# 设计文档

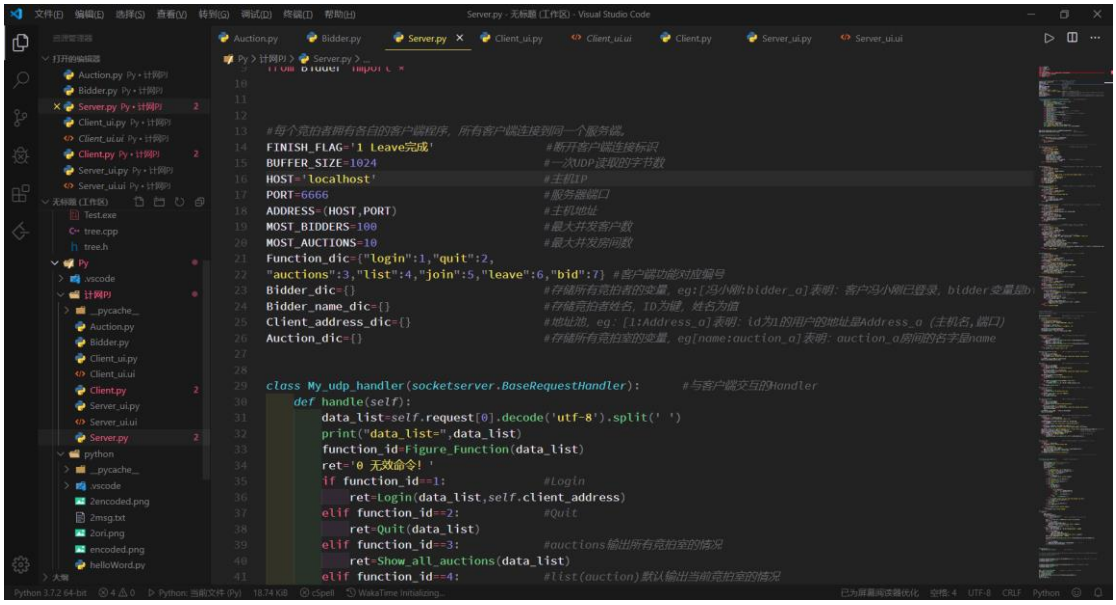
13300240003

王子优

# 技术栈：

Vscode+Python+Pyqt5

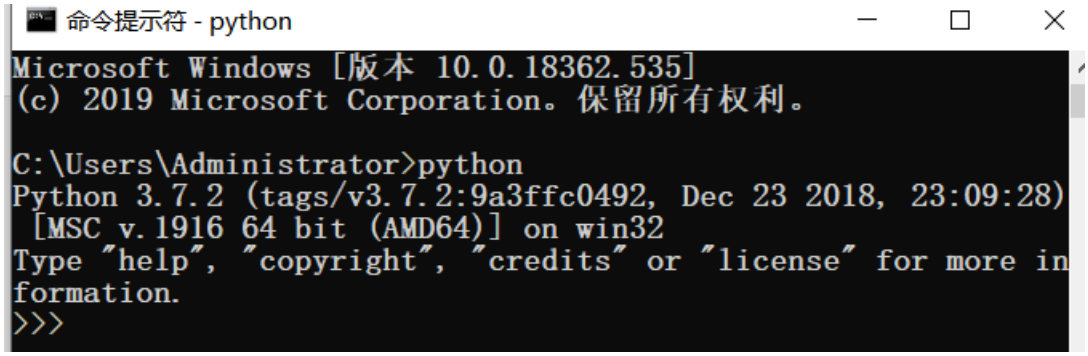
Vscode 截图：



```
#每个竞拍者拥有各自的客户端程序，所有客户端连接到同一个服务端。
FINISH_FLAG='1 Leave完成'                   #断开客户端连接标识
BUFFER_SIZE=1024                            #一次UDP读取的字节数
HOST='localhost'                            #主机IP
PORT=6666                                   #服务器端口
ADDRESS=(HOST,PORT)                         #主机地址
MOST_BIDDERS=100                            #最大并发客户数
MOST_AUCTIONS=10                            #最大并发房间数
Function_dic={"login":1,"quit":2,
"auctions":3,"list":4,"join":5,"leave":6,"bid":7} #客户端功能对应编号
Bidder_dic={}                               #存储所有竞拍者的变量，eg:['冯小刚':bidder_a]表明：客户冯小刚已登录，bidder变量是b
Bidder_name_dic={}                          #存储竞拍者姓名，ID为键，姓名为值
Client_address_dic={}                       #地址池，eg：[1:Address_a]表明：id为1的用户的地址是Address_a（主机名,端口）
Auction_dic={}                              #存储所有竞拍室的变量，eg[name:auction_a]表明：auction_a房间的名字是name


class My_udp_handler(socketserver.BaseRequestHandler):      #与客户端交互的Handler
    def handle(self):
        data_list=self.request[0].decode('utf-8').split(' ')
        print("data_list=",data_list)
        function_id=Figure_Function(data_list)
        ret='0 无效命令！'
        if function_id==1:                  #Login
            ret=Login(data_list,self.client_address)
        elif function_id==2:                #Quit
            ret=Quit(data_list)
        elif function_id==3:                #auctions输出所有竞拍室的情况
            ret=Show_all_auctions(data_list)
        elif function_id==4:                #list(auction)默认输出当前竞拍室的情况
```

