#### 一、课程设计概述

##### 1、设计内容

我们的目标是开发一款专为嵌入式Linux环境设计的局域网聊天应用。该应用基于TCP协议进行通信，核心架构包括一个负责消息中转和转发的服务器端，以及一个能够展示消息和发送消息的客户端。具体流程如下：客户端首先将消息格式化，随后发送至服务器；服务器接收这些消息，并根据需要将其转发给其他客户端或回发给原始发送者。每当客户端接收到消息时，它会解析并显示这些消息，以便用户进行交互和沟通。

此外该聊天室还包含群聊、私聊、发送文件、拍照等多项功能，能够完成原始的系统要求。

##### 2、开发工具及运行环境

Python 3.7，pyCharm IDE，Linux操作系统

##### 3、成员分工

代仲麒：总体构思，框架构成，独立使用C语言完成UI设计、登录界面设计和部分聊天室功能（私聊群聊禁言解禁等）。

姜吕涛：通信建立，群聊私聊功能的实现，bug修复

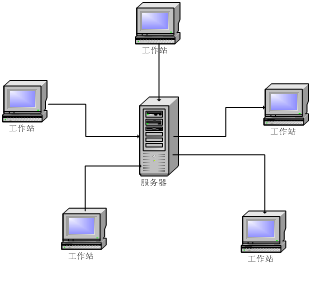
王翰禹：UI界面设计，消息撤回功能实现，bug修复

董磊：拍照功能实现，消息撤回功能实现，bug修复

雷文强：文件传输功能实现，bug修复

#### 二、系统需求分析

在数字通讯的现代领域，客户端/服务器（C/S）架构成为了标准配置，有效地将系统功能分为两个关键部分：前端的客户端与后端的服务器。此架构模式下，客户端的角色是发起请求，以访问和处理存储在远程服务器上的数据；服务器则承担接收这些请求，并进行相应处理的任务。特别地，在多用户环境中，服务器能够同时处理来自众多客户端的请求，展现出其处理并发请求的能力。C/S模型之所以在数据库应用和众多系统构建中广受欢迎，主要归功于网络技术的飞速发展。通过将任务分布在客户端和服务器端，这种模式充分利用了系统资源，提高了效率。服务器端的集中数据处理和存储机制进一步简化了数据访问过程，确保了所有客户端都能轻松获取所需信息。本质上，C/S架构依托于一种简洁高效的“请求/响应”模式。在这个模式下，一切交互始于客户端的请求，而服务器以被动的姿态等待并响应这些请求，返回所需的数据或服务。服务器在没有请求时处于待命状态，直至收到请求，便激活并执行相应的处理，满足客户端的需求。这种互动不仅确保了资源的优化利用，还为多用户访问提供了灵活性和高效性，是当下技术生态中不可或缺的架构模式。如下图所示。



根据上文对聊天系统进行的功能需求分析，可以确定该系统中应包含的所有必需模块，从而设计出聊天系统的层次图，其如下图所示。

基于linux的聊天功能设计与实现

聊天

拍照保存

消息撤回

文件传输

获得用户列表

登录

连接服务器

退出

群聊

私聊

##### 1、服务器功能设计

获得在线用户类表模块：用户连接成功后，收到客户端发来的用户名，并将其保存在服务器端，随后向客户端发送在线用户列表。

用户私聊模块：接收到客户端发送的私聊请求后，向指定用户转发聊天信息。

用户群聊模块：接收到客户端发送的群聊请求后，向所有在线用户转发聊天信息。

用户退出模块：更新在线用户列表，然后向所有在线用户发送在线用户列表。

文件传输模块：收到客户端发送来的文件，服务端对文件进行缓存，并将文件转发给接收人。

消息撤回模块：服务端收到消息撤回请求，将此请求转发给各个客户端，客户端收到撤回请求后，做出响应。

##### 2、客户端功能设计

连接服务器模块：客户端首先需要输入服务器的IP、端口号和用户名来连接已启动的服务器，当连接成功后才能和在线的用户聊天；若连接失败，则会要求重新输入IP和端口号。

私聊模块：私聊模块实现的是用户与另一用户之间的双方通信，其他用户对于聊天信息不可见，在该模块设计中发起聊天的用户可以输入sendto来实现向另一用户发起会话。

群聊模块：群聊模块实现的是当某一用户发起会话时，所有的用户均可收到聊天信息，所有在线用户可以共享聊天信息。

退出模块：当用户要退出该系统，则所有在线用户的在线用户列表都会刷新，在在线列表中将不会看到该用户的信息，从而实现退出功能。

界面模块：该系统的聊天界面，大体上分为左右结构右边的界面用来发送信息及显示聊天信息，左边的界面用来显示在线用户列表。

文件传输模块：当用户要向某一用户发送文件，需输入sendfile和文件路径以及被文件接收人实现向文件传输。

拍照模块：拍照模块实现了用户可以调用摄像头进行自拍并保存照片的功能，用户需输入takephoto来启动拍照界面。

消息撤回模块：消息撤回模块实现的是当用户需要撤回上一条信息时，输入withdraw便可撤回。

3、数据传输规范设计

网络聊天室的数据传输规范至关重要，它保障信息传递的准确性和安全性，提升用户体验。规范化的传输机制确保信息无论在何种设备或平台间传递，都能保持一致性和完整性，避免因格式不匹配导致的数据丢失。同时，统一的传输标准增强了不同设备和系统间的兼容性，确保了所有用户能享受到无缝且高质量的沟通体验。规范化的数据处理机制提升了系统的可维护性和扩展性，有助于开发者快速识别和解决问题，从而维护了网络聊天室的稳定运行和用户的信任。

本文采用字典形式作为数据传输的主要格式，通过对字典的键值对进行约束，从而对各个功能进行区分。

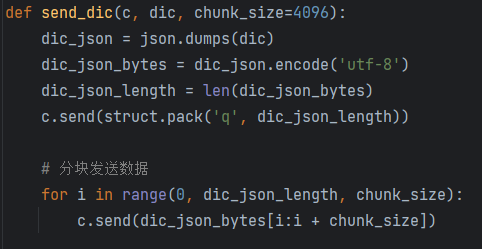
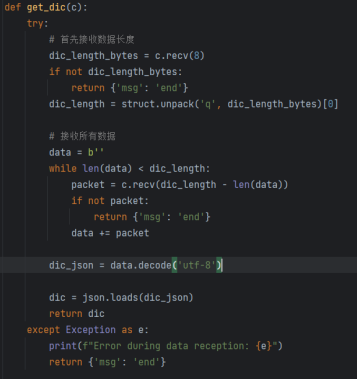
#### 三、系统设计与实现方案

##### 1、消息发送与接收原理

消息发送是通过send\_dic函数实现。在send\_dic函数中，首先，函数将字典转换为JSON字符串，然后计算这个字符串的字节长度。这个长度信息被打包（使用struct.pack方法）成固定长度（这里是8字节），然后发送给接收方。发送长度信息的目的是让接收方知道应该接收多少字节的数据，这是为了确保数据传输的完整性。

为了高效传输大量数据，send\_dic函数将数据分成多个块（每个块大小由chunk\_size参数指定，默认为4096字节），然后逐个发送这些数据块。这种方法可以减轻网络压力，避免一次性发送大量数据可能导致的网络阻塞。

消息接收通过get\_dic函数实现，。get\_dic函数的过程与send\_dic过程相反，因为在发送过程中，将长度信息固定为8字节进行打包，因此函数首先接收一个8字节长的数据。使用struct.unpack解析这8字节数据，得到实际数据的长度。随后循环接收数据，收到全部数据后，函数将字节数据解码为JSON字符串，然后使用json.loads方法将字符串反序列化为字典对象



