实验四非阻塞通信——非阻塞的航

班显示服务器的实现

姓 名 梁寒冰 学号170341317

小组编号 28 小组其他成员的学号170341301 170341306 170341308 170341331

实验日期 2019年12月01日 实验报告日期2019年12月01日

成 绩 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. 实验目的

1、 理解并掌握非阻塞通信的基本原理

2、 用Java实现非阻塞通信服务器的设计

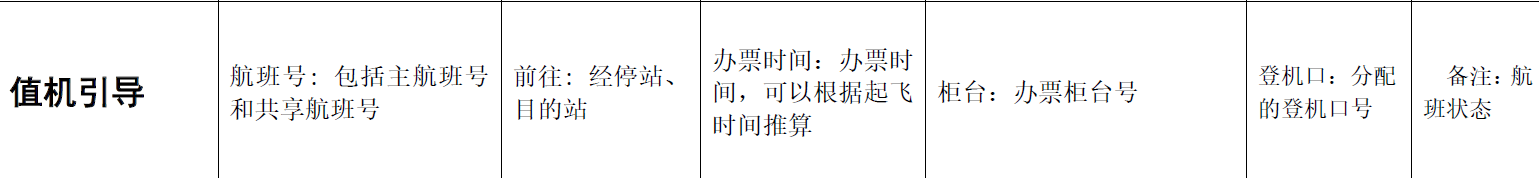
1. 实验环境

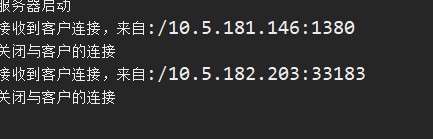
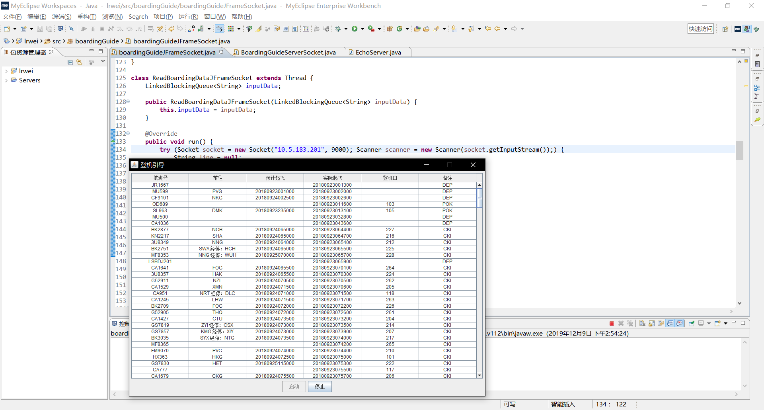
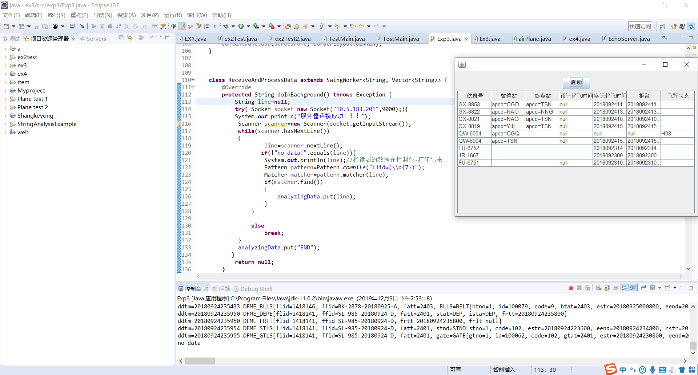
JDK，eclipse

文件里./src/ex/EchoServer为本次实验内容。

1. 实验实际完成内容及结果分析(请先说明你在小组中所承担的任务)

负责值机引导：



1. 本次实验主要是要综合运用非阻塞通信技术，在完成实验三航班动态显示系统客户端程序的基础上，开发航班显示系统数据服务器端程序，具体内容如下：创建应用服务器的监听端口（端口： 9999 ），从 航班信息文本文件fdsdata.txt 获取航班的动态信息，并使用尽量少的线程为每个连接服务器的航显终端提供动态的航班数据，实现具有并发能力的航显数据服务器。
2. 
3. 
4. 

