理解socket的阻塞

从socket中得到一个输入流InputStrean,然后从这个流中取数据,如果这个时候流里面的没有数据,可能是服务还没有发数据过来或发过来的数据也取完了,那么线程就会停在那里..直到服务器在发数据过来,从Socket的InputStream中取到了数据,这个线程才会向下走.所以当要和服务器交互通信的时候,就用一个while一真从Socket的流中数据取,流中的数据取完了就停在那里.等服务器在次发数据过来.---这就是阻塞的

这个时候就有一个问题了,当我从Socket中的流中取数据时,如果服务器一直不发数据过来,也就是从Socket中得到的InputStream流中一直没有数据,那么这个程序就一直停在从流中取数据的那一行.

如果过了很长时间服务器还没有数据发过来,就不等了,想让程序继续执行以后的代码.这时只一个办法,把服务器端关掉.客户端检测到这个Socket连接已断开,就不会停在从流中取数据的代码,而是执行下面的代码.

显然以把服务器关掉的方式来让Socket连接断开有点不合适,所以我想到的办法就是用多线程.让线程A执行主代码,线程B去从流中读取数据,如果服务器又迟迟不发数据过来,又不关掉这个连接,那么线程B就一真让它阻塞在那里.

上面的所有,用一包话来概括:javaIO流是阻塞流,(JavaNIO(java new io非阻塞流,NIO这里没有用到).

这里在说说在ServerSocket是中的accept()方法也是一个阻塞方法,也就是如果没有客户端来连接,那么就线程就一直停在那里.直到有一个客户端口连接了,才开始执行后面的代码.

从上面一点,可以看出如果要创建多客端的服务器,就要多次调用accept()方法,同要用多线程来做,线程A来一直循环执行accept().当有socket连接来过的时候就,创建一个新的线程来处理专门处理这个连接.

到这里还没有完,因为前面提到过,Socket连接里面的输入输出流也是阻塞的,如果这个Socket连好过后,客户端和服务器迟迟不发数据,那么这个新创建的线程也会阻塞在那里,(除非这个新创建的线程里面不操作Socket里面的IO),为了阻止这种情况,就要像上面说的,在新创建的线程里面在新创建一个线程处理Socket里面的流的问题.

下面的程序的测试客户端口建议用Sockettool的一个小软件

```
public class SocketServer {

    public static void main(String[] args) throws IOException {
        ServerSocket serversocket = new ServerSocket(10088);
        Socket socket = serversocket.accept();// 同样这也是一个阻塞方法
        //当第一个客户端连接进来过后就会执行这句话,
        System.out.println("accept方法通过了....");
        //这个程序如果没有第二个客户端连进,那以下的代码不会执行到的.
        Socket socket2 = serversocket.accept();
        System.out.println("accept2方法通过了....");
        InputStream instream = socket.getInputStream();
        BufferedReader reader = new BufferedReader(new InputStreamReader(instream));
        String line="";
        while (true){
            line=reader.readLine();
            if(line==null){
                System.out.println("line 为NULL 了");
                break;
            }
            System.out.println(line);
        }
        /**
         * 程序有一种情况可以执行到这里,就是socket客户端口断开.不是socket2断开
         * 第一个连接断开后line=reader.readLine();就不会在阻塞了.流中的数据读
         * 完过后跳就跳出了while向下执行了.
         */
        System.out.println("程序结束了");
```

```
        // 关闭资源
        socket.close();
        serversocket.close();
    }
}
```

所在要做网络编程就要到多线程.也这是多线程的一个理解:在多用户和多网络的的情况下,不可能为每一个用户,和网络请求都创建一个进程所以就有了线程.