//////////////////////////////lab2_简易聊天室(广播) PB19030800 陈磊

1. 实验内容

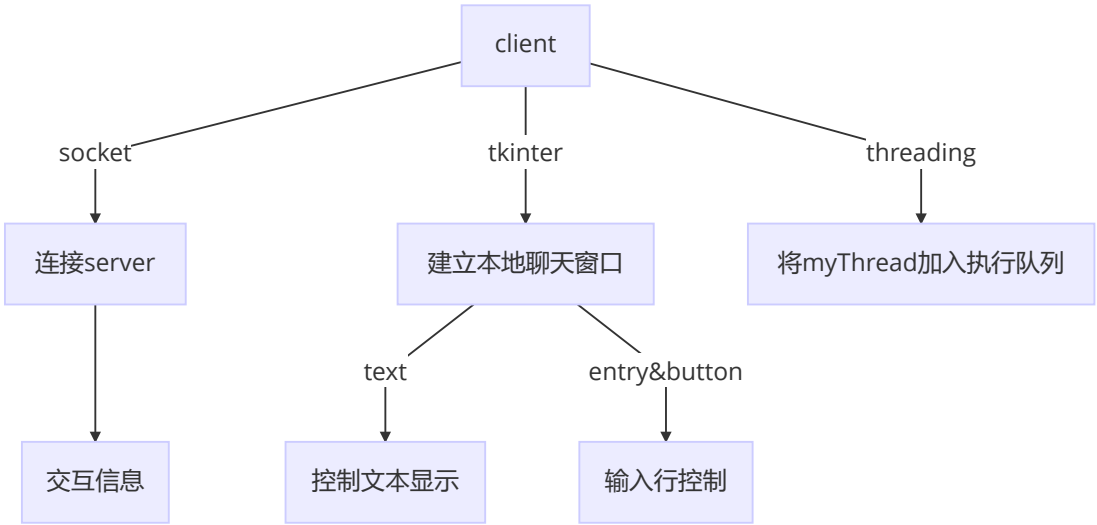
   本实验利用TCP协议实现一个聊天室，客户端显示聊天窗口，服务器仅仅在终端显示当前用户数

2. 参考博客：http://t.csdn.cn/Vzmir

   运行环境：windows11+python(需要安装Anaconda/命令行pip install下载所需Python库)