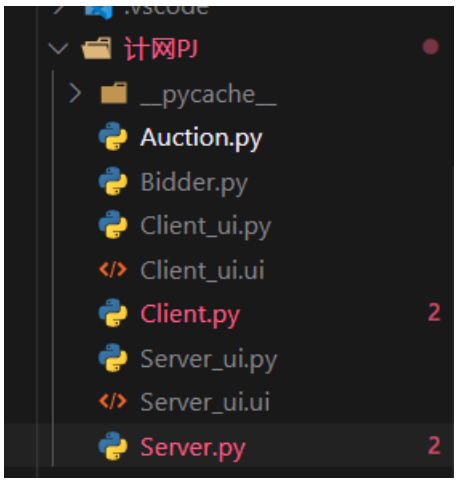
Python 版本：



```
命令提示符 - python
Microsoft Windows [版本 10.0.18362.535]
(c) 2019 Microsoft Corporation。保留所有权利。

C:\Users\Administrator>python
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28)
 [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more in
formation.
>>>
```

工程截图：




# 系统结构：

auction.py 实现 auction 类，bidder.py 实现 bidder 类。
Client_ui.ui 和 Client_ui.py 实现客户端的 GUI。
Server_ui.ui 和 Server_ui.py 实现服务端的 GUI。
Client.py 实现客户端的功能，Server.py 实现服务端的功能。
GUI 设计利用 pyqt5 库实现。
多线程利用 threading 库实现。

