设计方案

服务器设计方案 (server.py)

初始化Socket:

创建一个UDP (SOCK\_DGRAM) socket对象。

绑定到本地地址127.0.0.1和端口9999。

日志记录:

使用logging模块记录服务器启动信息。

用户管理:

使用字典user来管理已连接的客户端，其中键是客户端地址，值是客户端名称。

消息接收与广播:

服务器无限循环接收来自客户端的消息。

如果客户端是新用户，服务器会向所有已连接的客户端广播新用户的加入。

如果接收到的消息包含'EXIT'，则从用户列表中移除该客户端，并广播该用户的离开。

对于其他消息，服务器将其广播给所有已连接的客户端（不包括发送者自己）。

异常处理:

捕获ConnectionResetError异常，这通常发生在客户端意外断开连接时。

关闭服务器:

客户端设计方案 (client1.py 和 client2.py)

初始化Socket:

创建一个UDP (SOCK\_DGRAM) socket对象。

连接到服务器:

客户端绑定到服务器的IP地址127.0.0.1和端口9999。

多线程处理:

创建两个线程：一个用于接收服务器的消息(recv)，另一个用于发送消息到服务器(send)。

发送消息:

客户端接收用户输入的消息，并将其构造为格式"用户名 : 消息内容"。-

客户端将消息编码为UTF-8格式，并发送到服务器。

接收消息:

客户端接收来自服务器的消息，并将其解码为UTF-8格式的字符串。

客户端打印接收到的消息。

退出机制:

如果用户输入'EXIT'（不区分大小写），客户端将停止发送消息，并退出程序。

关闭连接:

客户端在退出时关闭socket连接。

系统特点

UDP协议: 使用UDP协议，提供面向无连接的通信。

多客户端支持: 服务器能够处理来自多个客户端的消息。

广播消息: 服务器将接收到的消息广播给所有已连接的客户端。

用户加入和离开通知: 服务器向所有客户端通知新用户的加入和用户的离开。

线程安全: 客户端使用线程来处理发送和接收操作，避免阻塞。

安全性和异常处理

异常处理: 服务器端捕获客户端断开连接的异常。

输入验证: 客户端应添加对用户输入的基本验证，避免发送非法数据。

用户界面

控制台交互: 客户端通过控制台与用户交互，接收用户的输入并显示消息。