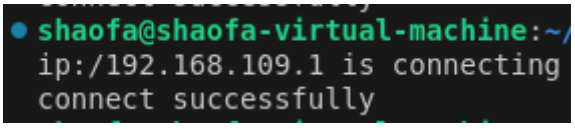
建立连接

1.把虚拟机当作服务器，把实体机当作客户端。首先在虚拟机上监听8080端口

```
//在服务端写的代码
while(true){
            try{
                 ServerSocket serverSocket = new ServerSocket(8080);
                 Socket socket = serverSocket.accept();//如果在这个端口上监听到客户端连接，那么就返回一个socket对象
                  System.out.println("ip:"+socket.getInetAddress()+" is connecting");
                 System.out.println("connect successfully");
                 break;
             }catch(IOException e){
                //如果出现端口被占用，则会抛出异常，需要捕获
                 System.err.println("disconnect with server");
                 //e.printStackTrace();
             }
        }
```




```
//在客户端写的代码
```

```
try {
    Socket socket = new Socket("192.168.109.131",8080);
    //先运行服务端的程序开始监听，然后再运行客户端的程序，如果客户端的socket对象创建成功，则说明两者的连接建立，
    //在服务端会显示connect successfully

    System.out.println("连接成功");
} catch (IOException e) {
    throw new RuntimeException(e);
}
```

发送和接收消息

1.服务端的代码

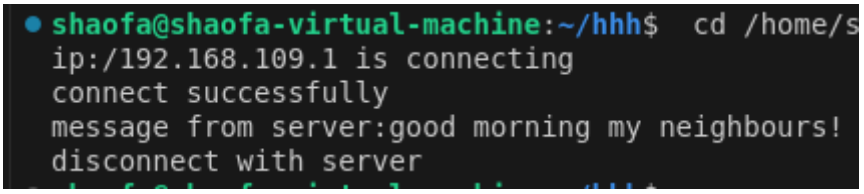
```
        BufferedReader read = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
//socket.getInputStream()返回一个InputStream对象，用来读取从客户端发送过来的字符流
//new InputStreamReader()接收一个InputStream对象，将字节流转换成字符流
//new BufferedReader()是包装流，在这里包装了一个InputStreamReader对象，能调用
readline()方法一次读取一行

        String line;
        line = read.readLine();//read the whole line
        System.out.println("message from server:"+line);
        socket.close();
```

2.客户端的代码

```
//发送一段消息的到服务器
PrintWriter out = new PrintWriter(socket.getOutputStream(), true);//true 表示自动刷
新，确保消息能及时发送
//socket.getOutputStream()返回一个OutputStream对象，用于向服务器发送数据
//PrintWriter是一个包装器类，它可以将一个OutputStream或Writer对象包装起来，提供方便的打印方
法，如print、println等。
//
String message = "good morning my neighbours!";
out.println(message);
System.out.println("已发送消息到服务器：192.168.109.131");
```

先运行服务端，然后再运行客户端的代码，就可以看到服务端显示客户端发送的消息



服务端的完整程序