在 中)

Client ( )

```
BIOServer x Client01 x Client02 x
"C:\Program Files\Java\jdk1.8.0_161\bin\java.exe" ...
服务端启动成功，监听端口为8000，等待客户端连接...
客户连接成功，客户信息为: /127.0.0.1:52861
```

```
BIOServer x Client01 x Client02 x
"C:\Program Files\Java\jdk1.8.0_161\bin\java.exe" ...
TCP连接成功
请输入:
```

个制台入，入后会关个，在务发二个上

了

```
BIOServer x Client01 x Client02 x
"C:\Program Files\Java\jdk1.8.0_161\bin\java.exe" ...
TCP连接成功
请输入:
666666
TCP协议的Socket发送成功
```

.

```
BIOServer x Client01 x Client02 x
"C:\Program Files\Java\jdk1.8.0_161\bin\java.exe" ...
服务端启动成功，监听端口为8000，等待客户端连接...
客户连接成功，客户信息为: /127.0.0.1:52861
666666
客户连接成功，客户信息为: /127.0.0.1:52905
```

BIO 信代 :

- 协 使 信: 务 (先 )

.



```

package com.test.io;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.ServerSocket;
import java.net.Socket;

//TCP Socket BIO
public class BIOServer {
    // main
    public static void main(String[] args) {
        // Socket
        ServerSocket server = null;
        Socket socket = null;
        //
        InputStream in = null;
        OutputStream out = null;
        try {
            server = new ServerSocket(8000);
            System.out.println("            8000            ...");
            while (true){
                socket = server.accept(); //
                System.out.println("            " +
socket.getRemoteSocketAddress());
                in = socket.getInputStream();
                byte[] buffer = new byte[1024];
                int len = 0;
                //
                while ((len = in.read(buffer)) > 0) {
                    System.out.println(new String(buffer, 0, len));
                }
                //
                out = socket.getOutputStream();
                out.write("hello!".getBytes());
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

- 协 使 信: ( 二 )

```

package com.test.io;

import java.io.IOException;
import java.io.OutputStream;
import java.net.Socket;
import java.util.Scanner;

//TCP Socket BIO
public class Client01 {
    public static void main(String[] args) throws IOException {
        //            socket ip port
        Socket socket = new Socket("127.0.0.1", 8000);
    }
}

```

```

        //      socket
    //
    OutputStream outputStream = socket.getOutputStream();
        //      IO
    System.out.println("TCP      \n      ");
        String str = new Scanner(System.in).nextLine();
        byte[] car = str.getBytes();
        outputStream.write(car);
        System.out.println("TCP      Socket      ");
        //
    outputStream.flush();
        //
    socket.close();
    }
}

```

- 协 使 信: ( 三 )

```

package com.test.io;

import java.io.IOException;
import java.io.OutputStream;
import java.net.Socket;
import java.util.Scanner;