源码：

package ex;

import java.io.\*;

import java.nio.\*;

import java.nio.channels.\*;

import java.nio.charset.\*;

import java.net.\*;

import java.util.\*;

public class EchoServer{

private Selector selector = null;

private ServerSocketChannel serverSocketChannel = null;

private int port = 9000;

private Charset charset=Charset.forName("GBK");

public EchoServer()throws IOException{

selector = Selector.open();

serverSocketChannel= ServerSocketChannel.open();

serverSocketChannel.socket().setReuseAddress(true);

serverSocketChannel.configureBlocking(false);

serverSocketChannel.socket().bind(new InetSocketAddress(port));

System.out.println("服务器启动");

}

public void service() throws IOException{

serverSocketChannel.register(selector, SelectionKey.OP\_ACCEPT );

while (selector.select() > 0 ){

Set readyKeys = selector.selectedKeys();

Iterator it = readyKeys.iterator();

while (it.hasNext()){

SelectionKey key=null;

try{

key = (SelectionKey) it.next();

it.remove();

if (key.isAcceptable()) {

ServerSocketChannel ssc = (ServerSocketChannel) key.channel();

SocketChannel socketChannel = (SocketChannel) ssc.accept();

System.out.println("接收到客户连接，来自:" +

socketChannel.socket().getInetAddress() +

":" + socketChannel.socket().getPort());

socketChannel.configureBlocking(false);

//ByteBuffer buffer = ByteBuffer.allocate(1024);

BufferedReader buffer = new BufferedReader(new InputStreamReader(new FileInputStream(new File("fdsdata.txt"))));

socketChannel.register(selector,

SelectionKey.OP\_READ |

SelectionKey.OP\_WRITE, buffer);

String ln=null;

ln=buffer.readLine();

if(ln.equals("no data")){

buffer.close();

}

}

if (key.isReadable()) {

receive(key);

}

if (key.isWritable()) {

send(key);

}

}catch(IOException e){

//e.printStackTrace();

try{

if(key!=null){

key.cancel();

key.channel().close();

}

}catch(Exception ex){e.printStackTrace();}

}

}//#while

}//#while

}

public void send(SelectionKey key)throws IOException{

SocketChannel socketChannel = (SocketChannel) key.channel();

BufferedReader reader=(BufferedReader)key.attachment();

String ln=null;

ln=reader.readLine();

ByteBuffer echoBuffer = charset.encode(ln+"\r\n");

echoBuffer.rewind();

while(echoBuffer.hasRemaining()) {

try {

socketChannel.write(echoBuffer);

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

echoBuffer.compact();

if(ln.equals("no data")){

key.cancel();

reader.close();

socketChannel.close();

System.out.println("关闭与客户的连接");

}

}

public void receive(SelectionKey key)throws IOException{

ByteBuffer buffer=(ByteBuffer)key.attachment();

SocketChannel socketChannel=(SocketChannel)key.channel();

ByteBuffer readBuff= ByteBuffer.allocate(1024);

socketChannel.read(readBuff);

readBuff.flip();

buffer.limit(buffer.capacity());

buffer.put(readBuff);

}

public String decode(ByteBuffer buffer){ //解码

CharBuffer charBuffer= charset.decode(buffer);

return charBuffer.toString();

}

public ByteBuffer encode(String str){ //编码

return charset.encode(str);

}

public static void main(String args[])throws Exception{

EchoServer server = new EchoServer();

server.service();

}

}

1. 思考题
2. 什么是非阻塞通信？

所谓阻塞方式的意思是指，当试图对该文件描述符进行读写时，如果当时没有东西可读，或者暂时不可写，程序就进入等待状态，直到有东西可读或者可写为止。而对于非阻塞状态，如果没有东西可读，或者不可写，读写函数马上返回，而不会等待（这也与设置的超时时间有关）

1. Java中实现非阻塞通信需要哪些类和接口的支持？

ServerSocketChannel:ServerSocket的替代类，支持阻塞通信与非阻塞通信。

SocketChannel：Socket的替代类，支持阻塞通信与非阻塞通信。

Selector:为ServerSocketChannel监控接受连接就绪事件，为SocketChannel监控连接就绪，读就绪和写就绪事件。

SelectionKey:代表ServerSocketChannel及SocketChannel向Selector注册事件的句柄。当一个SelectionKey对象位于Selector对象的selected-keys集合中时，就表示与这个SekectionKey对象相关的事件发生了。

ByteBuffer表示字节缓冲区，SocketChannel的read()和write()方法都会操纵ByteBuffer。ByteBuffer类继承于Buffer类。ByteBuffer中存放的是字节，为了把它们转换为字符串，还需要用到Charset类，Charset类代表字符编码，它提供了把字节流转换为字符串（解码过程）和把字符串转换为字节流（编码过程）的实用方法