中间件技术 实验1

22920212204392 黄勖、22920212204385 胡翼翔、22920212204492 张靖源

1　实验目的

- 理解中间件的基本概念
- 理解消息型中间件的基本概念
- 掌握基于中间件技术进行简单开发的基本过程

2　实验环境

系统：Windows 10

软件：

- VSCode with C++
- Winsock
- Java with Springboot
- Activemq

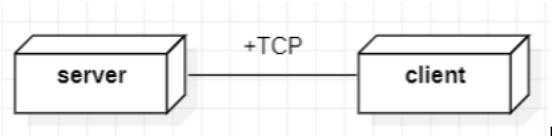
3　实验步骤

3.1　step1

此部分相关文件存放于step1文件夹下。

a）第一版设计：

可以通过Winsock和套接字编程实现，分为服务器（Server）和客户端（Client）两个程序。服务器负责接收客户端传递过来的数据。服务器和客户端之间应该是一对多的关系。



当客户端或非正常退出（例如崩溃或被其他程序中止）时，服务器应该断开连接但不崩溃。客户端可以定期（例如每30分钟）向服务器报告其状态。如果超过一定时间没有收到报告，服务器则认定客户端崩溃。

当服务器崩溃时，客户端应该能够在重新启动时恢复连接。服务器重启后，如果有新的客户端尝试连接，服务器不应拒绝已认证用户的连接。

b）在a的基础上编程，实现通过消息传递来传递关机指令。

客户端和服务端的初始化如下：

```cpp
void initialization() {
    //初始化套接字库
    WORD w_ver = MAKEWORD(2, 2);//版本号
    WSADATA wsaData;
    int err;
    err = WSAStartup(w_ver, &wsaData);
    if (err != 0) {
        cout << "客户端初始化失败" << endl;
        WSACleanup();
    }
    //检测版本号
    if (LOBYTE(wsaData.wVersion) != 2 || HIBYTE(wsaData.wHighVersion) != 2) {
        WSACleanup();
        cout << "客户端初始化失败" << endl;
    }
    cout << "客户端初始化成功" << endl;
}
```

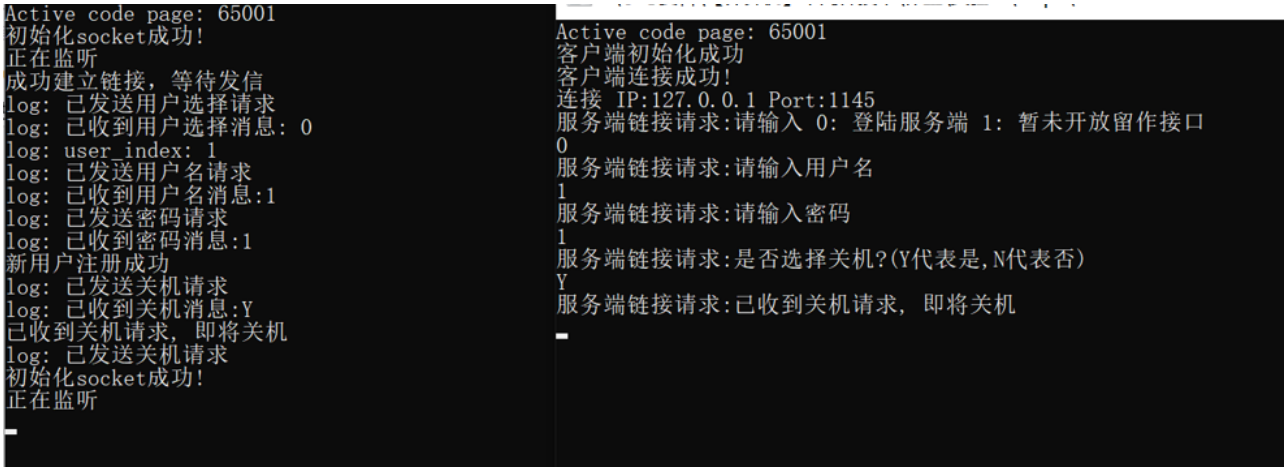
客户端连接如下：

```cpp
    //创建套接字
    s_server = socket(AF_INET, SOCK_STREAM, 0);//吸收socket流判断是否链接
    if (s_server <= 0) {
        cerr << "socket连接失败" << endl;
        return 1;
    }
    if (connect(s_server, (SOCKADDR*)&A.ServerAddress, sizeof(SOCKADDR)) == -1) {
        cout << "失败!" << endl;
        WSACleanup();
    }
    else {
        cout << "客户端连接成功!" << endl;
        cout << "连接 IP:" << SERVER_IP << " Port:" << SERVER_PORT  << endl; //创建成功
    }
```