//TCP      Socket
public class Client02 {
    public static void main(String[] args) throws IOException {
        //      socket      ip port
        Socket socket = new Socket("127.0.0.1", 8000);
        //      socket
    //
    OutputStream outputStream = socket.getOutputStream();
        //      IO
    System.out.println("TCP      \n      ");
        String str = new Scanner(System.in).nextLine();
        byte[] car = str.getBytes();
        outputStream.write(car);
        System.out.println("TCP      Socket      ");
        //
    outputStream.flush();
        //
    socket.close();
    }
}

```

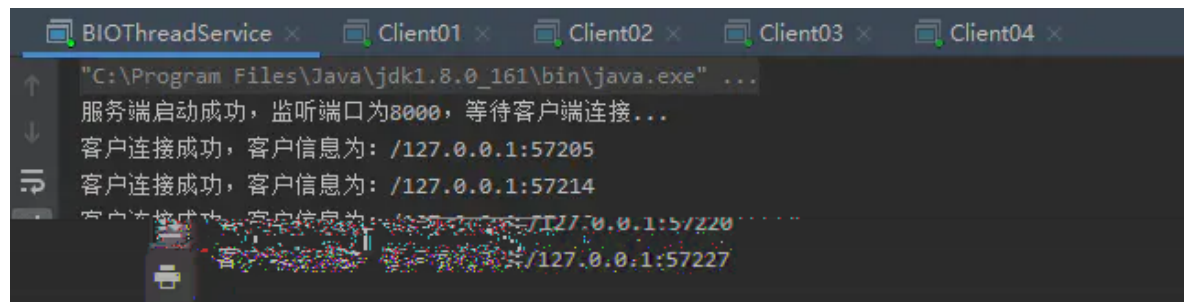
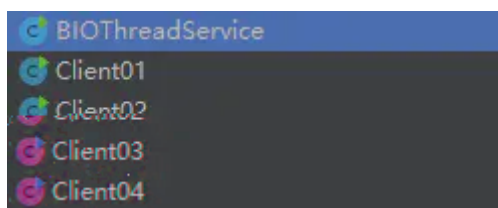
## 2 决BIO 会出

人 会 , 不 决了 ?

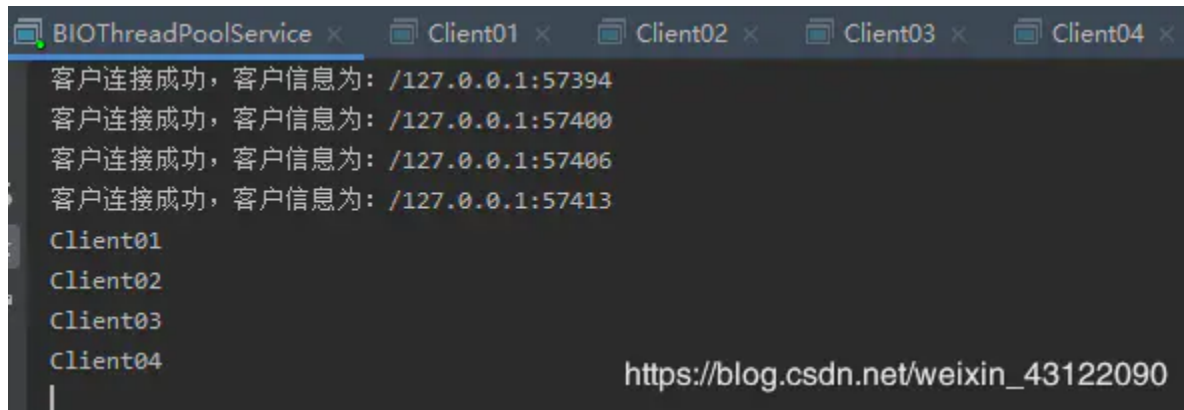
- 使 可以 决 , 因为他可以充分发
- , 不 , 上下 切 , 从



- 四个 , 制了 个