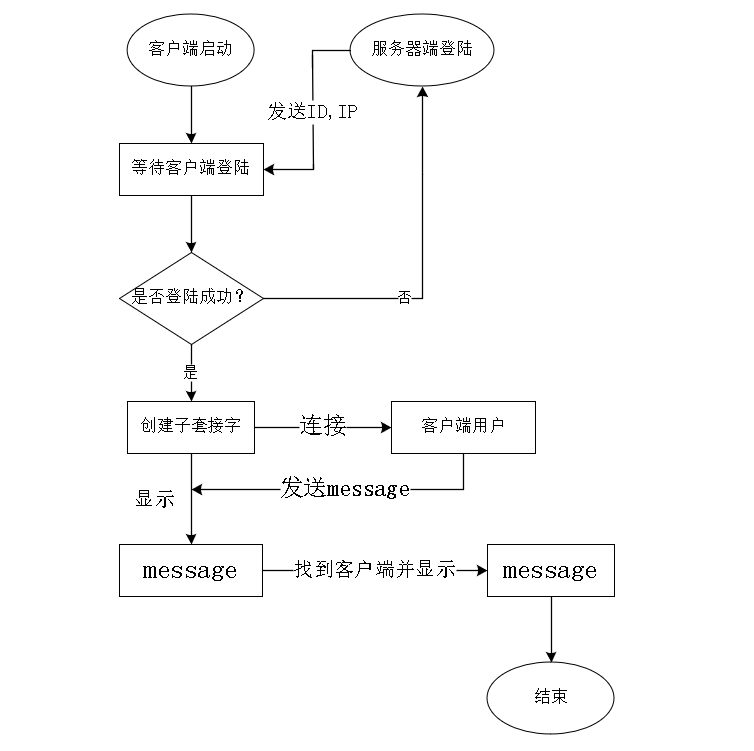
##### 2、数据规范设计

消息采用字典的格式，其聊天基本字段包括了几个关键字段：msg字段携带消息的主体内容，这是通信的核心；name字段标识消息发送者的用户名，用于在聊天室中标识不同的参与者；针对私聊功能，recipient字段用于指定消息的接收者，private作为私聊标志位，向服务端传送是否为私聊信息，而在群聊模式下，此字段可能不被使用或为空，表示消息是公开发送给所有在线用户的。

##### 3、服务器设计原理

3.1 建立通信

服务端通过创建socket对象并绑定到指定的IP地址和端口上，等待客户端的连接请求。使用socket.listen()监听来自客户端的连接请求，当一个客户端请求连接时，通过socket.accept()方法接受连接，建立客户端和服务端之间的通信通道。考虑到聊天室需要同时处理多个客户端的消息，服务端采用ThreadPoolExecutor来实现并发处理。每当一个新客户端连接到服务端时，都会为这个客户端创建一个新的线程（通过线程池提交任务），用于监听和处理该客户端发送的所有消息。这种并发处理机制使得服务端能够高效、实时地响应多个客户端的请求，保证聊天室的流畅通信。以下是客户端与服务端连接的流程图：



3.2 用户管理

服务端创建了一个客户端列表c\_list，记录了所有已连接客户端的信息（包括地址、socket对象和用户名）。当需要广播消息时，服务端遍历这个列表，将消息发送给列表中的每一个客户端。同样，当用户列表发生变化时（新用户加入或用户退出），服务端也会通过broadcast\_user\_list函数更新所有客户端的用户列表，实现聊天室成员管理。



3.3 信息处理

服务端支持群聊、私聊功能，通过解析客户端发送的消息字典中的private和recipient字段来判断消息是否为私聊以及私聊消息的接收者。对于私聊消息，服务端只将消息转发给指定的接收者，而不是广播给所有客户端，保证了私聊消息的隐私性。此外，服务端还能处理特殊的消息类型，如文件传输请求和撤回消息请求。

3.4 文件传输

用户选择文件传输功能后，消息处理函数执行对应选择状态，提示用户输入将要发送文件的目标用户，接着提示用户输入所要发送文件的文件名，服务器端调用消息处理函数，选择文件名对应的文件标识符，然后执行文件复制的函数，将文件读取写入到另一用户的地址中，发送成功后，提示消息发送成功。

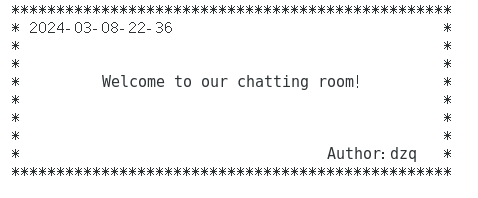
3.5 异常处理与日志记录

服务端在处理客户端消息和文件传输过程中，加入了异常处理逻辑，确保了服务端的稳定运行。当遇到异常情况时，服务端通过日志记录功能记录下异常信息，便于问题的追踪和调试。

##### 4、客户端设计原理

##### 4.1 界面设计

登录界面设计为两种，第一种为由c语言编写，结合时钟功能用printf敲出逐步显示的welcome to our chatting room。第二种使用python自带的库文件完成界面的设计。其中，c语言的设计如下图所示：



Python界面设计：

在设计客户端界面时，我们经过团队商量与讨论决定采用python的curses库，该库是在Linux/Unix下广泛应用的图形数据库，可以在终端内绘制简洁的图形用户界面，curses用stdscr和curscr来映射终端屏幕，当我们调用refresh函数时，函数库会将当前显示的屏幕curscr刷新为改变的stdscr的样子，这也是我们客户端UI显示信息的基本原理之一。

在设计客户端界面各模块的时候，我们根据老师所提供的思路结合目前的流行的客户端式样，将其分为三大部分：左上角为用户列表，客户端可以在此看到实时在线的用户，右上角为聊天框，也是客户端的主要模块；整个界面的最下面为input框，用于接收输入的信息以及各种命令。

在设计主体框架时，首先使用getmaxyx函数获取终端长宽height、width,接着使用curses.newwin函数创建三大主体框，框的长宽与上面获取的height、width存在一定的函数关系，在绘制框实体时依次使用erase、box和addstr实现消息的显示，此外，我们在刚登陆时设置curses.curs\_set(0)，使光标不可见，此外也设置了 stdscr.nodelay(0)和stdscr.timeout(-1)分别用于设置stdscr.getch()为阻塞模式和禁用超时，能够使整个客户端运行起来更符合实际逻辑。

在代码编写的过程中我们采用了curses的特有的刷新机制，在每次将要刷新的地方使用noutrefresh函数，将其标记为刷新但等待，避免过多的刷新导致屏幕的卡顿等等问题，并在合适的位置调用curses.doupdate()函数进行整体刷新

此外，在编写UI过程中我们大量使用了getstr函数用于获取输入框中输入的文本信息，将获取的信息用于个函数进行处理， 以及echo和noecho函数用于控制从键盘输入的信息是否显示在终端屏幕上也广泛用于各函数。在光标位置的控制上采用了move函数将光标移至指定位置。

4.2 通信建立

通过输入IP 和 端口号传入Socket中建立连接，为了防止IP和端口号输入错误直接报错退出程序，因此设计了检查IP和端口号格式是否正确的函数，以及对连接无法建立的异常情况进行处理，发生此类异常情况，界面会显示连接无法建立，按下任何按键进行重新输入。

##### 4.3 群聊和私聊模块

Python实现：

消息的发送主要在send\_msg函数中实现，该函数首先等待用户在图形界面的输入窗口（input\_win）输入文本。用户输入的文本通过input\_win.getstr方法捕获，并根据输入内容的不同（如普通消息、私聊指令sendto、发送文件sendfile等），执行相应的逻辑处理。