3. 代码分析：

   client.py代码相对简单，从模块角度介绍：

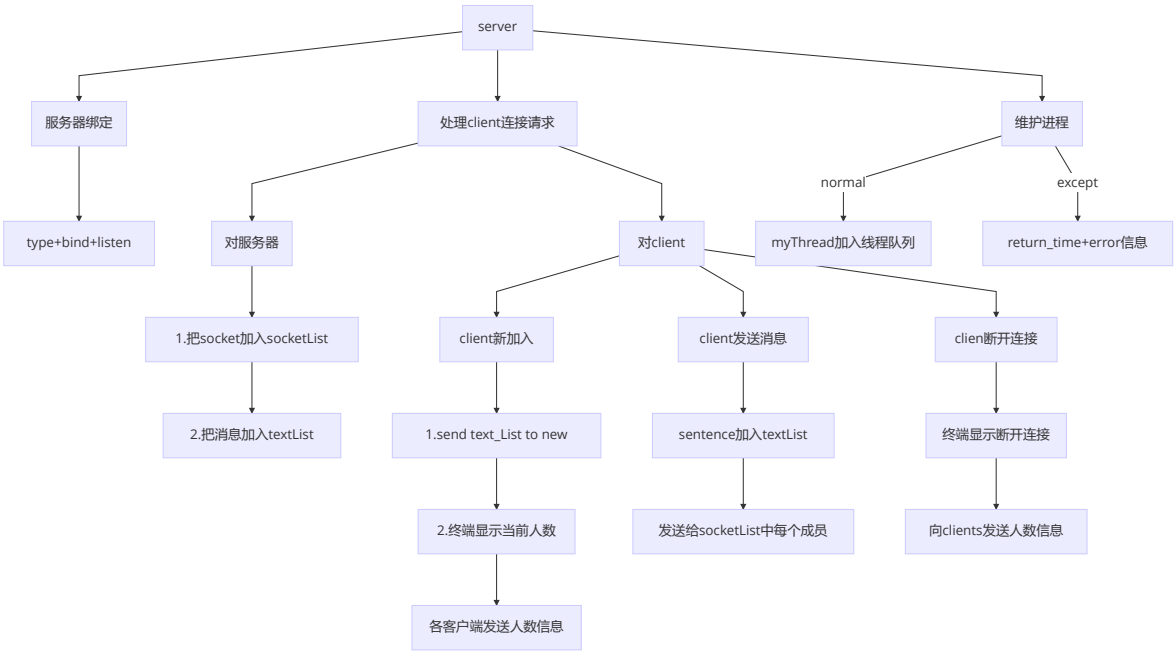



```
import os
import time
import threading
from socket import *
from tkinter import *

serverName = gethostbyname(gethostname())#服务器外网IP
serverPort = 12000

root = Tk() #创建窗口供聊天
root.title('聊天室')

text = Text(root,width = 100,height = 20) #tkinter文本控件，显示多行文本

scroll = Scrollbar(root)
scroll.pack(side = RIGHT,fill = Y)
scroll.config(command = text.yview)

text.config(yscrollcommand = scroll.set)
```

```
text.pack()

entry = Entry(root,bd = 5,width = 70)   #输入窗格，显示简单的文本内容
entry.pack(side = LEFT)                 #窗格显示在左侧
entry2 = Entry(root,bd = 5,width = 20)  #第二个窗格

#客户端添上套接字，并尝试与服务器建立连接(这里服务器为本机)
clientSocket = socket(AF_INET, SOCK_STREAM)
print('正在连接...')
clientSocket.connect((serverName,serverPort))
print('连接成功!')

msg = clientSocket.recv(2048)#先接收历史记录

if msg.decode() != 'NO_MESSAGE':#有历史记录时
    text.insert(END,msg.decode() + '\n----以上是历史记录----\n')
text.see(END)

#返回本机ip地址
def get_ip_address():#为windows系统，在本机只有一个ip地址时可用
#    mac = uuid.UUID(int = uuid.getnode()).hex[-12:]
#    return ':'.join([mac[e:e+2] for e in range(0,11,2)])
#    return socket.gethostbyname(socket.gethostname()),
    return gethostbyname(gethostname())

def timemark():                         #显示时间，可用于完成第一个实验
    timestamp = int(time.time())    #获得当前时区时间戳
    timestr = time.strftime('%Y-%m-%d %H:%M:%S',time.localtime(timestamp))  #将当前时间转换为
字符串
    return '[' + timestr + ']'

def send():
    #如果第二个窗格为空，文本控件显示"请输入用户名"
    if entry2.get().replace(' ','') == '':
        text.config(state = 'normal')
        text.insert(END,'请输入用户名! \n')
        text.see(END)
        text.config(state = 'disabled')
        return
    #如果第一个窗格为空，文本控件显示"发送内容为空"
    if entry.get() == '':
        text.config(state = 'normal')
        text.insert(END,'发送内容不能为空! \n')
        text.see(END)
        text.config(state = 'disabled')
    #否则文本控件输出以格式"时间+ip地址+用户名+输入板消息"
    else:
        clientSocket.send((timemark() + ' ' + get_ip_address() + ' - ' +
                           entry2.get() + '\n    ' + entry.get()).encode())
        entry.delete(0,'end')

#在输入栏右侧显示发送按钮

button = Button(root,text = 'SEND',command = send)
```

```
button.pack(side = RIGHT)

entry2.pack(side = RIGHT)
#定义线程类myTread
class myThread(threading.Thread):
    def __init__(self):
        threading.Thread.__init__(self)
    def run(self):
        while True:
            msg = clientSocket.recv(2048)
            text.config(state = 'normal')
            text.insert(END,msg.decode() + '\n')
            text.see(END)
            text.config(state = 'disabled')
textThread = myThread()
#myThread
textThread.start()
#循环运行聊天窗口
root.mainloop()
```