```
`client01 client02
```



```
BIOThreadPoolService x Client01 x Client02 x Client03 x Client04 x
客户连接成功, 客户信息为: /127.0.0.1:57394
客户连接成功, 客户信息为: /127.0.0.1:57400
客户连接成功, 客户信息为: /127.0.0.1:57406
客户连接成功, 客户信息为: /127.0.0.1:57413
Client01
Client02
Client03
Client04
https://blog.csdn.net/weixin_43122090
```

**BIO 信代 :**

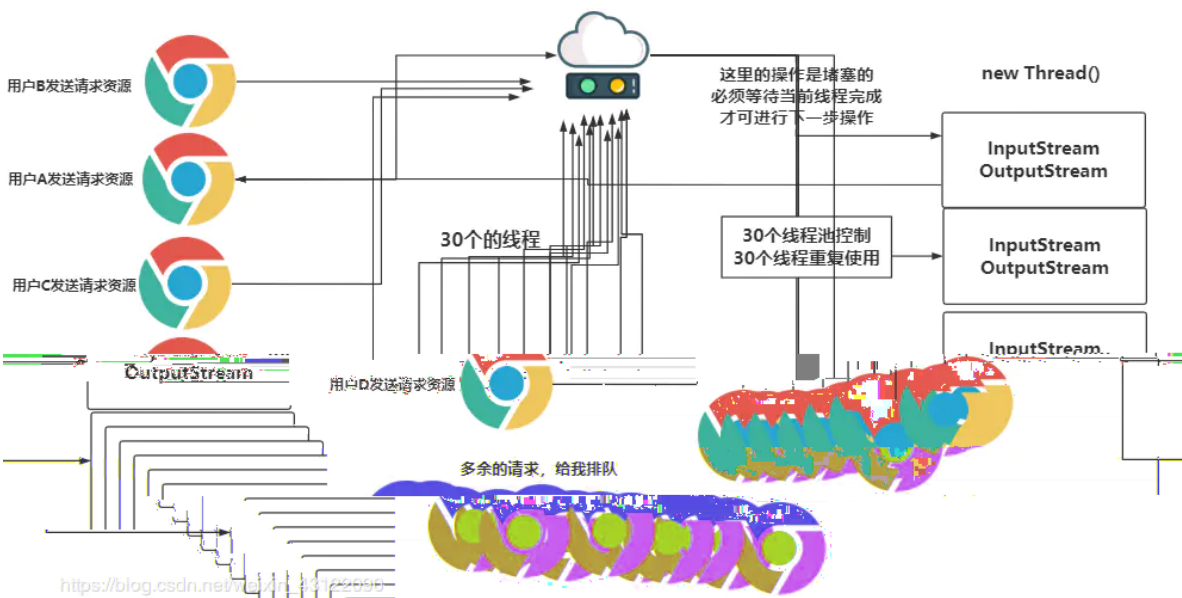
- 

```
package com.test.io;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.ServerSocket;
import java.net.Socket;

//TCP Socket BIO
public class BIOThreadService {
    public static void main(String[] args) {
        try {
            ServerSocket server = new ServerSocket(8000);
            System.out.println("8000 ... ");
            while (true) {
                Socket socket = server.accept();//
                System.out.println(" " +
socket.getRemoteSocketAddress());
                // IO
                //
                Thread thread = new Thread(new Runnable() {
                    public void run() {
                        try {
                            InputStream in = socket.getInputStream();
                            byte[] buffer = new byte[1024];
                            int len = 0;
                            //
                            while ((len = in.read(buffer)) > 0) {
                                System.out.println(new String(buffer, 0, len));
                            }
                            //
                            OutputStream out = socket.getOutputStream();
                            out.write("hello".getBytes());
                        } catch (IOException e) {
                            e.printStackTrace();
                        }
                    }
                });
                thread.start();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

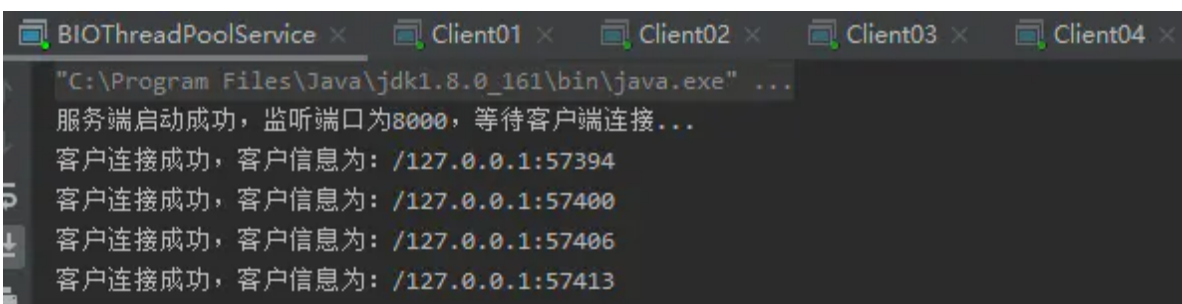
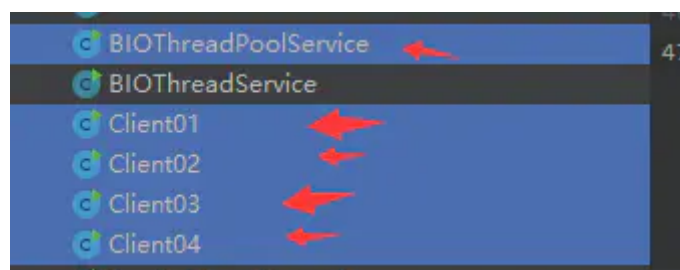
人 会 , TM ?