对于普通群聊消息，用户输入的文本被封装成一个包含消息内容、发送者名称等信息的字典对象，通过common.send\_dic函数发送给服务器。

客户端收到服务端转发过来的数据后，对消息进行接收。这一部分主要是通过get\_msg函数实现，该函数在一个单独的线程中运行，不断监听来自服务器的消息。当服务器转发其他客户端发送的消息时，get\_msg函数通过common.get\_dic方法接收并解析这些消息

接收到的消息首先被检查其类型（如普通群聊消息、私聊消息或是文件传输指令等）。根据消息类型和内容，客户端更新本地的聊天记录，并通过display函数实时更新聊天界面，展示新接收的消息。

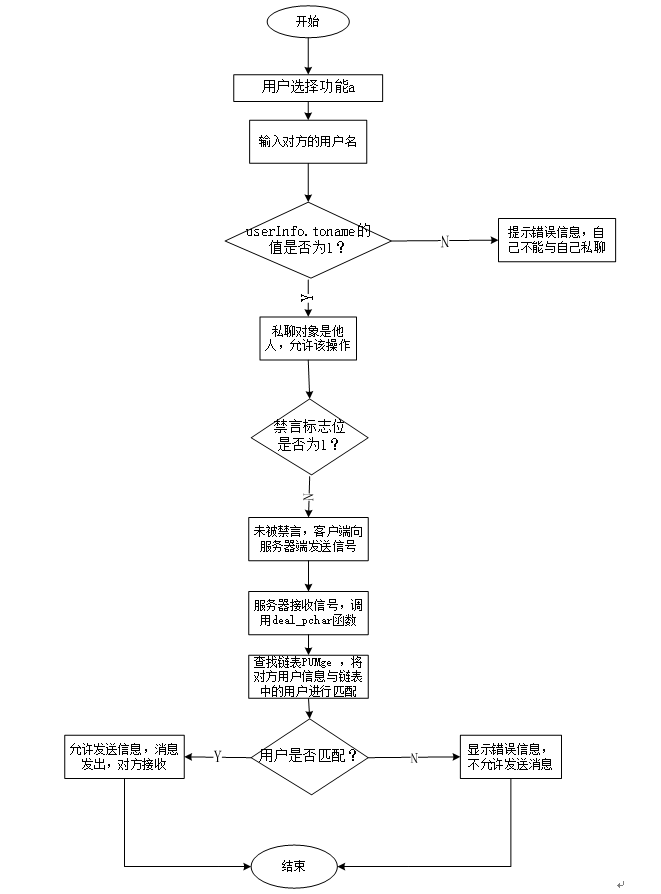
在接收和处理消息的过程中，同时对各种异常情况，如网络断开、消息格式错误等，进行日志记录，以及相应处理，以确保客户端的稳定运行和良好的用户体验。

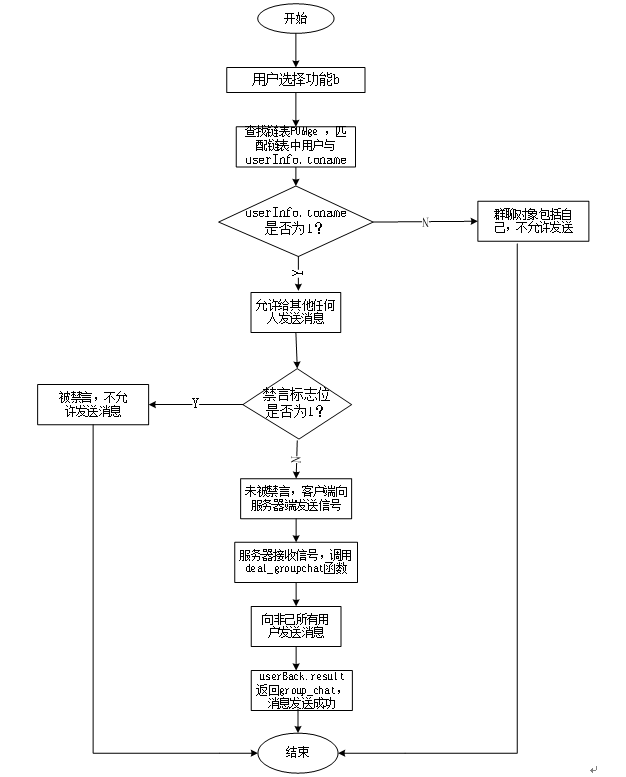
对于私聊信息，若用户输入指令发起私聊（例如sendto），则会额外输入接收者的用户名，消息通过上文中讲解的发送过程被封装并发送给服务器，但服务器只会将这些消息转发给特定的接收者。这种实现方式确保了私聊消息的隐私性和目标性，即消息只被发送和显示给指定的用户。

以下是C语言实现群聊和私聊的思维流程图，总体思路两种语言完成情况是相同的，具体函数有所不同。

私聊功能：用户选择私聊功能后，提示输入用户想要私聊的另一个用户名，然后提示要发送的消息，用户输入后，显示消息发送中，然后借由套接字与服务器建立通信，服务器的消息处理函数判断该条命令为私聊，则检索userinfo结构中的数据，找到目的用户套接字进行通信，处理成功后消息发送到目标用户，同时给消息发送端发送消息成功发送的信号，消息发送端收到后显示成功发送。

选择群聊功能后，输入发送消息，回车后检测userinfo结构中name中发送对象与自己是否匹配，即发送对象是否为自己，若匹配，则拒绝请求，若不匹配则允许发送，利用套接字与服务器通信，再由服务器消息处理函数广播消息给其他用户，要求输入要发送的消息，输入后查找链表，允许发送后，由服务器处理，发送成功后返回消息发送成功的提示，并且检测到回车键后返回主界面





##### 4.4 python多线程处理

在Python中，多线程是通过创建Thread对象来实现的。Python的多线程模块threading提供了一个高级的线程接口，它封装了低级的\_thread模块，提供了更多的功能和方便的API。

在本文中，客户端服务端的多线程都是通过ThreadPoolExecutor函数实现。ThreadPoolExecutor作为Python并发编程中的一项核心工具，其设计理念是通过线程池来管理和调度线程，有效减少了线程创建和销毁的开销，提高了程序的执行效率。线程池内部维护一个任务队列和一组工作线程，任务队列用于存放待执行的任务，工作线程则负责执行这些任务。当ThreadPoolExecutor被实例化时，可以指定池中线程的数量，一旦设定，这个数量通常不会改变。当提交一个任务给ThreadPoolExecutor时，它会首先尝试将任务放入任务队列。如果队列中已有任务在等待，新的任务就会排队等待执行；如果有空闲线程，则会立即从队列中取出任务进行执行。工作线程执行完毕后不会立即销毁，而是继续在池中等待新的任务。这种方式大大提升了任务执行的响应速度，并减轻了系统对线程创建和销毁操作的负担。ThreadPoolExecutor还提供了灵活的任务提交方式，包括submit和map等方法，使得并行任务的实现更加简单直观。此外，通过Future对象，我们可以获取任务的状态和结果，实现了对并发任务更细致的控制。整体来看，ThreadPoolExecutor通过高效的线程复用和任务调度机制，优化了多线程程序的性能，是Python并发编程中不可或缺的工具。