客户端收发信息如下：

```cpp
    //发送,接收数据
    while (1) {
        recv_len = recv(s_server, recv_buf, 100, 0);
        if (recv_len < 0) {
            cout << "连接已断开！接受失败!" << endl;
            cout << "正在尝试重新连接..." << endl;
            break;
        }
        else {
            cout << "服务端链接请求:" << recv_buf << endl;
        }
        cin >> send_buf;
        send_len = send(s_server, send_buf, 100, 0);
        if (send_len < 0) {
            cout << "连接已断开！发送未成功!" << endl;
            cout << "正在尝试重新连接..." << endl;
            break;
        }
    }
```

服务端连接、收发信息编写原理同上，完成后运行结果如下，成功实现client使server关机功能。



之后电脑关机，step1成功完成。

3.2 step2

从中间件的角度，分析步骤1软件设计的优缺点，从架构、工作量、标准化、跨异构能力等角度提出若干改进点，并给出理由。完成改进版设计，此为第二版设计。

优点：

1. 逻辑和架构相对简单，易于实现。
2. 基于计算机网络标准——TCP/IP协议，使用socket进行实现。

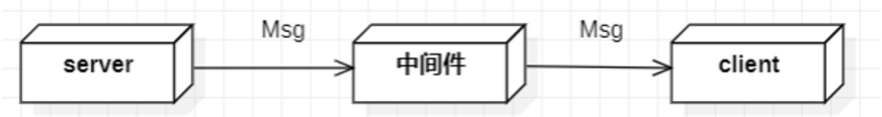
缺点：

1. 工作量较大。单从代码量的角度看，使用socket编程需要编写两个`.h` + `.c`文件，每个文件代码量均约为150行。
2. 跨异构能力较差。由于网络相关的库文件在Linux和Windows中名称、函数均有不同，因此在跨系统实现时必须修改相关代码。
3. 扩展性较差。对于后续可能的变化需求，需要对服务器（server）和客户端（client）进行大范围的代码修改。

改进点：

通过引入中间件技术，对消息传输进行封装。这样做的原因在于，通过中间件可以使得需要利用服务的人不必了解底层逻辑的具体实现，而是直接使用中间件提供的结果。这一改进可以有效解决工作量大、跨异构能力差以及对变化的保护不足的问题。