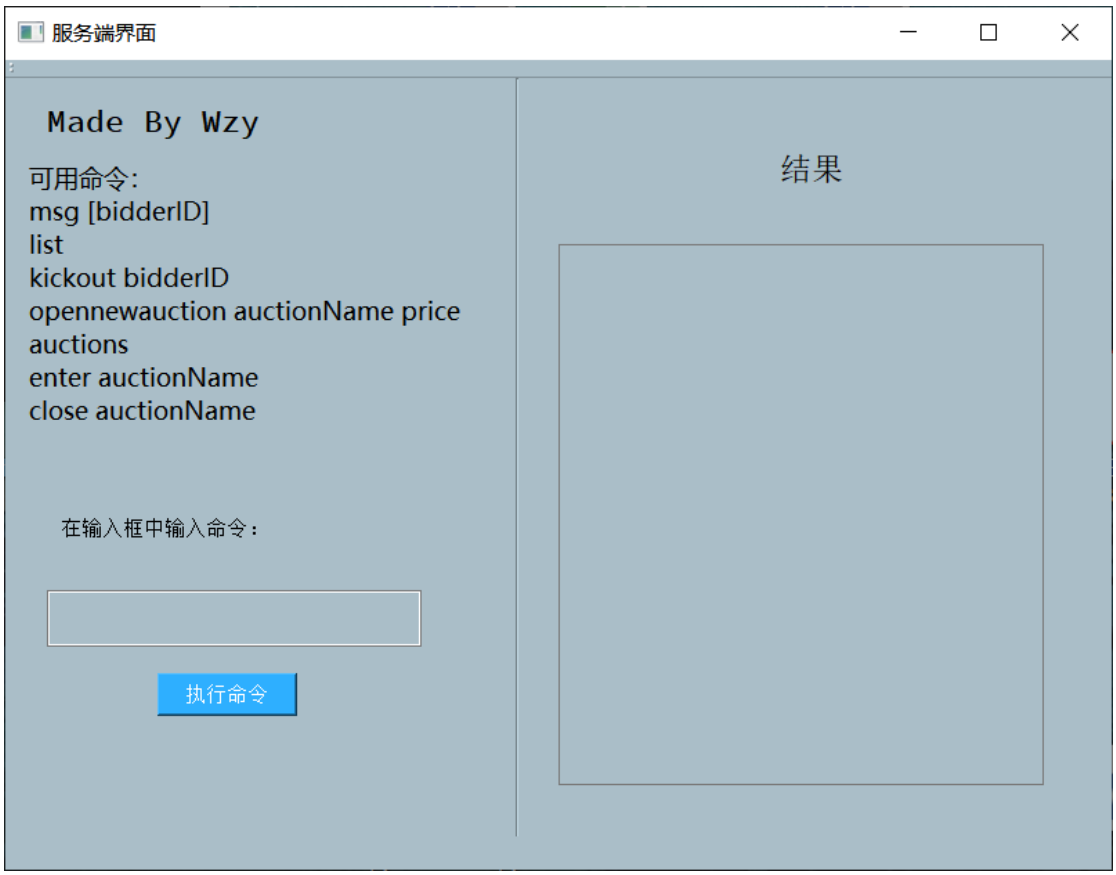
# 应用程序协议：

客户端和服务端利用 UDP 协议交互信息。

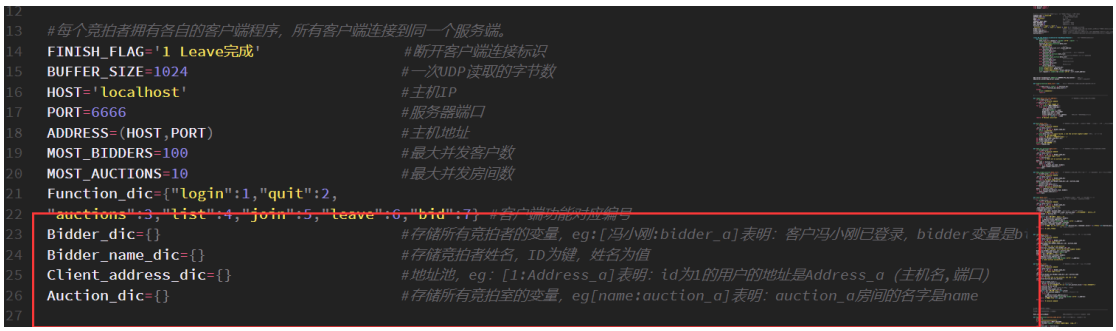
# 关键代码解释：

Server.py：

界面：




Server.py 实现了文档中要求的全部 7 个功能。



```
#每个竞拍者拥有各自的客户端程序，所有客户端连接到同一个服务端。
FINISH_FLAG='1 Leave完成'                    #断开客户端连接标识
BUFFER_SIZE=1024                             #一次UDP读取的字节数
HOST='localhost'                             #主机IP
PORT=6666                                    #服务器端口
ADDRESS=(HOST,PORT)                          #主机地址
MOST_BIDDERS=100                             #最大并发客户数
MOST_AUCTIONS=10                             #最大并发房间数
Function_dic={"login":1,"quit":2,
"auctions":3,"list":4,"join":5,"leave":6,"bid":7} #客户端功能对应编号
Bidder_dic={}                                #存储所有竞拍者的变量，eg:[冯小刚:bidder_a]表明：客户冯小刚已登录，bidder变量是b
Bidder_name_dic={}                           #存储竞拍者姓名，ID为键，姓名为值
Client_address_dic={}                        #地址池，eg：[1:Address_a]表明：id为1的用户的地址是Address_a（主机名,端口）
Auction_dic={}                               #存储所有竞拍室的变量，eg[name:auction_a]表明：auction_a房间的名字是name
```

这几个字典分别记录了所有的用户信息、竞拍室信息和用户地址对应关系等。它们作为全局变量可以被其他所有功能函数调用改动。

```python
class My_udp_handler(socketserver.BaseRequestHandler):      #与客户端交互的Handler
    def handle(self):
        data_list=self.request[0].decode('utf-8').split(' ')
        print("data_list=",data_list)
        function_id=Figure_Function(data_list)
        ret='0 无效命令! '
        if function_id==1:              #Login
            ret=Login(data_list,self.client_address)
        elif function_id==2:            #Quit
            ret=Quit(data_list)
        elif function_id==3:            #auctions输出所有竞拍室的情况
            ret=Show_all_auctions(data_list)
        elif function_id==4:            #list(auction)默认输出当前竞拍室的情况
            ret=Show_single_auction(data_list)
        elif function_id==5:            #Join(auction)
            ret=Join(data_list)
        elif function_id==6:            #Leave(auction)
            ret=Leave(data_list)
        elif function_id==7:            #Bid(price)
            ret=Bid(data_list)
        print("Auction_dic=",Auction_dic)
        print("bidder_dic=",Bidder_dic)
        print("Client_address_dic=",Client_address_dic)
```

这里利用 socketserver 库实现 UDP 服务器，可以等待客户端发消息，接收并处理。

主线程（即 UI 线程）：

```python
def Terminal_input_handler(ui):             #处理服务器输入命令的函数
    # while True:
        # cmd=input('>>:').strip()
        cmd=ui.lineEdit.text()  #读命令输入框的命令
        ui.lineEdit.setText('') #清空输入框
        cmd_list=cmd.split(' ')
        print(cmd_list)
        x=cmd_list[0].lower()
        try:
            if x=='opennewauction' and len(cmd_list)==3:
                OpenNewAuction(cmd_list[1],int(cmd_list[2]))
            elif x=='enter' and len(cmd_list)==2:
                Enter_auction(cmd_list[1])
            elif x=='list':
                if len(cmd_list)>1:
                    ui.textEdit.append('Invalid command')
                List()
            elif x=='auctions' and len(cmd_list)==1:
                List_all_auctions()
            elif x=='leave' and len(cmd_list)==1:
                Leave_auction_cmd()
```

对于服务端管理员输入的命令（如开新房间或喊话），在主线程中处理，通过绑定 pushbutton 点击来调用对应的函数。

线程 1：
服务器线程，负责处理客户端发来的功能请求。不同功能通过编号区分，对应关系存储在 function_id 中，调用对应的函数处理并返回消息给客户端。

```python
T_terminal_thread=threading.Thread(target=Server_init,args=[])#线程4处理UDP服务器的运行
T_terminal_thread.setDaemon(True)
T_terminal_thread.start()
```

线程 2：
处理所有房间的喊价。我设定的逻辑是这样的：如果有竞拍者叫了价（即不是初始的价格了），每隔 10 秒喊一次价，广播给房间内的所有竞拍者。第三次就落锤成交，通知房间内所有人并关闭房间，所有人回到大厅。

```
T_terminal_thread=threading.Thread(target=Broadcast,args=[])#线程3处理所有竞拍室的实时广播价格和叫价
T_terminal_thread.setDaemon(True)
T_terminal_thread.start()
```

```
def Broadcast():
    while 1:
        time.sleep(10)
        need_deleted_auc=set()
        for auc in Auction_dic.values():
            if not auc.has_bidders():       #没人出价，就先不叫，有人出价了再叫价
                continue
            msg='第'+str(auc.cnt+1)+'次叫价，当前出价：'+str(auc.price)
            auc.cnt+=1
            for i in auc.bidders:
                i_address=Client_address_dic[i]
                Udp_server_socket.sendto(msg.encode('utf-8'),i_address)
            if auc.cnt==3:  #成交，发通知并善后处理
                need_deleted_auc.add(auc)    #加入待删除集合
        for auc in need_deleted_auc:
            msg=str(auc.last_bidder_id)+'号用户拍得该商品，您已退出到大厅'
            #给没拍到商品的人发通知
            for i in auc.bidders:
                if i!=auc.last_bidder_id:
                    i_address=Client_address_dic[i]
                    Udp_server_socket.sendto(msg.encode('utf-8'),i_address)
            msg='恭喜您拍得了该商品，您已退出到大厅'
            #给拍到商品的人发通知
```

Client.py：

界面：


Client.py 实现了文档中要求的全部 6 个功能，额外实现了一个 quit 功能（退出整个客户端）。

```
########################################
while not MY_BIDDER.is_logined():       #旋转登陆
    msg='0 '
    msg+=input('>>请登入(命令: login Yourname)')
    client.sendto(msg.encode('utf-8'),('localhost',PORT))
    new_msg,_=client.recvfrom(BUFFER_SIZE)
    new_msg=new_msg.decode('utf-8')
    if new_msg==FINISH_FLAG:
        ui.textEdit.append('您已登出，请关闭界面退出')
        sys.exit(0)
    if new_msg[0]=='0':                 #返回报文首字节是0，说明出错
        continue
    new_msg=new_msg.split(' ')
    if len(new_msg)!=3:
        continue
    MY_BIDDER.bidder_id=int(new_msg[1])
    MY_BIDDER.bidder_name=new_msg[2]
    ui.textEdit.append('您已登入'+'您的ID为: '+str(MY_BIDDER.bidder_id))
########################################
```

首先旋转登录，在控制台输入命令。
一旦登入，初始化 UI 界面，之后的交互都由 GUI 实现。

主线程（ui 线程）：

```
def Sendto_server(ui):
    if MY_BIDDER.is_logined():
        msg=ui.lineEdit.text()      #读入输入框的命令
        ui.lineEdit.setText('')     #清空输入框
        msg=str(MY_BIDDER.bidder_id)+' '+msg#报文开头附上身份标识，这样服务器才知道是哪个客户端发送的请求，以及是否已
        client.sendto(msg.encode('utf-8'),('localhost',PORT))
    else:
        ui.textEdit.setText('您没有登录')
        ui.pushButton.setText('点击这里退出')
```

```
77
78
79    #主线程：等待用户输入并将消息发给服务器
80    ui.pushButton.clicked.connect(partial(Sendto_server,ui))
81    MainWindow.show()
82    sys.exit(app.exec_())
```

输入命令并发送给服务端，利用 pushbutton 绑定函数实现。

线程 1：
将服务器发来的包以正确的格式展示。

```
    # print('登陆成功')
    #线程1在右侧展示从服务器发来的通知消息
    T_cmd=threading.Thread(target=Print_received_msg,args=[ui,])
    T_cmd.setDaemon(True)
    T_cmd.start()
```

```
23
24  def Print_received_msg(ui):
25      while MY_BIDDER.is_logined():
26          new_msg=client.recv(BUFFER_SIZE)
27          new_msg=new_msg.decode('utf-8')
28          if new_msg==FINISH_FLAG:        #输入leave命令则关闭当前套接字并退出
29              MY_BIDDER.bidder_id=0
30              client.close()
31              sys.exit(0)
32          print(new_msg)
33          if len(new_msg)>1 and new_msg[0] in {'0','1'} and new_msg[1]==' ':
34              new_msg=new_msg[2:]
35          ui.textEdit.append(new_msg)     #服务器返回的命令执行结果打在右侧输出框中
```