##### 4.5 拍照模块

在拍照模块中，我们使用了tkinter库和opencv库实现对电脑摄像头的调用和创建拍照UI。我们首先定义了take\_photo()和save\_photo()两个执行函数，用来实现对图片帧进行抓取和保存。同时还定义了show\_camera\_feed()和show\_image()实现将图片帧转化成视频流进而实时显示摄像头以及对拍摄拍照进行预览。其中lmain.after(10,show\_camear\_feed)使用 Tkinter 的 after 方法，show\_camera\_feed 函数将在 10 毫秒后再次被调用。这创建了一个递归循环，使得 GUI 能够持续更新并显示摄像头的实时视频流。需要注意的是，OpenCV 默认使用 BGR 颜色空间，而 Tkinter 使用的 GUI 库（如 PIL/Pillow）则使用 RGB 颜色空间。因此，需要将图像从 BGR 转换为 RGB。这里使用 cv2.cvtColor 函数完成颜色空间转换。通过对执行函数和底层函数的分开定义，能够使代码更具有可读性和修正性，方便对代码进行调试和修改。对于拍照UI的创建，我们选用了tkinter库函数，实现对摄像头的预览和执行功能。设定’拍照’、‘保存照片’、‘退出’三个Button，以‘拍照’为例，使用bct\_take\_photo = tkinter.Button(root, text = ‘拍照’, common = take\_photo)将‘拍照’Button和take\_photo()这一函数关联起来，达到点击Button即实现拍照功能的目的。由于我们的主程序是在终端UI上执行的，当想要进行拍照时，不得不生成一个tkinterUI，但在我们的调试过程中，发现终端UI和tkinterUI跳转会有冲突，导致在创建tkinterUI界面后终端UI直接崩溃，无法进行其他操作。针对这个问题，我们采用多线程来解决，定义terminal\_ui\_handler()函数，在这个函数下，当终端UI传入‘takephoto’时，即可创建一个新线程，实现对tkinter的创建。

4.6消息撤回模块

在消息撤回模块中，我们也采用字典传输信息的方式将输入的指令‘withdraw’传输给服务端，并在服务端对接收到的信息进行处理，当接收到‘withdraw’时对所有客户端进行广播，并在客户端的接收信息函数中进行聊天框消息的撤回处理，采用message.pop消除最后一条消息，从而达到消息撤回的效果。

4.7文件传输模块

主要包括两个部分：receive\_file 函数用于接收文件，send\_file 函数用于发送文件。

receive\_file 函数主要经过以下步骤：

1）、接收文件信息的长度：首先从连接中接收10个字节，这10个字节代表着后续要接收的文件信息（JSON格式）的长度。之所以固定接收10个字节是为了确保可以准确地解析出长度信息。接收到的字节串可能包含填充的空字符（\x00），因此需要去除这些空字符来获取实际的长度字符串，再将其转换成整数。

2）、接收文件信息：根据步骤1中解析出的长度，接收文件信息本身。这部分信息是一个JSON字符串，包含了文件名(filename)和文件大小(filesize)。接收到的信息被解码成字符串，并转换成JSON对象以提取这些信息。

3）、准备接收文件内容：根据接收到的文件名，在指定的目录中创建新文件，准备将接收到的文件内容写入这个新文件。

4）、接收文件内容：通过循环接收文件内容。每次循环接收的最大字节数为4096字节，直到接收到的数据大小等于文件预期的总大小。接收到的每个数据块（chunk）被写入文件中。如果在过程中连接关闭（即recv返回空字符串），则终止循环。

send\_file 函数主要经过以下步骤：

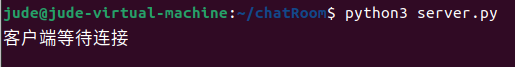
1）、准备文件信息：首先获取文件的基本信息，包括文件名（从完整路径中提取）和文件大小。然后将这些信息封装成一个字典，并转换成JSON字符串。

2）、发送文件信息的长度和文件信息：首先发送文件信息的长度，为了确保服务器能正确解析长度信息，将长度转换成字符串并用空字符填充到10个字节。然后发送文件信息本身的JSON字符串。

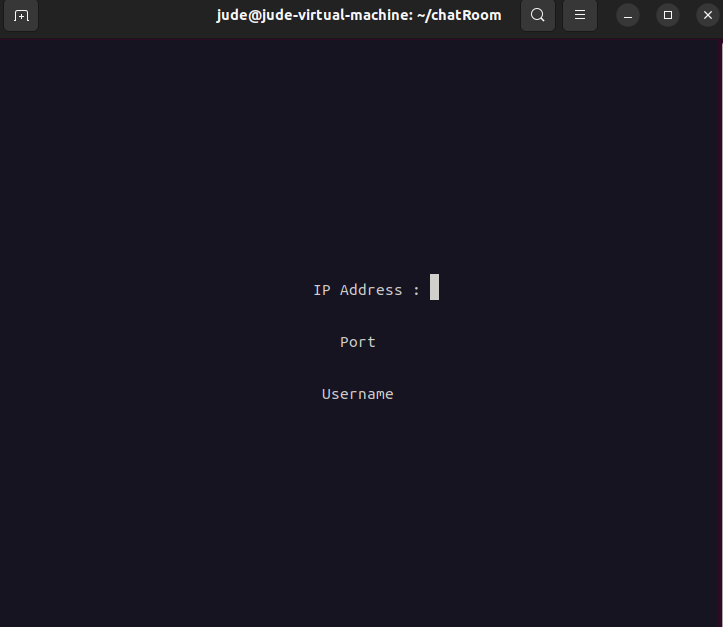
3）、发送文件内容：以二进制模式打开文件，循环读取每个4096字节大小的数据块，然后通过连接发送这些数据。循环直到文件结束。

