实验6——基于tcp类型socket的多人聊天室

21120860 刘虚谷

**1.概述**

为了实现客户端聊天室GUI界面的效率，我使用了python语言。

聊天室大致的设计思想是：设计一个服务端，用于与客户端建立链接，转发某个客户端发送的消息给其余已经和服务端建立的客户端，在客户端则使用GUI界面优化，可以收到其余客户端发来的消息，还能收到系统消息，即服务端的消息。

**2.以下是服务端代码的解析**：

相关库的导入以及相关变量的设置

from socket import \*

#导入socket库，这样才可以使用socket

from sqlite3 import connect

#导入sqlite库，后面要使用数据库

import threading

#多线程

from datetime import \*

#获取绝对时间

ISOTIMEFORMAT = '%Y-%m-%d %H:%M:%S'

#时间格式声明，用于后面的记录系统时间

IP = '127.1.1.1'

PORT = 30000

#设置ip地址与端口号

user\_list = []

socket\_list = []

#用户列表与套接字列表，用于后面给每一个套接字发消息

相关不断调用的函数的编写

#将聊天记录放在serverlog.txt文件中

try:

with open('serverlog.txt', 'a+') as serverlog:

curtime = datetime.now().strftime(ISOTIMEFORMAT)

serverlog.write('\n\n-----------服务器打开时间：'+str(curtime)+'，开始记录聊天-----------\n')

except:

print('ERROR!')

s = socket()

s.bind((IP, PORT))

s.listen()

#创建套接字并绑定，进入listen状态

#用于服务端接受客户端发送的消息，发送移除则移除相关用户

def read\_client(s, nickname):

try:

return s.recv(2048).decode('utf-8')

#获取此套接字（用户）发送的消息

except:

#一旦断开连接则记录log以及向其他套接字发送相关信息

curtime = datetime.now().strftime(ISOTIMEFORMAT) #获取当前时间

print(curtime)

print(nickname + ' 离开了聊天室!')

with open('serverlog.txt', 'a+') as serverlog: #log记录

serverlog.write(str(curtime) + ' ' + nickname + ' 离开了聊天室!\n')

socket\_list.remove(s)

user\_list.remove(nickname) #从列表移除已经离开的用户

for client in socket\_list:

client.send(('系统消息：'+ nickname + ' 离开了聊天室!').encode('utf-8'))

#其他套接字通知（即通知其他聊天窗口）

#用于向除与s套接字的其余客户端发送消息

def socket\_target(s, nickname):

try:

s.send((','.join(user\_list)).encode('utf-8'))

#将用户列表送给各个套接字，用逗号隔开，便于客户端更新本地用户列表

while True:

content = read\_client(s, nickname)

#获取用户发送的消息

if content is None:

break

else:

curtime = datetime.now().strftime(ISOTIMEFORMAT) #系统时间打印

print(curtime)

print(nickname+'说：'+content)

with open('serverlog.txt', 'a+') as serverlog: #log记录

serverlog.write(str(curtime) + ' ' + nickname + '说：' + content + '\n')

for client in socket\_list: #向其他套接字通知

client.send((nickname + '说:'+ content).encode('utf-8'))

except:

print('Error!')

主进程与子线程的编写与应用

while True:

#不断接受新的套接字进来，实现“多人”聊天室

conn, addr = s.accept()

#获取套接字与此套接字的地址

socket\_list.append(conn) #套接字列表更新

nickname = conn.recv(2048).decode('utf-8') #接受昵称

if nickname in user\_list:

#昵称查重，相同则在后面加上数字

i = 1

while True:

if nickname+str(i) in user\_list:

i = i + 1

else:

nickname = nickname + str(i)

break

user\_list.append(nickname) #用户列表更新，加入新用户（新的套接字）

curtime = datetime.now().strftime(ISOTIMEFORMAT)

print(curtime)

print(nickname + ' 进入了聊天室!')

with open('serverlog.txt', 'a+') as serverlog: #log记录

serverlog.write(str(curtime) + ' ' + nickname + ' 进入了聊天室!\n')

for client in socket\_list[0:len(socket\_list)-1]: #其他套接字通知

client.send(('系统消息：'+ nickname + ' 进入了聊天室！').encode('utf-8'))

#加入线程中跑，加入函数为socket\_target，参数为conn,nickname，这样才能实现实时交互

threading.Thread(target=socket\_target, args=(conn,nickname,)).start()

3.以下是客户端代码的解析：

相关库的导入以及相关变量的设置

from tkinter import \*

#导入tkinter库，用于设计GUI界面

from datetime import \*

from socket import \*

import threading

import tkinter

import tkinter.messagebox

#导入messagebox模块，用于后面的错误信息显示

from tkinter.scrolledtext import ScrolledText

#导入scrolledtext模块，用于客户端消息的消息

ISOTIMEFORMAT = '%Y-%m-%d %H:%M:%S'

s = socket()

相关实现窗口的代码的编写

#登录窗口

def Login\_gui\_run():

root = Tk()

root.title("小刘聊天系统·登录") #窗口标题

frm = Frame(root)

root.geometry('300x150') #窗口大小

nickname = StringVar() #昵称变量

def login\_in(): #登录函数（检查用户名是否为空，以及长度）

name = nickname.get() #考虑长度是因为用户列表那边可能无法完整显示

if not name:

tkinter.messagebox.showwarning('Warning', message='用户名为空！')

elif len(name)>10:

tkinter.messagebox.showwarning('Warning', message='用户名过长！最多为十个字符！')

else:

root.destroy() #窗口的销毁

s.connect(('127.1.1.1', 30000)) #建立连接

s.send(nickname.get().encode('utf-8')) #传递用户昵称

Chat\_gui\_run() #打开聊天窗口

#登录按钮、输入提示标签、输入框

Button(root, text = "登录", command = login\_in, width = 8, height = 1).place(x=100, y=90, width=100, height=35)

#tkinter库的按钮组件

Label(root, text='请输入昵称', font=('Fangsong',12)).place(x=10, y=20, height=50, width=80)

#tkinter库的标签组件

Entry(root, textvariable = nickname, font=('Fangsong', 11)).place(x=100, y=30, height=30, width=180)

#tkinter库的输入框组件

root.mainloop() #进入窗口主循环

#聊天窗口

def Chat\_gui\_run():

window = Tk()

window.maxsize(650, 400) #设置相同的最大最小尺寸，将窗口大小固定

window.minsize(650, 400)

var1 = StringVar()

user\_list = []

user\_list = s.recv(2048).decode('utf-8').split(',')

#从服务器端获取当前用户列表，与服务端的socket\_target函数功能对应

user\_list.insert(0, '------当前用户列表------')

nickname = user\_list[len(user\_list)-1] #获取正式昵称，经过了服务器端的查重修改

window.title("聊天室--"+nickname) #设置窗口标题，体现用户专属窗口（不是）

var1.set(user\_list) #用户列表文本设置

listbox1 = Listbox(window, listvariable=var1) #用户列表，使用Listbox组件

listbox1.place(x=510, y=0, width=140, height=300)

listbox = ScrolledText(window) #聊天信息窗口，使用ScrolledText组件制作

listbox.place(x=5, y=0, width=500, height=300)

#接收服务器发来的消息并显示到聊天信息窗口上，同时用于用户列表更新

def read\_server(s):

while True:

content = s.recv(2048).decode('utf-8')

#接收服务器端发来的消息

curtime = datetime.now().strftime(ISOTIMEFORMAT)

#获取当前系统时间

listbox.insert(tkinter.END, curtime)

#聊天信息窗口显示（打印）

listbox.insert(tkinter.END, '\n'+content+'\n\n')

listbox.see(tkinter.END)

#see()是ScrolledText组件的方法，可以自动定位到结尾，否则只有消息在涨，窗口拖动条不动

listbox.update() #更新聊天信息窗口，显示新的信息

#判断是否为服务端消息，如果是，则用发来的消息更新客户端本地用户列表

if content[0:5]=='系统消息：':

if content[content.find(' ')+1 : content.find(' ')+3]=='进入':

user\_list.append(content[5:content.find(' ')])

var1.set(user\_list)

if content[content.find(' ')+1 : content.find(' ')+3]=='离开':

user\_list.remove(content[5:content.find(' ')])

var1.set(user\_list)

threading.Thread(target = read\_server, args = (s,)).start()

#创造子线程，用于一直接受服务端转发的消息

var2 = StringVar() #聊天输入口变量

var2.set('')

entryInput = Entry(window, width = 140, textvariable=var2)

entryInput.place(x=5, y=305, width = 600, height = 95)

#发送按钮对应的函数，即发送信息函数

def sendtext():

line = var2.get()

s.send(line.encode('utf-8'))

var2.set('') #发送完毕清空聊天输入口

sendButton = Button(window, text = '发 送', font=('Fangsong', 18), bg = 'white', command=sendtext) #发送按钮组件

sendButton.place(x=500, y=305, width = 150, height = 95)

window.mainloop() #进入window窗口的主循环

Login\_gui\_run() #主线程首先进入登录界面

1. **实验中遇到的问题**

**在实验中，为了实现客户端之间实时交互，需要使用多线程，多进程等模块，但由于对多线程和多进程的概念及运行方式不清，这导致了代码编写的堵塞。**

**在网上查阅了相关的资料，理解了多线程和多进程的区别与相似之处，而这者分别对应python的thread模块和process模块，在使用process模块时会因为函数定义的方式，列如在主进程中定义，导致在子进程中无法调用，但是使用thread模块代替process模块，并将UDP套接字，改为TCP套接字后，程序就能顺利运行了。**