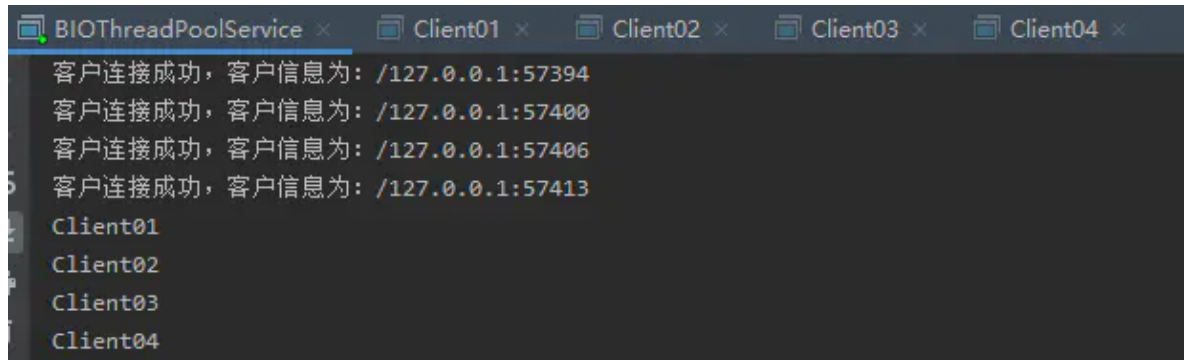
- 固 可以 决 个 , 万 不 们  
作, 上 了 去创 去 , , 做 去了,  
也不 呀
- : 坐不住了, 兄 交 , 你 决: NIO

**BIO代 例:**

- 四个



Client01 Client02

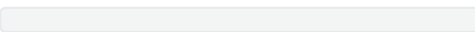


The screenshot shows a Java IDE with a terminal window and several tabs. The terminal window displays the following output:

```
客户连接成功, 客户信息为: /127.0.0.1:57394
客户连接成功, 客户信息为: /127.0.0.1:57400
客户连接成功, 客户信息为: /127.0.0.1:57406
客户连接成功, 客户信息为: /127.0.0.1:57413
Client01
Client02
Client03
Client04
```

The tabs at the top are: BIOThreadPoolService, Client01, Client02, Client03, and Client04.