**四、系统测试**

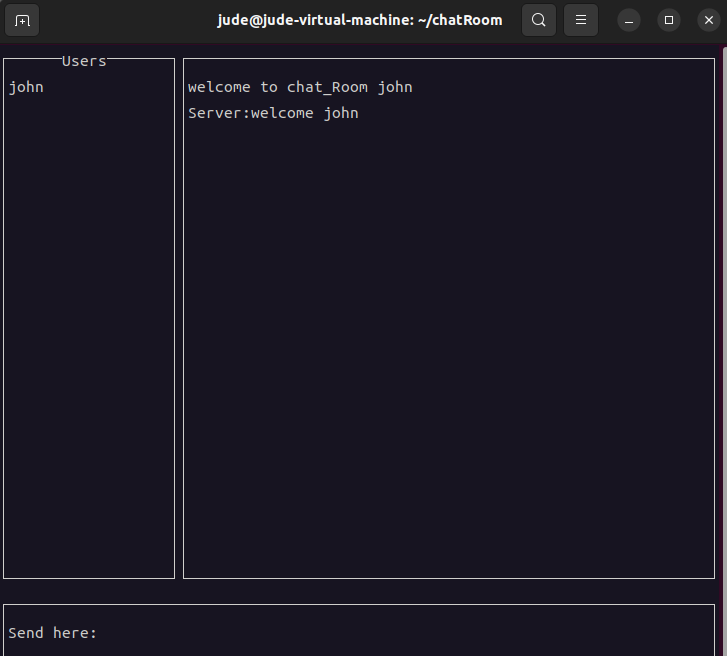
1、启动服务器



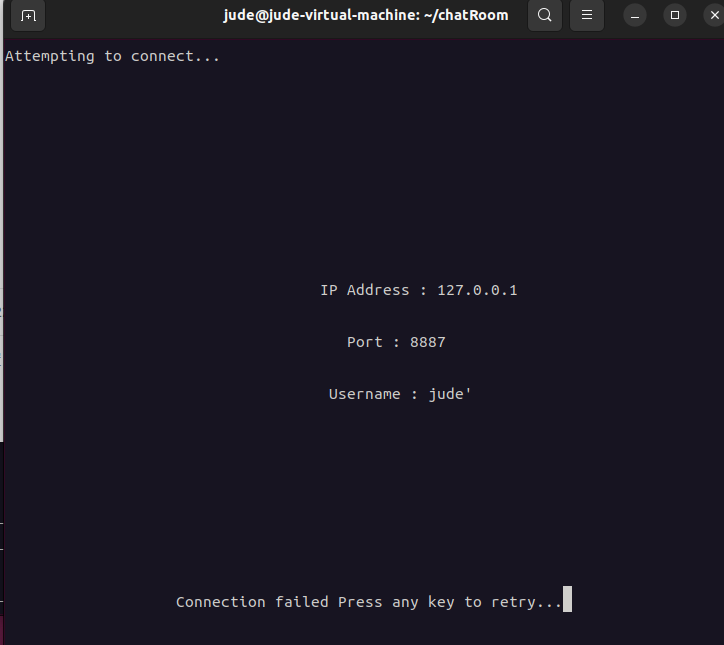
2、启动客户端，显示登陆界面



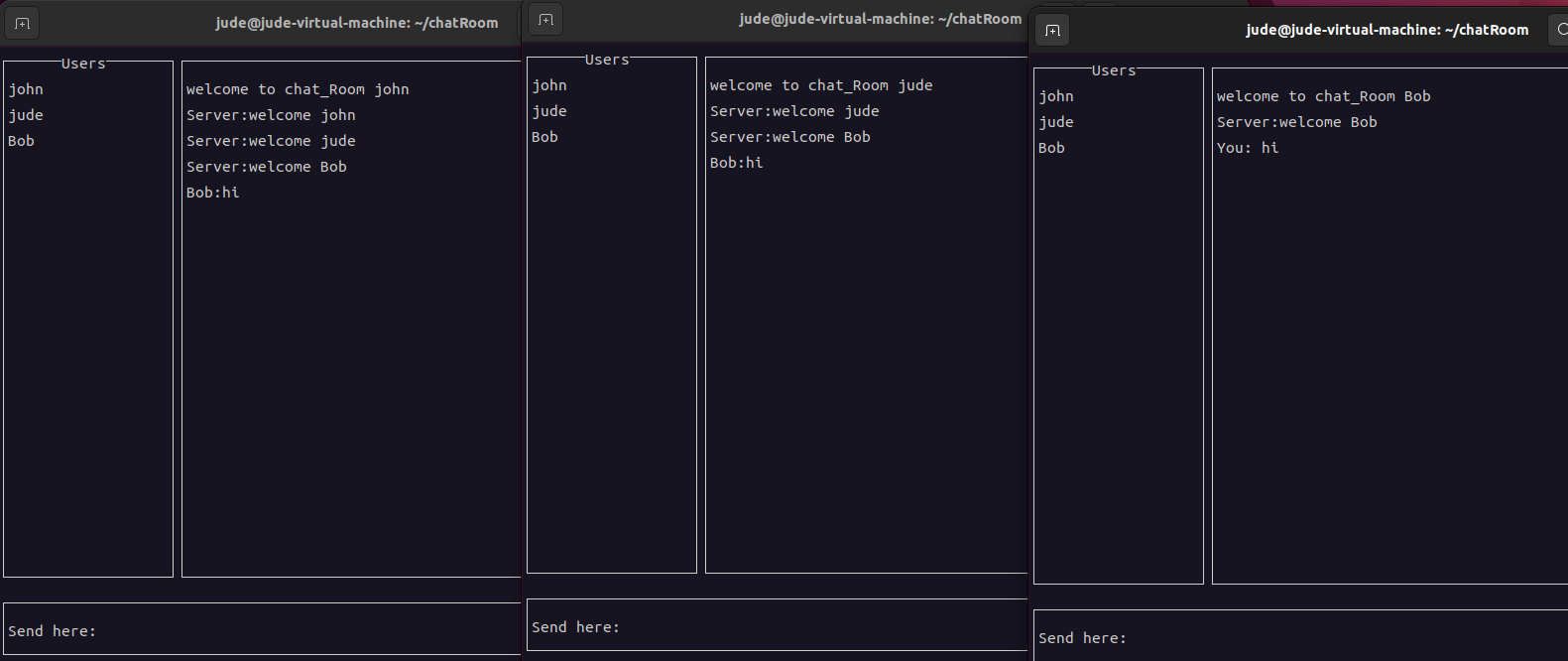
1. 正确输入IP和Port，进入主界面



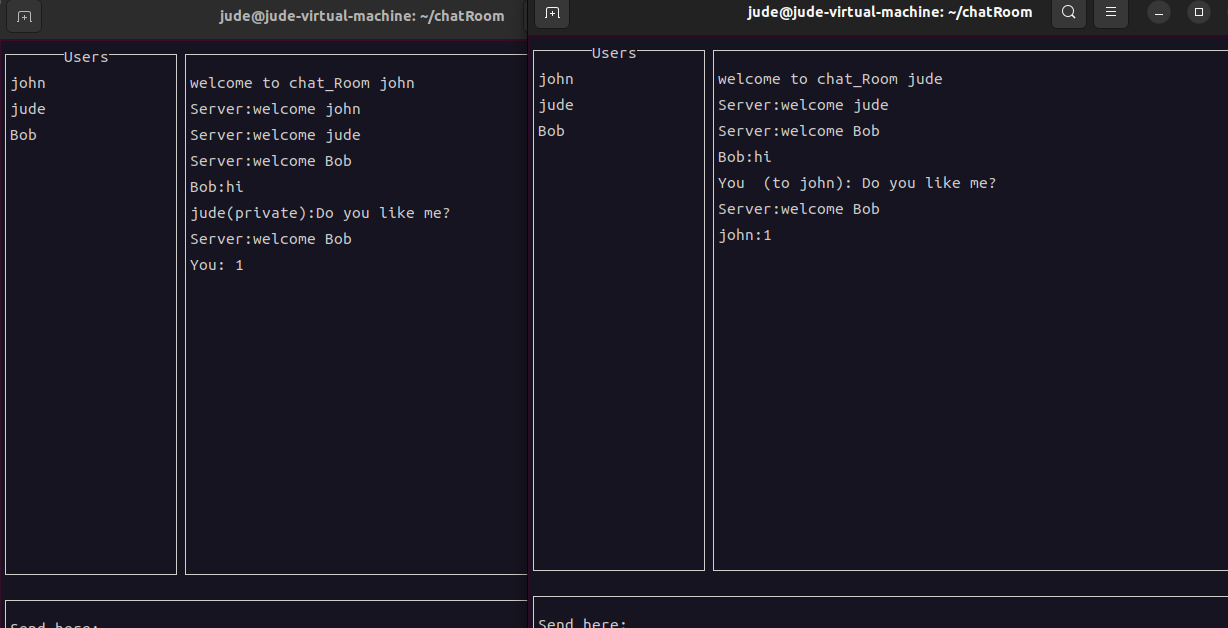
如果错误输入连接信息，则会出现错误提示，并重新输入：



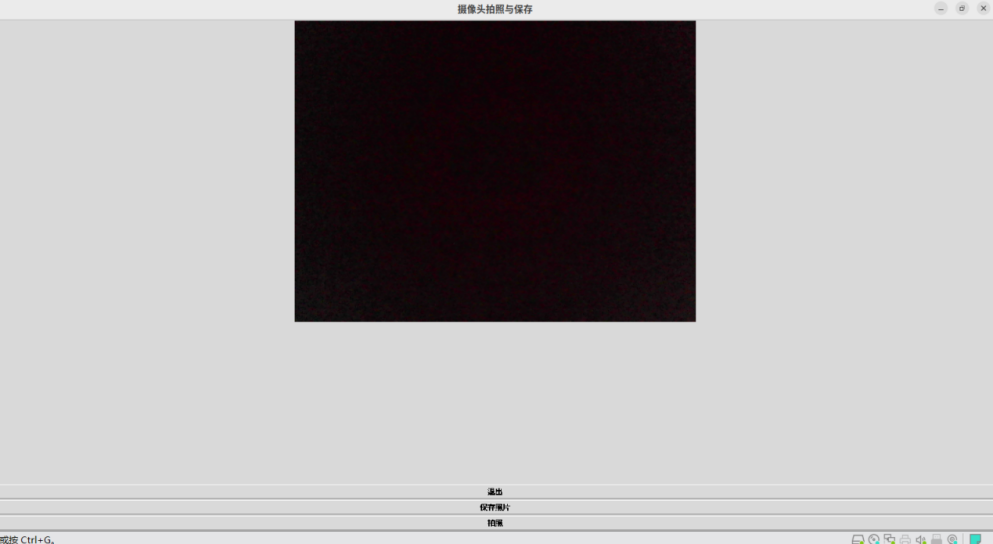
1. 在输入框中输入群聊信息，并发送



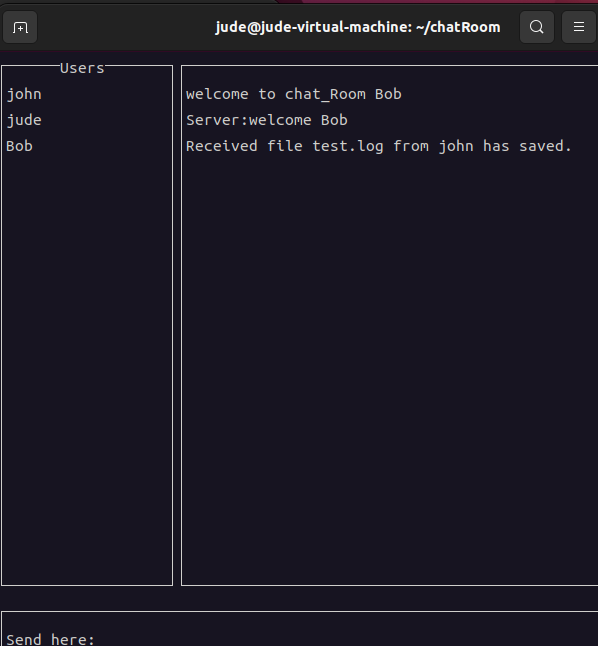
1. 输入框输入sendto，并输入私聊对象用户名，发送私聊信息



1. 输入框输入takephoto进入拍照模式：

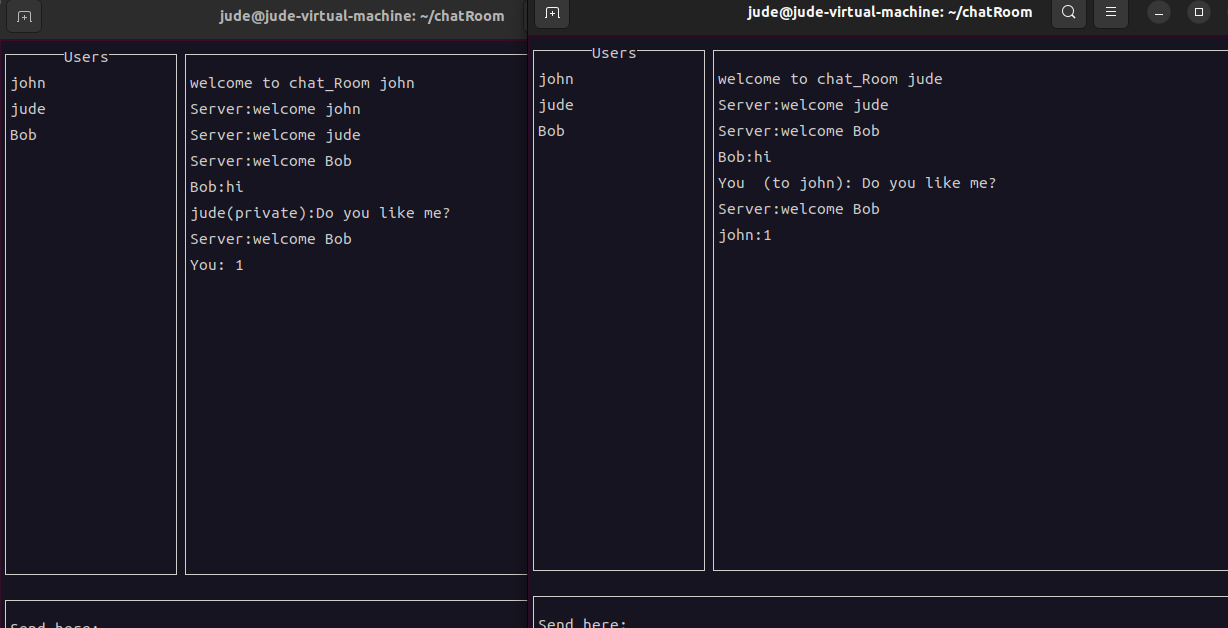


1. 输入sendfile以及要发送的用户和文件路径发送文件：

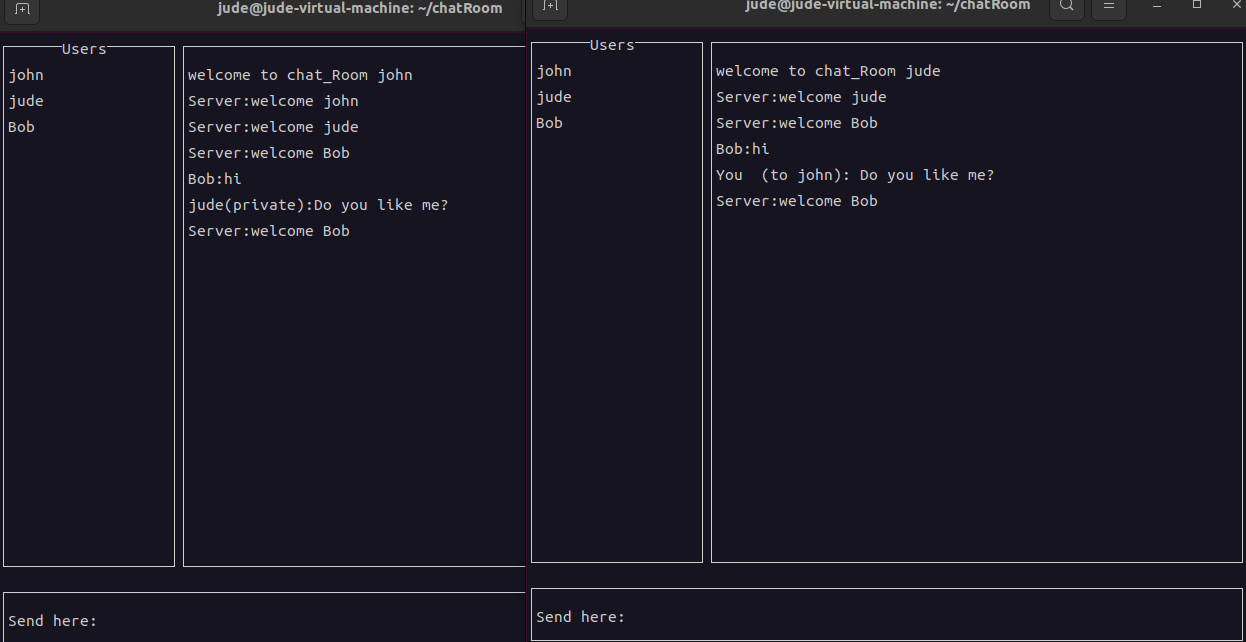


8、输入withdraw撤回信息

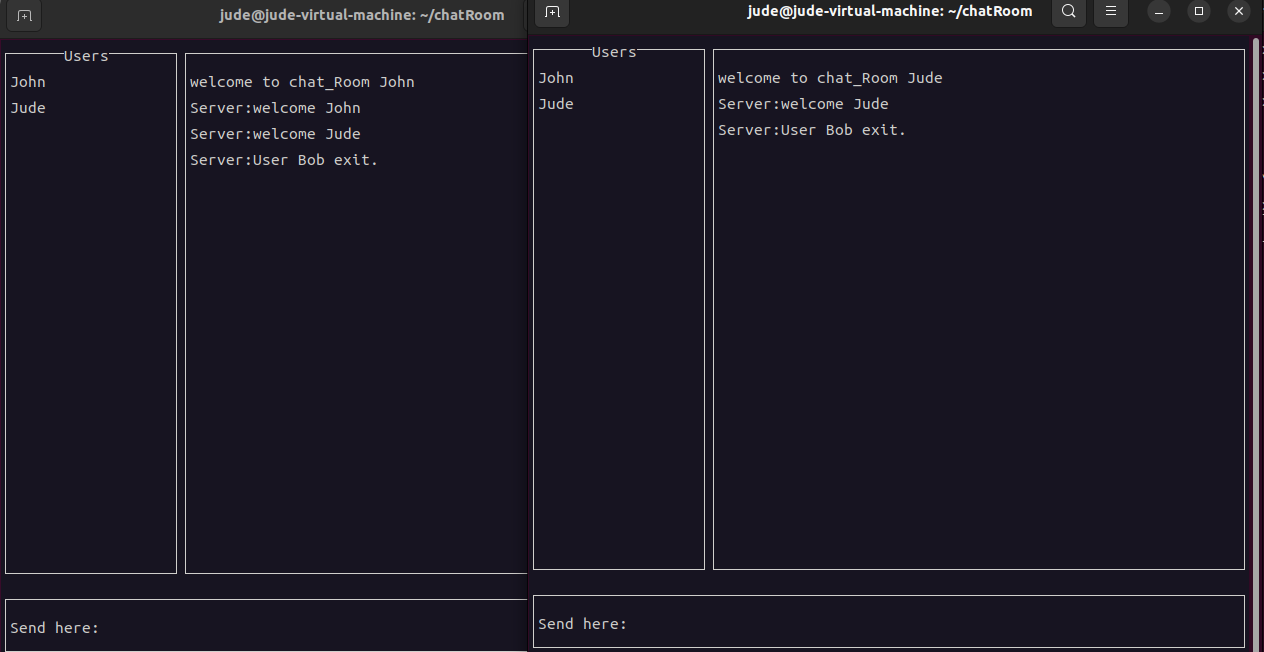
撤回前：



撤回后：



9、输入end退出



**五、系统部署与优化**

1、我们设计了两种UI，但是由于语言不同，不便于在同一处展示。使用Python可以顺利完成UI设计，简洁美观，C语言的UI代码冗长，但是设计出来的效果由于代码长度使得效果较好。

2、C语言设计的聊天系统无法实现拍照、文件传输等现代化功能，在答辩时展示了登录、群聊私聊、禁言解禁、改变个性签名等功能。最终报告由于篇幅限制仅展示Python制作的一版。两者思路可以交叉融合；若有后续使用需要可以考虑用Python语言将部分功能移植到所示系统中。

3、python设计的聊天系统可以实现私聊、群聊、拍照、文件发送、消息撤回功能、退出功能。在该系统下，通过客户端和服务器端的数据传输，上述功能在同一局域网下有效实现，并具有高度的模块集成和移植性，为后续完善功能打下基础。

**六、遇到的主要问题及解决办法**

1、报文规范设计并不复杂，但是需要缜密的思路，难点在于拆分和识别信息以及判断传输的对象。在尝试多种格式后，比如在信息前面添加头部等等，发现使用字典作为报文格式更加有利于拆分和识别。

2、两台电脑，一台充当服务器，另一台充当客户端，在局域网下建立通信，服务器端先用ifconfig命令查看IP地址，客户端运行时在./client后面加上服务器端IP地址即可

3、在vim和gedit里中文的复制粘贴会有乱码出现，使用iconv -f GBK -t UTF-8 旧文件名 -o 新文件名，即可消除乱码；

4、向其他用户发送文件时，选择文件发送后显示段错误(核心已转储)，后来发现居然是文件太大的缘故，改发较小的文件即可成功发送；

5、在设计UI界面时光标经常出现在终端的最左上角，后经代码排查发现是在执行了一段代码后未将光标的位置重新设置，后使用了move函数解决了该问题。

6、调用拍照功能时，tkinterUI和终端UI会发生冲突，导致拍照后终端UI无法正常使用其他功能且会出现乱码。采用多线程创建tkinterUI后，解决了冲突问题。

**七、心得体会**

**姜吕涛：**

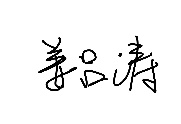
开发Linux网络聊天室项目让我深刻体会到，技术的每一步深入都伴随着新的挑战和收获。在这个过程中，Python的网络编程成为我探索的起点，通过实践，我不仅掌握了socket编程的基础，更对TCP/IP协议有了直观的认识。这种从零开始的过程，让我对如何在代码中实现客户端与服务器之间的消息传递有了深刻的理解，也体会到了编写能够在真实网络环境下运行的程序的乐趣。

面对多用户的实时交互需求，我开始涉足并发编程的领域。使用ThreadPoolExecutor管理线程池时，我感受到了并发处理带来的效率提升，每一个细微的优化都可能对用户体验产生直接影响。同时，这也是一次对Python异步编程理解的深化，从asyncio到多线程，每一种模型都有其适用场景，选择合适的工具解决问题成为了我在项目中不断思考的问题。、

我在项目中主要负责通信建立、以及群聊私聊模块的编写，在程序开发初期，我遇到一个关键的问题，就是该如何设置消息报文的格式，在上学期学过嵌入式、计算计网络，因此深知在这样一个CS架构的项目中，消息的传输尤为重要，因此消息报文的格式也是整个项目最关键的问题。在网上搜寻多个开源项目后，我选择了用字典的形式建立消息报文，让报文更加直观。而我在之前接触过java的网络编程，因此在聊天模块设计中并未碰到过多的难题。

项目的进展离不开开源社区的支持。无论是寻找解决方案，还是选择合适的第三方库，我都深受其益。在解决问题的过程中，我逐渐习惯于查阅文档、参考社区的最佳实践，甚至在GitHub上寻求帮助。开源的力量不仅在于代码的共享，更在于知识和经验的传递，这种互帮互助的文化深深吸引了我。

随着对项目的深入，我也遇到了很多bug，每一次尝试优化都是对我能力的挑战。在这个过程中，我学会了不断地测试和分析，学会了利用日志功能对程序进行测试和开发记录。

****回顾整个开发过程，我感到最宝贵的是持续学习和解决问题的能力。这次项目不仅仅是一个功能实现的过程，更是一次全面的技能提升之旅。它让我更加自信地面对未来的技术挑战，也让我坚信在不断的学习和探索中，我们能够创造出更多有价值的作品。

学生本人签字：

**王翰禹：**

在开发该款Linux系统下的聊天室的过程中我收获颇丰，我曾在上学期尝试使用Java语言编写了网络聊天室，本想着只需要将语言转换为python即可，但是在实际编写过程中还是遇到了不少困难。

