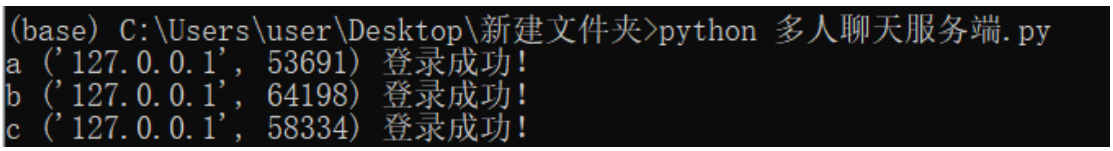
计算机网络项目——开发文档

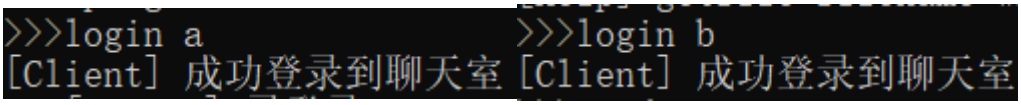
该程序由客户端与服务器构成，使用 UDP 服务，服务器端绑定本地 IP 和端口，客户端由系统随机选择端口。实现了群发、私发、点对点文件互传功能。客户端自建了一个类继承了 Cmd 模块，使用自定义的命令 command 进行操作，调用相应的 do_command 方法。使用 json 模块进行消息的封装序列化，在接收方进行解析。

测试结果：

服务器开启+监控用户登录：

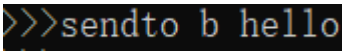


多用户登录：

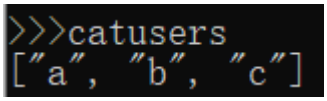


聊天功能：

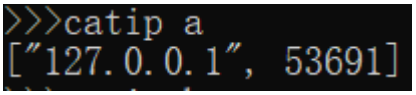
私聊功能：



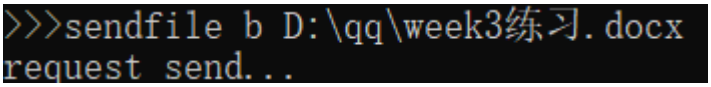
用户列表查看：



单个用户 ip 查看：



发送文件：



接收文件：

```
getfile week3练习.docx a yes
you agree to reveive the file( week3练习.docx ) from a
>>>week3练习.docx is received.
```

服务器监控文件传输：

```
a request to send file to b
b agree to receive file from a
```

开发中遇到的问题：

1.文件传输问题：

在实现文件发送的功能时，当我选择一个位于本机 C 盘的文件进行发送时，系统会提示我 **permission denied**，也就是没有权限读写这个文档。

解决这个问题，我首先查看该文件是否处于打开的状态，如果文件处于打开状态，其他应用就没有权限来继续读取这个文件。在确定这个文件没有打开后，打开这个文件的属性配置，

确定该文件第二个权限开启：该文件内容可以被索引，如果没有开启这个权限，则不能开启这个发送文件线程。

```python
def __send_file_thread(self):
    """
    发送文件线程
    :param message: 消息内容
    """
    filecount = 0
    print('[system]', 'sending the file...')
    while filecount * 1024 <= self.sendfilesize:
        self.__socket.sendto(
            self.file_send.read(1024), self.fileto_addr)
        filecount += 1
    self.file_send.close()
    self.sendfile = False
    print('[system]', 'the file is sended.')
```

因为本机的 python 和 Vscode 安装在 D 盘，所以在访问 C 盘的文件需要管理员权限。因此在传输文件时，将所要传输的文件转移到 C 盘以外的盘中，就可以正常传输文件了。

2.私聊功能的实现

刚开始没有想到如何解决私聊的方法，因为私聊发送消息需要双方看到，但是私聊之外的用户不能看到私发消息的内容和动作。之后想到的解决方法是从聊天消息中做出改变。用户发起私聊的动作用不同于普通聊天的字符规定，然后在服务器识别到之后，就在服务器截断这一信息，然后再由服务器转发给私聊的双方，这样就可以做到除私聊双方之外的用户看不到私聊信息的要求。

发送消息：

```
def __send_broadcast_message_thread(self, message):
    """
    发送广播消息线程
    :param message: 消息内容
    """
    self.__socket.sendto(json.dumps({
        'type': 'broadcast',
        'nickname': self.__nickname,
        'message': message,
    }).encode(), self.__host)
```

```
def do_send(self, args):
    """
    发送消息
    :param args: 参数
    """
    if self.__nickname is None:
        print('请先登录! login nickname')
        return
    message = args
    # 开启子线程用于发送数据
    thread = threading.Thread(
        target=self.__send_broadcast_message_thread, args=(message, ))
    thread.setDaemon(True)
    thread.start()
```

私聊消息：

```
def do_sendto(self, args):
    """
    发送私发消息
    :param args: 参数
    """
    if self.__nickname is None:
        print('请先登录! login nickname')
        return
    who = args.split(' ')[0]
    message = args.split(' ')[1]
    # # 显示自己发送的消息
    # print('[' + str(self.__nickname) + '(' + str(self.__id) + ')' + ']', message)
    # 开启子线程用于发送数据
    thread = threading.Thread(
        target=self.__send_whisper_message_thread, args=(who, message))
    thread.setDaemon(True)
    thread.start()
```

```
# 私发消息
elif js['type'] == 'sendto':
    who = js['who']
    nickname = js['nickname']
    message = js['message']
    # 检查用户是否存在
    if who not in user_list:
        obj.sendto(json.dumps(
            {
                'type': 'message',
                'message': who + ' not exist or not online.please try later.'
            }).encode(),
            client_address)
    else:
        obj.sendto(json.dumps(
            {
                'type': 'message',
                'message': nickname + ' whisper to you: ' + message
            }).encode(),
            conn_list[user_list.index(who)])
```