BIO 信代 :

- 

```
package com.test.io;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

//TCP Socket BIO
public class BIOThreadPoolService {
    public static void main(String[] args) {
        //
        ExecutorService executorService = Executors.newFixedThreadPool(30);
        try {
            ServerSocket server = new ServerSocket(8000);
            System.out.println("8000 ...");
            while (true) {
                Socket socket = server.accept();
                System.out.println(" " +
                    socket.getRemoteSocketAddress());
                //
                executorService.execute(new Thread(new Runnable() {
                    public void run() {
                        try {
                            InputStream in = socket.getInputStream();
                            byte[] buffer = new byte[1024];
                            int len = 0;
                            //
                            while ((len = in.read(buffer)) > 0) {
                                System.out.println(new String(buffer, 0, len));
                            }
                            //
                            OutputStream out = socket.getOutputStream();
                            out.write("hello".getBytes());
                        } catch (IOException e) {
                            e.printStackTrace();
                        }
                    }
                }));
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

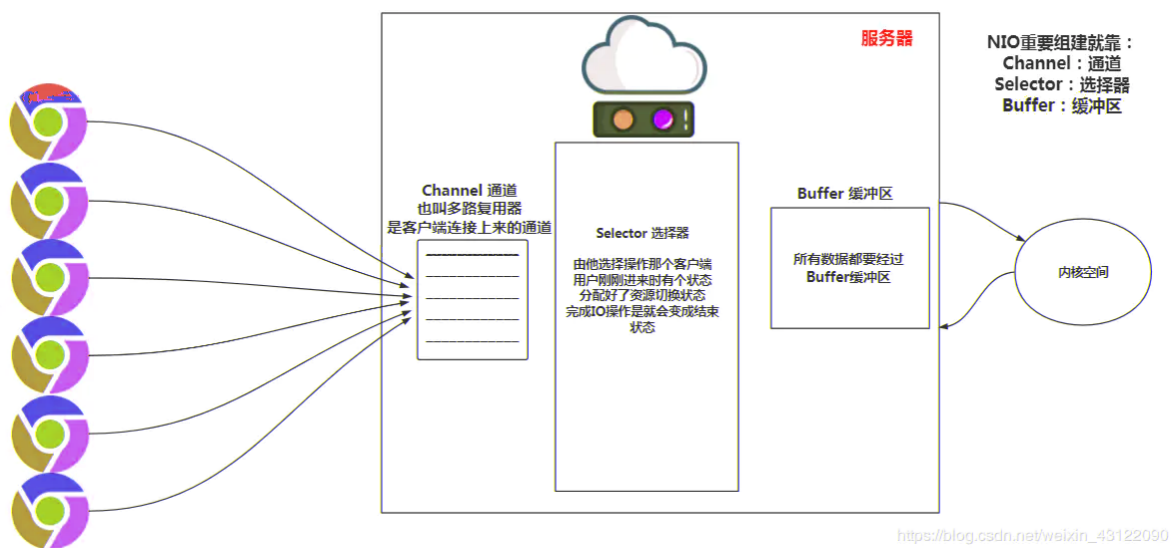
```

    }
    })
    );
}
} catch (IOException e) {
    e.printStackTrace();
}
}
}

```

#### 4 使用 NIO 通信

- 提供操作，他，变，服务器，一个，一个，启动一个，器到，于，且，（作），服务器，发于中，之后



[https://blog.csdn.net/welxin\\_43122090](https://blog.csdn.net/welxin_43122090)

#### 什么是 Channel (Channel)

- 一个，可以取和写入，们，写入包含一个，一个，缓冲区，后再，区，写入到，中，从，入，缓冲区，再从，缓冲区，取
- 似于原，中，（），但，区别：
  - 单向，双向，可，可写
  - 写，可以，写

#### 什么是 Selector (Selector)

- 可以，他为，合，了之后，们会，册到，中，且，们，他，个，在，判，否发，变化，作，后在，出

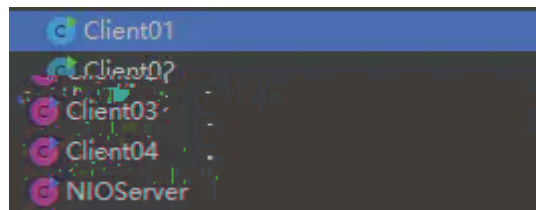
#### 什么是 Buffer (缓冲区)

- 一个，冲，包含，些，写入，刚，出
- 在，向，中，写入，到，中，了，（，冲区）后，到，（冲区）中

- 缓冲区上每个  个  ,  
 内部几个变量,可以在同一块缓冲区上反复写(不  
 再写)

代 例:

- 



- 例,先 务,在 制台 入 了:



- NIO

```
package com.test.io;

import com.lijie.iob.RequestHandler;

import java.io.IOException;
import java.net.InetSocketAddress;
import java.nio.ByteBuffer;
import java.nio.channels.SelectionKey;
import java.nio.channels.Selector;
import java.nio.channels.ServerSocketChannel;
import java.nio.channels.SocketChannel;
import java.util.Iterator;
import java.util.Set;

public class NIOServer {
    public static void main(String[] args) throws IOException {
        //111111111
        //Service Channel
        ServerSocketChannel serverChannel = ServerSocketChannel.open();
        //
        serverChannel.configureBlocking(false);
        //nio api
        serverChannel.bind(new InetSocketAddress(8000));
        // Channel
        System.out.println("      8000      ..."+
serverChannel.getLocalAddress());
```