在本次团队合作中我主要负责编写UI界面以及处理一些代码bug问题，由于UI界面是整个聊天室的根基，所以在我看来不能出一点的差错，但是由于课程设计的要求是在终端上实现交互界面，我对该种类型的UI并未有过多的了解，所以也是在网上一点点的搜查资料并自学完成的，在一开始编写的过程中存在超级多的bug,例如界面刷新后的光标位置无法回到输入框，用户登陆后用户列表框无法显示在线用户，又例如在获取输入框界面信息时提取文字的位置不正确等等，但在一遍遍的梳理代码逻辑和Debug之后，成功解决了这些问题。这不仅让我对curses这个库有了更深的了解，也让我对UI界面的设计过程与细节处理有了深刻的认识。

此外我也参与了文件传输与消息撤回部分的bug修复，在解决问题的过程中我掌握了使用log日志来分析问题所在并能够尝试自己编写test文件来测试某个代码模块的正确性，例如在解决文件传输无法接收到文件的问题时，我尝试在服务端接收函数内加入print打印信息，通过观察服务端输出信息来判断是客户接收端出问题还是服务接收端出问题，并成功排查了其中的小漏洞。在修复消息撤回部分的bug时，我们遇到的问题主要是输入指令之后聊天框内无法撤回消息，后经我仔细排查发现是显示函数内没有先擦除聊天框内所有内容，而是直接加入新的消息，从而导致了该问题的出现。

在此次嵌入式课程设计的过程中，我不仅学会了与团队成员分工合作，也大大提升了自己的代码编写能力，从原来的不敢写代码以及无从下手到现今的熟能生巧，总体来说对于我个人是一次很大的进步。此外，在这次设计中，我不单单只是掌握了UI界面的设计知识，也在和其他成员商讨问题的过程中掌握了不同部分的知识体系。我相信在接下来的学习中我会更加勇于去面对挑战，去用代码和创新性思维去解决每一个问题，不断学习与探索，并不断进步和提升自己。

学生本人签字：

**代仲麒：**

这次嵌入式聊天系统是一次很好的锻炼，将上学期学到的有关嵌入式的知识充分的运用到实践，而且课题任务很吸引人，写一个简易的聊天程序。这样的聊天程序让我们都联想到了QQ和微信的功能，感觉很有意思。虽说简易，上手操作起来却没有那么简单。我们从基于tcp的socket套接字通信开始，先研究代码，我刚开始想直接开启多个客户端实现群聊，发现实际运行起来并不是我们想的那样，他只是在每个客户端与服务器之间建立的单独通信，客户端与客户端之间却没有通信，于是我们便查阅资料，找到一种环形链结构，在服务器端实现存储不同客户端的信息。而在客户端，我们定义一个结构来创建我们想实现的功能列表，起初也是遇到了很多的问题，后来上网看到一些比较经典的结构例程，发现许多细节我们没有考虑到，比如在各功能之间应先定义好标志位，以便在之后的选择结构当中很方便的直接利用宏定义来饮用，避免逻辑不严谨导致的变量定义与赋值冲突，学习之后便径直运用到我们的程序当中来。

我在软件中使用了一些特殊的库，比如sqlite3这款用于嵌入式的轻量级数据库，虽然在以前学习过sql server数据库和mysql数据库，但是那些结构化查询语句在linux里并不能直接用，我装的虚拟机ubuntu系统里缺少sqlite3.h库，安装后经过很长时间的摸索才对它有了初步的了解。在程序编写过程中也遇到过许多问题，改bug是最头疼的事，我通常改bug都是先查看其所在的位置，找到后先尝试修改重新编译，或者上网查询相应的错误原因再去修改，本次课设在ubuntu里使用中文经常出现乱码，应该是因为语言库的原因，

这期间我们收获很大，发现问题，解决问题的能的得到了提升，过程中也并不枯燥，总的来说是一次很愉快的经历。

学生本人签字：

**雷文强：**

在完成嵌入式系统设计与开发课程设计的过程中，我学到了很多，过程虽然艰难，但是有上个学期的知识学习，也就是linux网络编程的相关知识，我们对完整的TCP通信进行了复习，为我们的网络聊天室的设计和实现打下了扎实的理论基础。

在完成过程中我们遇到了很多的问题，因为想实现的功能过多，使用c语言进度滞后，功能模块无法耦合，这一部分最后交给代仲麒完成。而我们及时转换思路使用python语言进行编写，虽然python语言比较便利，但是其中各种的函数库也是让人眼花缭乱，还好我们都选修过python语言程序设计这门课，所以最基本的还是没有问题。我们在学习了如何使用python进行TCP通信后，进行功能的编写，因为我们已经对c语言中TCP通信比较熟悉，所以python也是很快上手，最后进行各种功能的编写也是没问题了，虽然最后比较仓促，但是还好是完成了聊天室的基本功能。

在完成课程设计的过程中，我在我的队友身上也是学到了很多的专业知识，c语言和python语言的熟练度也是提高了不少。本次课程设计完成过程中最主要的是队友的帮助，因为我的知识储备比较少，所以进行学习的过程中与队友进行交流也是很多，提问题也多，但是最后的结果还是勉强完成了功能的实现。

很高兴进行本次嵌入式课程设计，我学到了很多，并且也感谢老师与同学对我的帮助，让我顺利完成了本次课程。

学生本人签字：

**董磊：**

在最近的一周里，我们进行了嵌入式系统设计与开发的课程设计，这是一次极具挑战性和收获的学习经历。我们利用Python实现一个具有图形化界面的局域网聊天室。通过这个过程，我不仅对嵌入式系统设计与开发有了更深入的了解，也对python库函数强大的功能所震惊，尤其是tkinter库，颠覆了我曾经使用C语言编写代码的稚嫩手法。

首先，这次课程设计让我深刻体会到了理论知识与实践操作之间的紧密联系。上学期，我们学习了Linux网络编程的相关知识，掌握了TCP通信的基本原理和过程。这些知识为我们这次的网络聊天室设计和实现提供了坚实的基础。然而，理论知识只是基础，真正将其应用到实际操作中，还需要我们不断地尝试、调试和优化。在开发过程中，我们遇到了许多预料之外的问题，如网络通信的稳定性、图形界面的布局和交互等。通过不断地查阅资料和尝试，我们逐渐解决了这些问题，最终成功实现了聊天室的功能。在我负责的拍照功能中，我对于解决问题有了更加深入的理解。以前我遇到bug和报错，总是简单的依赖于csdn，企图通过其他人的途径来解决我遇到的问题。但这次的我们搭建的聊天室框架是独立自主的，所以很难找到完全相似的情况。这在我进行优化终端UI和tkinterUI跳转冲突时，卡壳了很久。后来我认真思考，学习上学期嵌入式教材的基本原理，终于在不断尝试下，找到了利用多线程解决冲突问题的办法。最终成功实现了拍照功能。

其次，这次课程设计也让我深刻认识到了团队合作的重要性。在初期，由于我们对任务的理解不够深入，分工不够明确，导致进度一度滞后。然而，通过及时的沟通和讨论，我们逐渐明确了每个人的职责和任务，并及时止损。在后续的开发过程中，我们相互支持、相互学习，共同攻克了一个又一个难题。这次经历让我意识到，一个人的力量是有限的，而团队的力量是无穷的。只有充分发挥每个人的优势，形成合力，才能取得更好的成果。

同时，这次课程设计也让我认识到了自己在学习和实践中的不足之处。我发现自己在很多方面的知识储备还不够，需要在未来的学习中不断加强和拓展。同时由于以前更多的是看代码，很少实操，导致我的编程能力还很薄弱，所以我在进行模块实现的过程中进度很慢。通过这次的课程设计，不仅提升了我的专业知识，更提升了我的代码能力，也增加了我对嵌入式系统的兴趣。在未来的学习中，我会更加努力地学习新知识、掌握新技能，不断提升自己的能力和水平。

学生本人签字：

日期：2024年3月12日

**主讲教师评分及评语：**