****

**计算机网络课程设计**

**报告**

实习任务名称： Project 1

学 号： 20181003443

姓 名： 孙思思

**中国地质大学地理信息工程学院**

**软件工程系**

**2020年 7 月**

目录

[1 客户端 2](#_Toc44885346)

[1.1 需求简析 2](#_Toc44885347)

[1.2 顶层设计 2](#_Toc44885348)

[1.3 详细设计 2](#_Toc44885349)

[1.3 学到的知识 3](#_Toc44885350)

[1.4 测试成绩 4](#_Toc44885351)

[2 服务器 4](#_Toc44885352)

[2.1 需求简析 4](#_Toc44885353)

[2.2 顶层设计 4](#_Toc44885354)

[2.3 详细设计 5](#_Toc44885355)

[2.4 测试成绩 6](#_Toc44885356)

## 1 客户端

### 1.1 需求简析

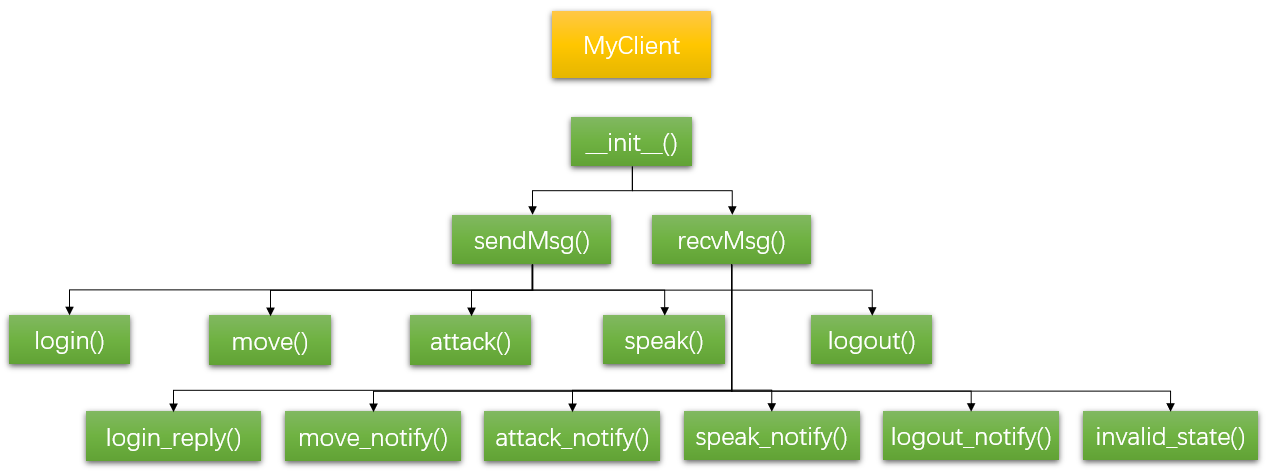
当用户输入login、move、attack、speak、logout命令时，依据TWW协议把命令打包发送到服务器；

当收到服务器返回包时，需要解析出login\_reply、move\_notify、attack\_notify、speak\_notify、logout\_notify、invalid\_state消息，并打印出正确的提示信息；

当与服务器的连接出现异常的时候，以及收到的回包无效的时候，需要退出客户端。

### 1.2 顶层设计

客户端的主要功能由两个线程实现：一个负责接收玩家命令并向服务器发送请求，对应sendMsg（）函数；另一个负责接收服务器传送过来的消息，并解析输出到终端窗口，对应recvMsg（）函数。



图一 客户端MyClient类的主要函数依赖关系

### 1.3 详细设计

**（1）sendSwitch、recvSwitch字典**

python没有switch语句，但是它有超级好用的字典，可以把关键字与函数对应起来，以此来实现switch功能。

self.sendSwitch = {**"login"**: self.login,  
 **"move"**: self.move,  
 **"attack"**: self.attack,  
 **"speak"**: self.speak,  
 **"logout"**: self.logout,  
 }  
self.recvSwitch = {2: self.login\_reply,  
 4: self.move\_notify,  
 6: self.attack\_notify,  
 8: self.speak\_notify,  
 10: self.logout\_notify,  
 11: self.invalid\_state,  
 }

**（2）SendMsg函数**

**def** sendMsg(self):  
 **while** True:line = raw\_input()  
 arr = line.split(**" "**, 1)  
 **try**:  
 self.**sendSwitch**.get(arr[0], self.defaultSend)(arr)  
 **except**:  
 os.\_exit(2)

**（3）RecvMsg函数**

**def** recvMsg(self):  
 **while** True:data = self.tcpCliSock.recv(1024)  
 **if** len(data) == 0:  
 **print** (**"The gate to the tiny world of warcraft has disappeared."**)  
 sys.stdout.flush()  
 os.\_exit(2)  
 **while** len(data) > 0:version, length, kind, info = struct.unpack(**">BHB"** + str(len(data) - 4) + **"s"**, data)  
 **if** length - 4 < len(info):  
 data = info[length - 4:len(info)]  
 **else**:  
 data = **""** info = info[0:length - 4]  
 **if** version != self.version **or** len(info) % 4 != 0:  
 **print** (**"Meteor is striking the world."**)  
 sys.stdout.flush()  
 os.\_exit(2)  
 self.**recvSwitch**.get(kind)(info)  
 sys.stdout.flush()

### 1.3 学到的知识

**（1）struct**

用struct的pack与unpack来实现TWW协议，用来替换c++的联合体union，来实现字节组合与拆分。

* struct.pack(fmt,v1,v2,.....)
* struct.unpack(fmt,string)

常用的fmt有：

**B** unsigned char 1B

**H** unsigned short 2B

**i** int 4B

**s** char[] 1B 前面带数字

例：s = struct.pack(">BHB",1,1,1)

> big-endian 01 **00 01** 01

< little-endian 01 **01 00** 01

= native 01 **01 00** 01

例：msg = struct.pack(">BHB10sH", 4, 16, 1, "susns", 0)

04 00 10 01 73 75 73 6e 73 00 00 00 00 00 00 00

**（2）getopt**

获取的命令行参数为 python client.py -s 127.0.0.1 -p 12345

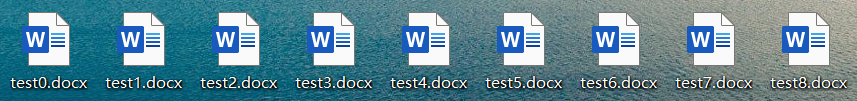
* opts, args = getopt.getopt(sys.argv[1:], "s:p:")

**（3）sys.stdout.flush()**

在Linux上用test1a测试的时候获取不了我写的客户端的print内容，可能是操作系统原因吧，需要加一个sys.stdout.flush()，实现每次print都输出，而不是滞留在缓存里面。

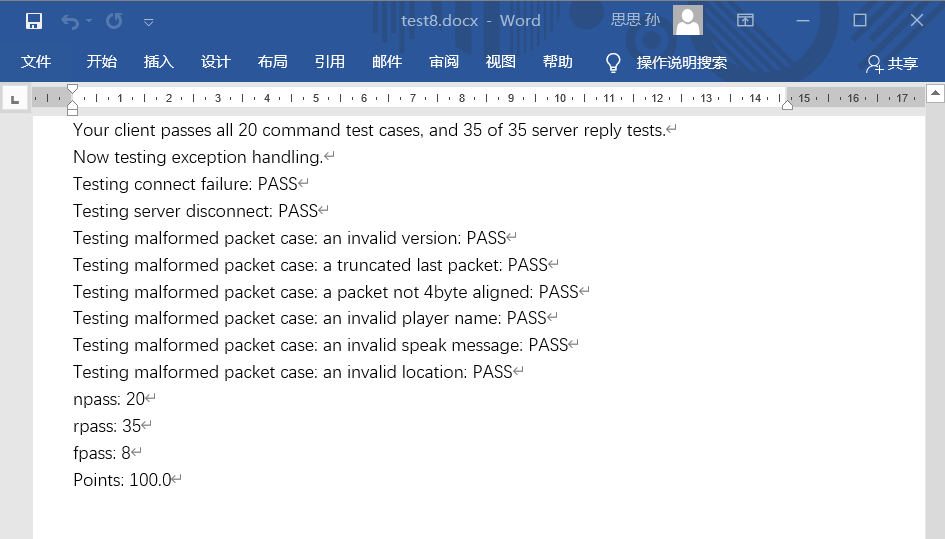
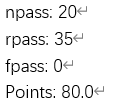
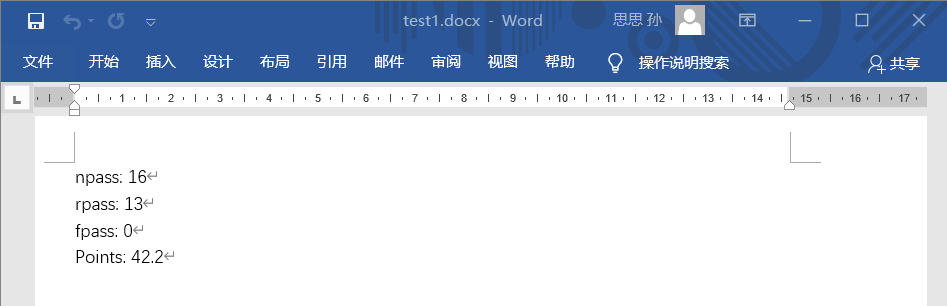
### 1.4 测试成绩

前前后后测试了9次，最后终于修修补补通过了所有测试。



图二 每次测试的记录文件

从24.7 -> 42.2 -> 80 -> 100:



图三 第一、二、五、九次测试成绩

## 2 服务器

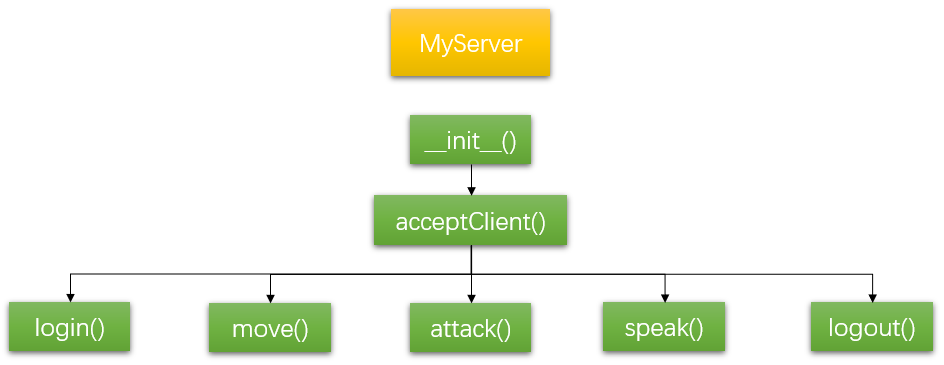
### 2.1 需求简析

响应服务器的login、move、attack、speak、logout请求，并按照游戏规定以及TWW协议进行回包或群发消息；

当与客户端的连接出现问题、收到的请求无效的时候，断开与客户端的连接，服务器还可以正常接受新的连接。

### 2.2 顶层设计

服务器用的是多线程技术，每一个线程处理一个客户端的连接，用来开启一个线程的函数是**acceptClient**。



图四 MyServer类的主要函数依赖关系

### 2.3 详细设计

**（1）数据结构**

|  |  |  |
| --- | --- | --- |
| 变量名 | 变量类型 | 描述 |
| version | 整型 | 用来存版本号 |
| tcpSerSock | Socket | 服务器的socket |
| loginer | 字符串数组 | 记录登录的用户名 |
| msg\_type\_switch | 字典 | 映射收到请求的处理函数 |
| players | 字典 | key为连接服务器socket的客户端，val为player\_info |

注：play\_info:字典，key：“name”、“HP”、“EXP”、“X”、“Y”、“date”

**name**：str，记录用户名

**HP**：int，玩家血量，生成新玩家时的初始血量或者读入文件存储的血量，一般不变

**EXP**：int，玩家经验

**X，Y**：int，玩家在地牢的坐标

**date**：double，生成新玩家的时间或从文件读入玩家的时间，用当前时间、date、HP可以获得玩家当前的血量，因为HP每5秒涨1

self.msg\_type\_switch = {1:self.login,  
 3:self.move,  
 5:self.attack,  
 7:self.speak,  
 9:self.logout,  
 }

**（2）消息预判断**

接收到move、speak、attack……请求时，可能该客户端还没有登录，这时需要产生invalid\_state消息，发送给请求的服务器。

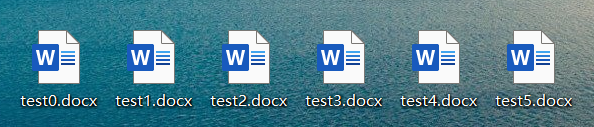
**def** acceptClient(self,tcpClient):  
 **try**:  
 **while** True:  
 data = tcpClient.recv(1024)  
 **……**

version,length,kind,info=struct.unpack(**">BHB"**+str(len(data)-4)+**"s"**,data)  
 **……**

**if** kind == 1 **and** len(self.players[tcpClient])>0: #重复登录  
msg = struct.pack(**">BHBB3s"**, self.version, 8, 11, 1, **'\0'**)  
 tcpClient.send(msg)  
 **elif** kind == 1 **or** len(self.players[tcpClient])>0:  
 self.**msg\_type\_switch**.get(kind,self.logout)(tcpClient,info)  
 **else**:#未登录却请求其他命令  
 msg = struct.pack(**">BHBB3s"**, self.version, 8, 11, 0, **'\0'**)   
 tcpClient.send(msg)  
 **except**:  
 ……

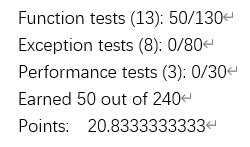
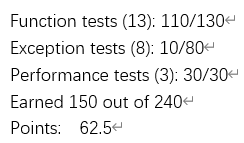
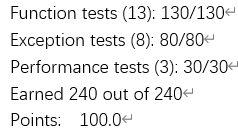
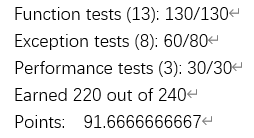
### 2.4 测试成绩

目前一共测试5次，最好成绩91.6



图五 test1b测试结果存档

20.8 -> 62.5 -> 91.6 -> 100:

图六 第一、三、五、六次测试