server.py主要使用socket模块，从函数角度介绍：



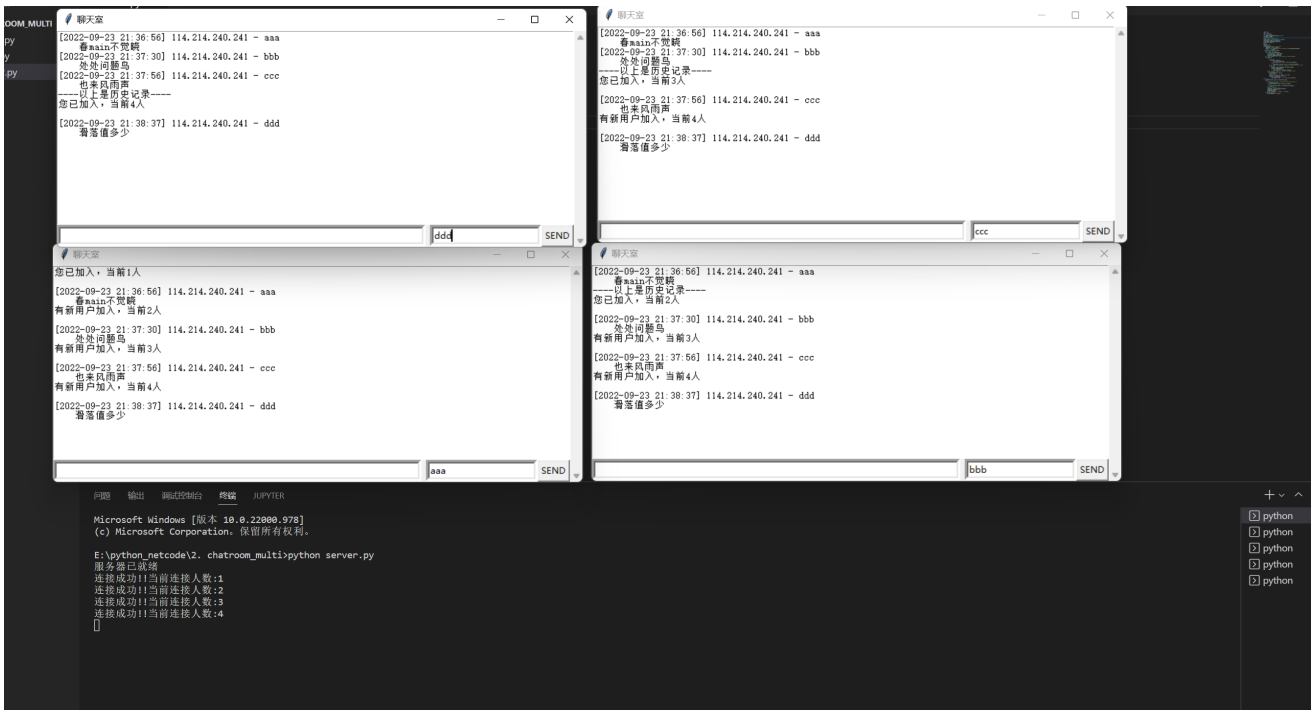

```
#server.py
import os
import time
import threading
from socket import *
IP = gethostbyname(gethostname()) #取得本机IP
serverPort = 12000

MAX_CONNECTION = 10
```

```
#将服务器绑定到本机端口并listen
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((IP,serverPort))
serverSocket.listen(MAX_CONNECTION)
print('服务器已就绪')
cnt = 0          #当前连接数
textList = []    #文本列表
socketList = [] #连接客户端套接字列表
def timemark(): #产生当前的时间字符串
    timestamp = int(time.time())     #获得当前时区时间戳
    timestr = time.strftime('%Y-%m-%d %H:%M:%S',time.localtime(timestamp))  #转化为时间字符
串
    return '[' + timestr + ']'
#线程类myThread
class myThread(threading.Thread):
    #将当前连接加入socketList
    def __init__(self,connection):
        threading.Thread.__init__(self)
        self.connection = connection
        socketList.append(connection)
    #在客户端连接成功后，每有一个新客户连入服务器，向各客户端报告
    def run(self):
        try:
            global cnt
            for sock in socketList:
                if sock == self.connection:
                    sock.send(('您已加入，当前%d人'%(cnt)).encode())
                else:
                    sock.send(('有新用户加入，当前%d人'%(cnt)).encode())
            #每次客户向服务器发送信息，服务器将信息加入textList,并广播到所有用户
            while True:
                sentence = self.connection.recv(2048).decode()
                textList.append(sentence)
                for sock in socketList:
                    sock.send(('\n' + sentence).encode())
        #每有一个连接断开，先后向服务器终端、所有用户报告当前人数
        except Exception as e:
            print(timemark() + repr(e))
            cnt = cnt - 1
            print('连接异常断开!当前人数:' + str(cnt))
            socketList.remove(self.connection)
            for sock in socketList:
                sock.send(('有用户退出，当前%d人'%(cnt)).encode())
while True:
    connectionSocket ,addr= serverSocket.accept()
    try:
        #如果textList==0，向新增连接发送NO_MESSAGE，否则发送textList
        if len(textList) == 0:
            connectionSocket.send('NO_MESSAGE'.encode())
        else:
            connectionSocket.send('\n'.join(textList).encode())
        cnt = cnt + 1

        #将对新增连接的处理线程加入正在运行的线程队列
```

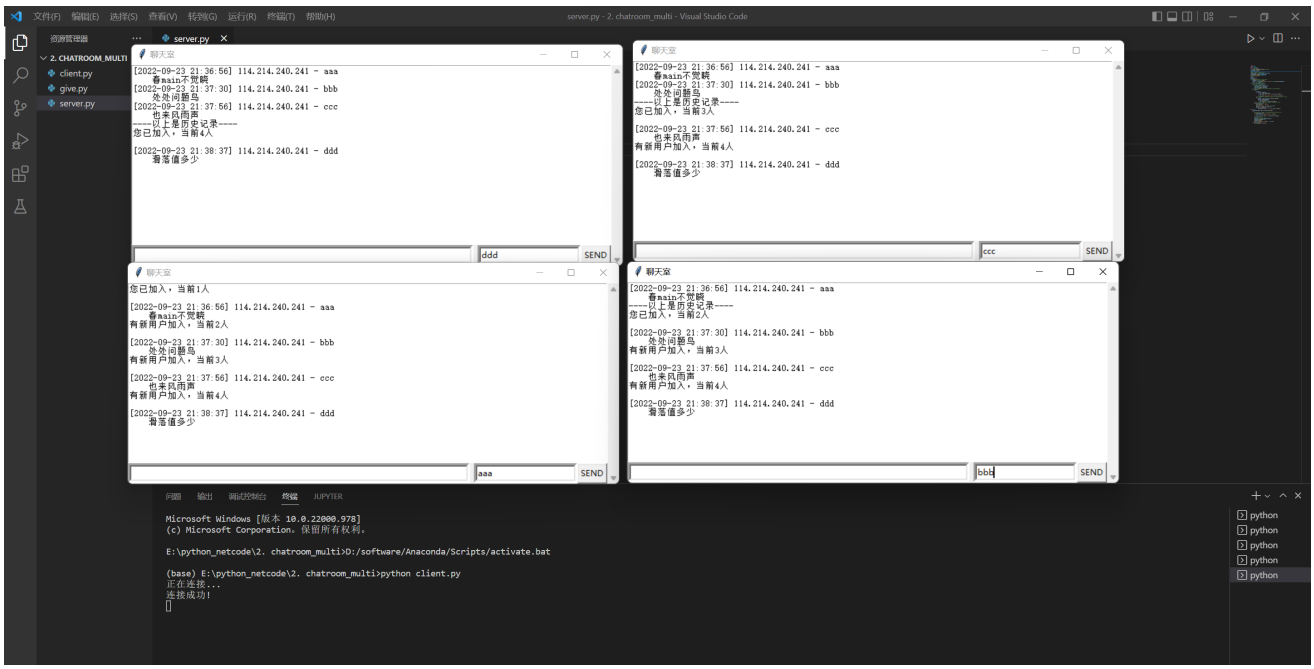
```
            newThread = myThread(connectionSocket)
            newThread.setDaemon(True)
            newThread.start()
            print('连接成功!!当前连接人数:' + str(cnt))
        #如果出现异常，返回当前时间与异常信息
        except Exception as e:
            print(timemark() + repr(e))
```

4. 实验截图

服务器终端：



client终端：



5. 关于简易聊天室(单播)的想法

对client：sentence中加一段目的ip地址，并在前后用特殊符号标记，收到历史记录时首先显示在tk的text中

对server：对接收到的sentence解析，通过对特殊符号的判断解析sentence目的地target，而后不是send to everyone in socketList，而是send to target； 同时不用textList（for all）,而是为每一个client分配一个历史记录表，单独记录单发给每个client的信息，并在client连接时首先发送，