第二版设计：



3.3 step3

自学消息型中间件Queue和Topic的简单概念（大约10-15分钟），选择你认为合适的模型，改进步骤2的设计，并完成第三版设计（写出理由）。

队列（Queue）和主题（Topic）是JMS支持的两种消息传递模型：

类型	Topic	Queue
概要	Publish Subscribe messaging 发布订阅消息	Point-to-Point 点对点
有无状态	topic数据默认不落地，是无状态的。	Queue数据默认会在mq服务器上以文件形式保存，比如Active MQ一般保存在$AMQ_HOME\data\kr-store\data下面。也可以配置成DB存储。
完整性保障	并不保证publisher发布的每条数据，Subscriber都能接受到。	Queue保证每条数据都能被receiver接收。
消息是否会丢失	一般来说publisher发布消息到某一个topic时，只有正在监听该topic地址的sub能够接收到消息；如果没有sub在监听，该topic就丢失了。	Sender发送消息到目标Queue，receiver可以异步接收这个Queue上的消息。Queue上的消息如果暂时没有receiver来取，也不会丢失。
消息发布接收策略	一对多的消息发布接收策略，监听同一个topic地址的多个sub都能收到publisher发送的消息。Sub接收完通知mq服务器	一对一的消息发布接收策略，一个sender发送的消息，只能有一个receiver接收。receiver接收完后，通知mq服务器已接收，mq服务器对queue里的消息采取删除或其他操作。

通过学习中间件的队列（Queue）和主题（Topic），我认为队列模型更为合适。在一对一的情况下，队列和主题的核心功能没有区别。但是队列的消息生产者不需要在消息接收者接收该消息期间处于运行状态，消息接收者也不需要在发送消息时处于运行状态，而主题在发布者和订阅者之间存在时间依赖性；且队列会保存未被接收的消息，而主题订阅者发送的消息若无人接收，则会被直接丢弃。因此，在一对一发送消息的情况下，队列更为合适。

我们可以通过采用Spring框架、Java语言结合使用队列实现发布/订阅的消息中间件，从而实现通信的方便快捷。至此，我们完成了第3版设计的初步思路构想，即将消息存储在队列中，同时通过前端将服务器和客户端的消息显示出来，从而确保客户端一定能够接收到服务器发送的消息。