```

//22222222
// selector
Selector selector = Selector.open();
// selector Channel
// Selector , Accepted
serverChannel.register(selector, SelectionKey.OP_ACCEPT);

//33333333
// buffer 1024
ByteBuffer buffer = ByteBuffer.allocate(1024);
RequestHandler requestHandler = new RequestHandler();

//4444444444
//
// Channel
while (true) {
    int select = selector.select();
    if (select == 0) {
        continue;
    }
    // Set Channel
    Set<SelectionKey> selectionKeys = selector.selectedKeys();
    Iterator<SelectionKey> iterator = selectionKeys.iterator();
    while (iterator.hasNext()) {
        // SelectionKey Channel
        SelectionKey key = iterator.next();
        // SelectionKey Channel OP_ACCEPT
        if (key.isAcceptable()) {
            // SelectionKey Channel
            ServerSocketChannel channel = (ServerSocketChannel)
key.channel();
            // Channel
            SocketChannel clientChannel = channel.accept();
            // Channel
            System.out.println(" " + clientChannel.getRemoteAddress());
            // Channel
            clientChannel.configureBlocking(false);
            // SelectionKey Channel OP_READ
            clientChannel.register(selector, SelectionKey.OP_READ);
        }
        // read
        if (key.isReadable()) {
            // buffer
            SocketChannel channel = (SocketChannel) key.channel();
            // buffer
            channel.read(buffer);
            //
            String request = new String(buffer.array()).trim();
            buffer.clear();
            // buffer
            System.out.println(String.format(" %s : %s",
channel.getRemoteAddress(), request));
            // buffer
            String response = requestHandler.handle(request);
            //
            channel.write(ByteBuffer.wrap(response.getBytes()));
        }
    }
}

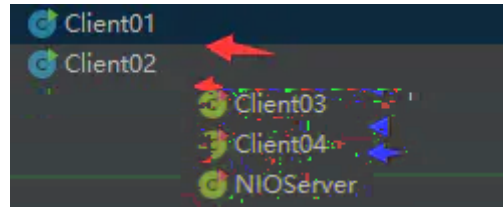
```

```

        iterator.remove();
    }
}
}
}

```

- 例: ( ) 制台 入 了



```

package com.test.io;

import java.io.IOException;
import java.io.OutputStream;
import java.net.Socket;
import java.util.Scanner;

//TCP Socket
public class Client01 {
    public static void main(String[] args) throws IOException {
        // socket ip port
        Socket socket = new Socket("127.0.0.1", 8000);
        // socket
        OutputStream outputStream = socket.getOutputStream();
        // IO
        System.out.println("TCP \n");
        while(true){
            byte[] car = new Scanner(System.in).nextLine().getBytes();
            outputStream.write(car);
            System.out.println("TCP Socket");
            //
            outputStream.flush();
        }
    }
}

```

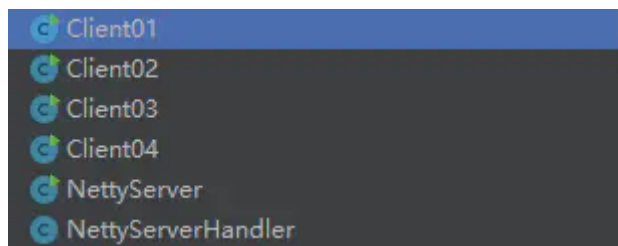
## 5 使 Netty 信

- 供 个 供 事件 动 和
- 具, 以 发 可 务器和
- 个 于 务器, 使 可以 保你 和 单 发出
- 个, 例 了 协, 务 化和 化了
- 发, 例, 和 务 发



Netty    NIO       NIO       NIO       Netty       NIO

- 原    , 他    于    个    ,    且优化了    , 使 他    便, 单
- 上代    :



- 先 加依    :

