11、net模块

- Node.js Net 模块提供了一些用于底层的网络通信的小工具，包含了创建服务器/客户端的方法，我们可以通过以下方式引入该模块：

```
const net = require("net")
```

- 具体方法参考手册

案例一：简单的socket连接

- server服务器端案例：

```
const net = require('net');//导入net模块
//创建一个socket服务器端对象server
const server = net.createServer((client) => {
  // 'connection' 监听器。
  console.log('客户端已连接');
  //为客户端添加断开连接的事件处理
  client.on('end', () => {
    console.log('客户端已断开连接');
  });
  client.write('你好\r\n');
  client.pipe(client); //管道：从哪里来的就流到哪里去
});
//为server绑定一个error事件处理
server.on('error', (err) => {
  throw err;
});
//开启server服务并监听8124端口
server.listen(8124, () => {
  console.log('服务器已启动');
});
```

- client客户端代码：

```
//使用net模块建议一个通讯的客户端
const net = require('net'); //导入net

//连接服务器，端口为8124
const client = net.connect({port:8124},() => {
  console.log("客户端已连接! ");
  client.write("world!\r\n");
});

//为当前客户端绑定接收数据的事件处理
client.on("data",(data) => {
  console.log(data.toString());
});
```

案例二：多客户端连接一服务器端实现信息交流

- server.js服务器端

```
//使用net模块建议一个可以连接很多客户端的服务器
const net = require('net'); //导入net

const clientSet = new Set(); //声明一个存放客户端对象的Set集合

//创建一个服务器端对象
const server = net.createServer((client) => {
  //此处的client就是客户端的连接对象
  // 'connection' 监听器。
  console.log(client.remoteAddress+'客户端已连接');
  clientSet.add(client); //将当前客户端对象添加到Set中
  //为客户端绑定一个断开的事件处理
  client.on('end', () => {
    clientSet.delete(client);
      console.log('客户端已断开连接');
  });

    //绑定一个数据的事件处理（接收客户端数据）
  client.on('data', (data) => {
    //遍历所有的客户端，并分发消息
    for(let cs of clientSet){
      cs.write(client.remoteAddress+":"+data.toString())
    }
  });
});

//绑定一个error错误事件处理
server.on('error', (err) => {
    //throw err;
    console.log("error! ")
});
```

```
//开启服务监听（端口8124）
server.listen(8124, () => {
  console.log('服务器已启动');
});
```

- 客户端：client.js

```
//使用net模块建议一个通讯的客户端
//运行的命令：node client2.js 192.168.1.7 8124
const net = require('net'); //导入net

//获取命令行中的参数
const hostname = process.argv[2];
const port = process.argv[3];


//连接服务器端
const client = net.connect({host:hostname,port:port},() => {
  console.log("客户端已连接! ");
  //接收键盘输入值，并发送服务器端
  process.stdin.setEncoding('utf8');//设置字符编码
  process.stdin.on('readable', () => {
    let chunk;
    // 使用循环确保我们读取所有的可用数据。
    while ((chunk = process.stdin.read()) !== null) {
      //判断输入q退出
      if(chunk == "q\r\n"){
        console.log("exit! ");
        process.exit();
      }
      client.write(chunk);
    }
  });
});

//为当前客户端绑定接收数据的事件处理
client.on("data",(data) => {
  console.log(data.toString());
});

//与服务器端断开连接事件处理
client.on("end",(data) => {
  console.log("客户端已断开连接");
});
```