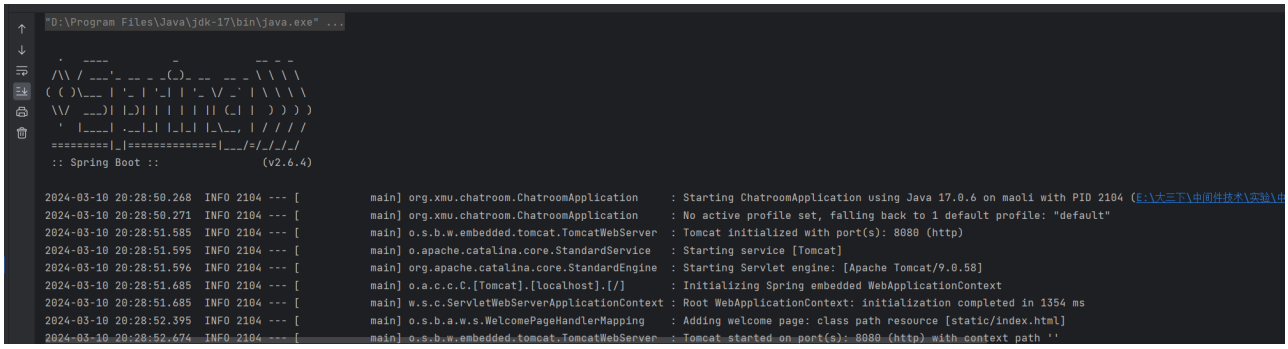
3.4 step4

此部分相关文件存放于step4文件夹下。

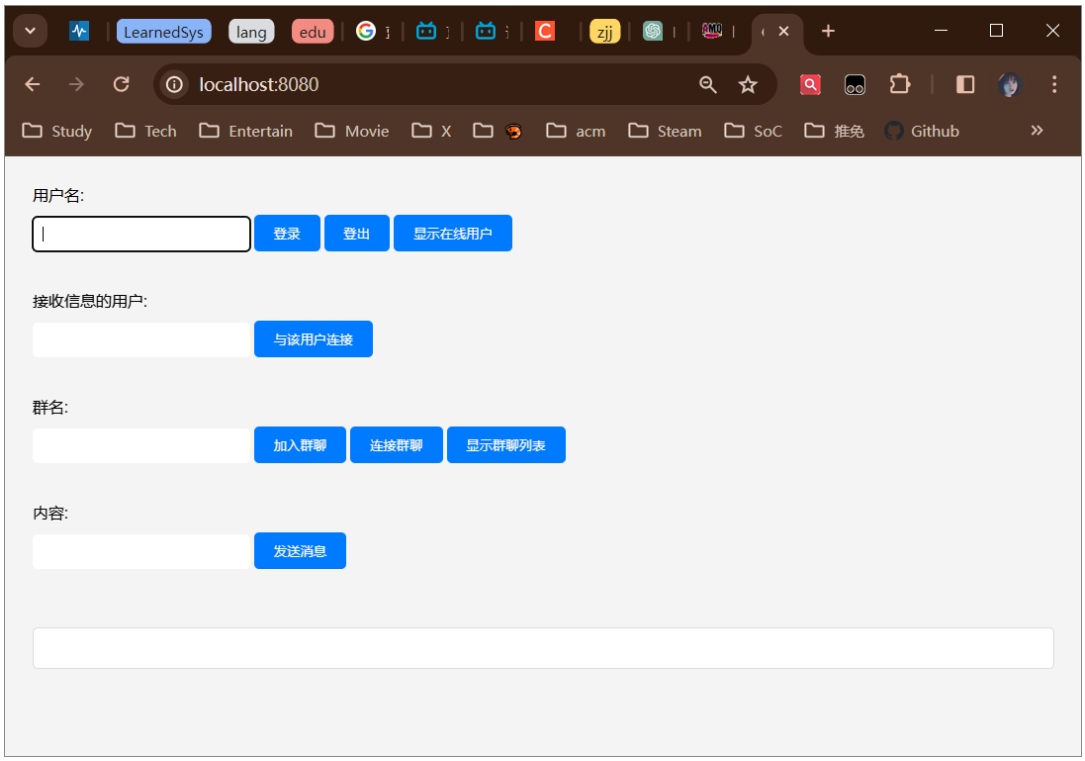
成果：采用springboot框架，java语言，中间件activemq完成即时通信软件，实现了群聊、私聊、消息存储，登陆登出等功能。

启动activemq：

后台启动:



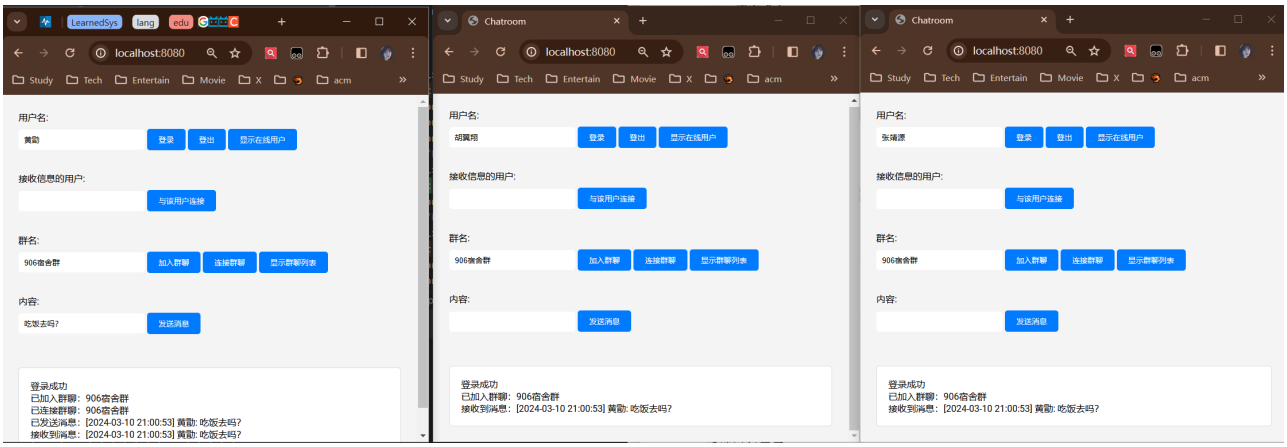
前端界面展示:



登陆登出功能展示:

私聊功能展示：

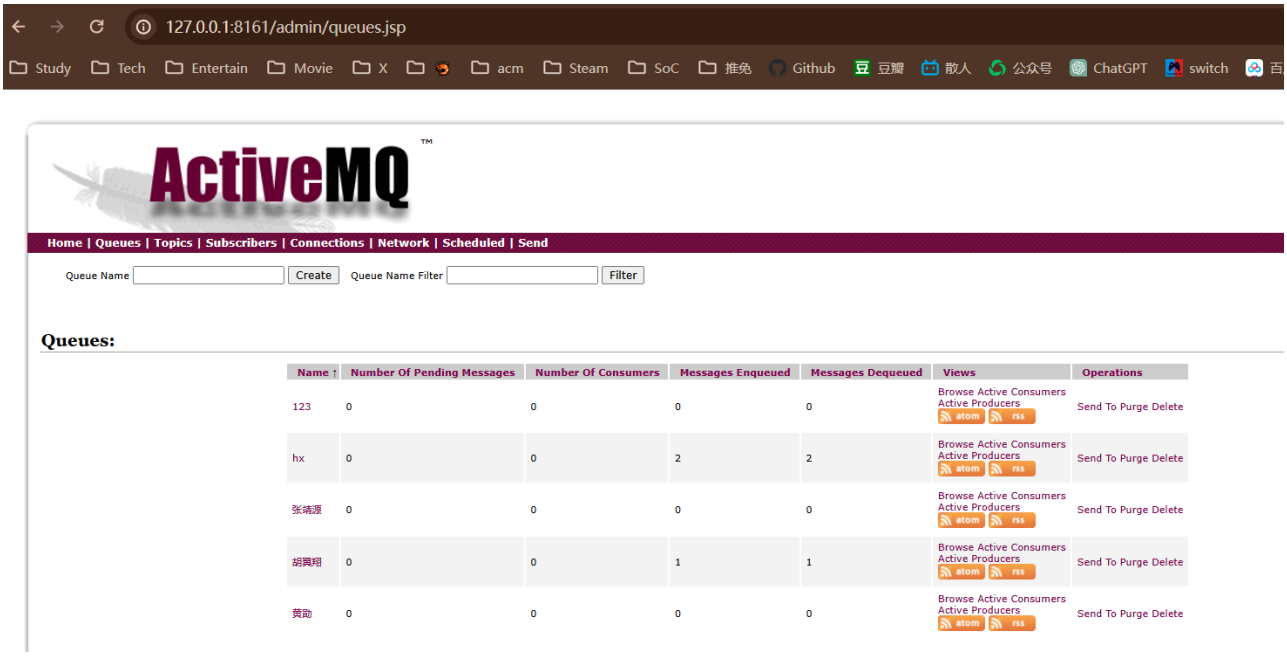
群聊功能：



显示在线用户：

显示群聊：

测试后activemq后台：



具体操作过程可以查看演示视频，至此成功实现功能。

4 实验总结

在本次实验中，我们成功地利用中间件相关技术实现了实验的要求，这让我们深刻体会到了中间件技术的便利性和重要性。通过使用中间件，我们可以将通信、数据传输等复杂的任务抽象成简单的接口，极大地简化了开发和维护的工作量。而且，中间件技术还能够提供高度可靠性和性能优化，使得系统在面对高并发、大规模数据处理等挑战时依然能够保持稳定和高效。总的来说，本次实验的成功实现不仅为我们提供了技术上的成果，也增强了我们对中间件技术在实际应用中的认识和信心。

5 实验分工

排序如下：

1 黄勖：完成步骤4、项目搭建

2 胡翼翔、张靖源：胡翼翔完成步骤1，张靖源完成步骤23