```
<dependency>
  <groupId>io.netty</groupId>
  <artifactId>netty-all</artifactId>
  <version>4.1.16.Final</version>
</dependency>
```

- ,    代    么    ,
- 

```
package com.lijie.iob;

import io.netty.bootstrap.ServerBootstrap;
import io.netty.channel.*;
import io.netty.channel.nio.NioEventLoopGroup;
import io.netty.channel.socket.SocketChannel;
import io.netty.channel.socket.nio.NioServerSocketChannel;
import io.netty.handler.codec.serialization.ClassResolvers;
import io.netty.handler.codec.serialization.ObjectEncoder;
import io.netty.handler.codec.string.StringDecoder;

public class NettyServer {
    public static void main(String[] args) throws InterruptedException {
        EventLoopGroup bossGroup = new NioEventLoopGroup();
```

```

        EventLoopGroup workerGroup = new NioEventLoopGroup();
        try {
            ServerBootstrap b = new ServerBootstrap();
            b.group(bossGroup, workerGroup)
              .channel(NioServerSocketChannel.class)
              .childHandler(new ChannelInitializer<SocketChannel>() {
                  @Override
                  protected void initChannel(SocketChannel socketChannel)
                      throws Exception {
                      ChannelPipeline pipeline = socketChannel.pipeline();
                      pipeline.addLast(new StringDecoder());
                      pipeline.addLast("encoder", new ObjectEncoder());
                      pipeline.addLast("decoder", new
io.netty.handler.codec.serialization.ObjectDecoder(Integer.MAX_VALUE,
ClassResolvers.cacheDisabled(null)));

                      //
                      //      IO      Handle
                      //      SpringMVC Controller
                      pipeline.addLast(new NettyServerHandler());

                      }
                  })
              .option(ChannelOption.SO_BACKLOG, 128)
              .childOption(ChannelOption.SO_KEEPALIVE, true);
            ChannelFuture f = b.bind(8000).sync();
            System.out.println("      : " + 8000);
            f.channel().closeFuture().sync();
        } finally {
            workerGroup.shutdownGracefully();
            bossGroup.shutdownGracefully();
        }
    }
}

```

- 做 作, 了

```

package com.lijie.iob;

import io.netty.channel.Channel;
import io.netty.channel.ChannelHandlerContext;
import io.netty.channel.ChannelInboundHandlerAdapter;

public class NettyServerHandler extends ChannelInboundHandlerAdapter {
    RequestHandler requestHandler = new RequestHandler();

    @Override
    public void handlerAdded(ChannelHandlerContext ctx) throws Exception {
        Channel channel = ctx.channel();
        System.out.println(String.format("      %s",
channel.remoteAddress()));
    }

    @Override
    public void channelRead(ChannelHandlerContext ctx, Object msg) throws
Exception {

```

```

        Channel channel = ctx.channel();
        String request = (String) msg;
        System.out.println(String.format("          %s : %s",
channel.remoteAddress(), request));
        String response = requestHandler.handle(request);
        ctx.write(response);
        ctx.flush();
    }
}

```

- 代 之前 代 , 在 制下 下吧

```

package com.test.io;

import java.io.IOException;
import java.io.OutputStream;
import java.net.Socket;
import java.util.Scanner;

//TCP Socket
public class Client01 {
    public static void main(String[] args) throws IOException {
        // socket ip port
        Socket socket = new Socket("127.0.0.1", 8000);
        // socket
        OutputStream outputStream = socket.getOutputStream();
        // IO
        System.out.println("TCP \n ");
        while(true){
            byte[] car = new Scanner(System.in).nextLine().getBytes();
            outputStream.write(car);
            System.out.println("TCP Socket ");
            //
            outputStream.flush();
        }
    }
}

```

- , 之前 , 启动 务 , 在启动 制台 入 了: