设计文档

环境 python 3.6。为了支持中文，在发送信息时用 gb2312 编码，接受信息后用 gb2312 解码。为了防范一台机器对服务的洪范攻击，所以各个用户不同的识别机制使用 ip 地址，一个 ip 地址只能绑定一个登录用户。所以每次识别以及解析不同用户的信息的区别，只需解析 ip 地址即可。为了减少用户的输入，我确定了客户端端口号为 6766，以及服务器端的端口号为 8082。

用户功能的设计：

首先，要让需要用户有身份鉴定的机制，所以需要用户名以及密码，因此在第一步时，需要将注册机制与登录机制结合起来，这也就需要服务器储存用户信息，无论是否启动在线，所以我用 user_data.txt 来存放这些信息，当开机启动时，就可以将其恢复。让注册用户正常运用。

其次，需要用户知道自己已存在的外语角以及在线用户，以及其它模块。需要支持中文以及英文所以在发送和接收信息时，使用了 gb2312 编码。

当用户打开并登录软件后，一边需要随时接受信息、另一边需要随时输入指令，且还需要 GUI 的操作，所以使用了多线程，GUI 操作为主线程，一个时刻接收消息和另一个随时等待输入命令的是两个子线程。

当接受到命令和数据后需要对命令数据进行进一步的分析处理。以及返回结果。

为了能够在登录以及注册成功的信息两个进程之间通信，我设置了全区变量 LoginOK，当接受到正确的信息后，我便通过操作 LoginOK，以及识别它的变化，来确定是否登录成功。

服务器的设计:

首先需要一个专门的类来处理，外语角的用户信息，包括登录信息，如下:

```
self.user = {'hailong': '0000', 'liangliang': '1111', 'junjun':
'2222'}    # 所有注册用户
self.login_user = set(['hailong'])     # 所有登录用户
self.login_user_detail = {'hailong': {'ip': '127.0.0.1',
'EnterCorner': 'English_show'}}    # 登陆用户的详细信息
self.corner_list = set(['English_show'])  # 所有打开的外语角
self.corner_list_detail = {'English_show': {'language': 'English',
'this_corner_user': set(['hailong'])}}  # 外语角的详细信息
```

下面是一些数据的处理函数:

其中 ip_to_ip_list()是由某个 ip 的用户获取他所在外语角的其它用户的 ip 列表，

发送信心以及加入、退出广播时，都需要用到它。

Ip_to_name2()是将 ip 信息转换成用户名，在聊天室打印信息时需要用来表明不

同的用户。

```
    def __del__(self):    # 当关闭服务器时，储存用户信息

    def dele_user_in_corner(self, temp_corner, kick_id):

    def name_to_corner(self,name):

    def ip_to_ip_list(self, ip):

    def ip_to_name2(self, ip):

    def name_to_ip(self, name):

    def ip_list(self, corner_name):

    def use_leave(self, ip):

    def add_corner(self, data):
    def list_corner(self):

    def c_list_corner(self, ip):
```

```
    def c_list_user(self, ip):


    def list_all_user(self):


    def c_join_corner(self, cornername, ip):

            s
    def enter_corner(self, data):


    def delect_user(self,id):

    def close_corner(self,concer_name):


    def if_in_user(self,id,key,ip):

    def add_in_user(self, id, key, ip):
```

接着是一些线程函数接收信息后处理信息以及 cmd 指令的执行命令函数。

```
# 收到来自网络的信息，分析操作收到的信息
def analyse_data(recv_data):

# 分析 cmd 输入的信息
def deal_with_cmd(cmd):
   # 分析输入的 cmd 指令
def deal_cmd(cmd):

# 等待输入 cmd 的指令
def Termi_cmd():

# 接受数据
def recvData():

# 组装发送信息的格式
def send_Data(sendInfo,ip_list):

# 具体的发送信息
def sendData(sendInfo):
```

```
# 私信的处理函数
def private_msg():


# 发送管理员信息
def msg():
   # 显示打开的外语角
def CORNER():
    # 显示在线的用户
def LISTUSER():
  def LEAVE():
    # 关闭一个外语角并通知，里面的所有用户
def CLOSECOR():

# 踢出某个用户，并通知其他人
def KICKOUT():

# 新开一个外语角
def open_corner():

def ENTER():


udpSocket = socket(AF_INET, SOCK_DGRAM)
destIp = ""
myserver_Port = 8082
cmd = ""
Data = data()
sendInfo = ""
threadLock = threading.Lock()
init_window = Tk()
GUI = None
ENTER_CORNER = None


def main():

    global udpSocket
    global destIp
    global myserver_Port
    global init_window
    global GUI
    myserver_Port = 8082
```

```
    udpSocket = socket(AF_INET, SOCK_DGRAM)
    udpSocket.bind(("", myserver_Port))

    GUI = MY_GUI(init_window)  # 设置根窗口默认属性
    GUI.set_init_window()

    tr = Thread(target=recvData)
    ts = Thread(target=Termi_cmd)
    tr.start()
    ts.start()
    init_window.mainloop()  # 父窗口进入事件循环，可以理解为保持窗口
运行，否则界面不展示
    tr.join()
    ts.join()

    udpSocket.close()


if __name__ == '__main__':
    main()
```