**实践报告**

**Task 3: TCP socket programming**

**Part 1:**

1. client 和 server都是基于TCP的命令行程序。client运行在host os, server运行在 guest os。
2. 有一个全英文可打印字符的ASCII文件(这个文件你要自己构造)，client 每次发送这个文件“不定长的一段/一块”给 server,由 server把这块文本 reverse(例如，“a little monkey“变成“yeknom elttila"，并返回给 cient, clien在终端打印显示。最终client要输出一个文本文件，该文件是原始文件的全部反转。
3. client 和 server之间的交互如下:

client首先发送Initialization给server，server再发送agreement给client，client再发送reverseRequest1给server，server再发送reverseAnswer1给client，client再发送reverseRequest2给server，server再发送reverseAnswer2给client，如图1.1所示：

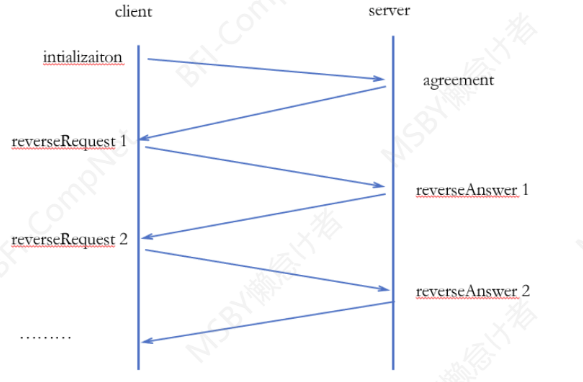


图1.1

1. client和server之间共4种类型报文，Initialization 、agree、reverseRequest、reverseAnswer，如图1.2所示：

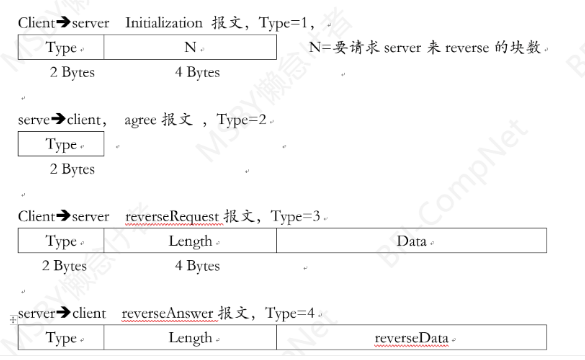


图1.2

1. 四种报文类型，通过Type字段来区分。client端向server端发送Initialization报文，告知 server要做reverse的块数N, server回复 agree 文。client逐次向server发送reverseRequest报文，包含了要做 reverse的数据; server端返回reverseAnswer报文，其中包含了reverse之后的数据。reverseRequest、reverseAnswer报文中Length字段分别指的是Data和 reverseData的长度，单位是Byte。
2. Server端启动后给出一个提示表示正在监听，如图1.3所示。 client端启动时在命令行方式下指定serverlP ，serverPort，ASCII文件路径，Lmin，Lmax，如图1.4所示。若未输入，则给出提示，要求输入正确的格式，如图1.5所示：

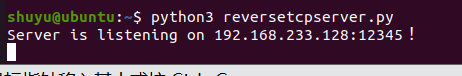


图1.3

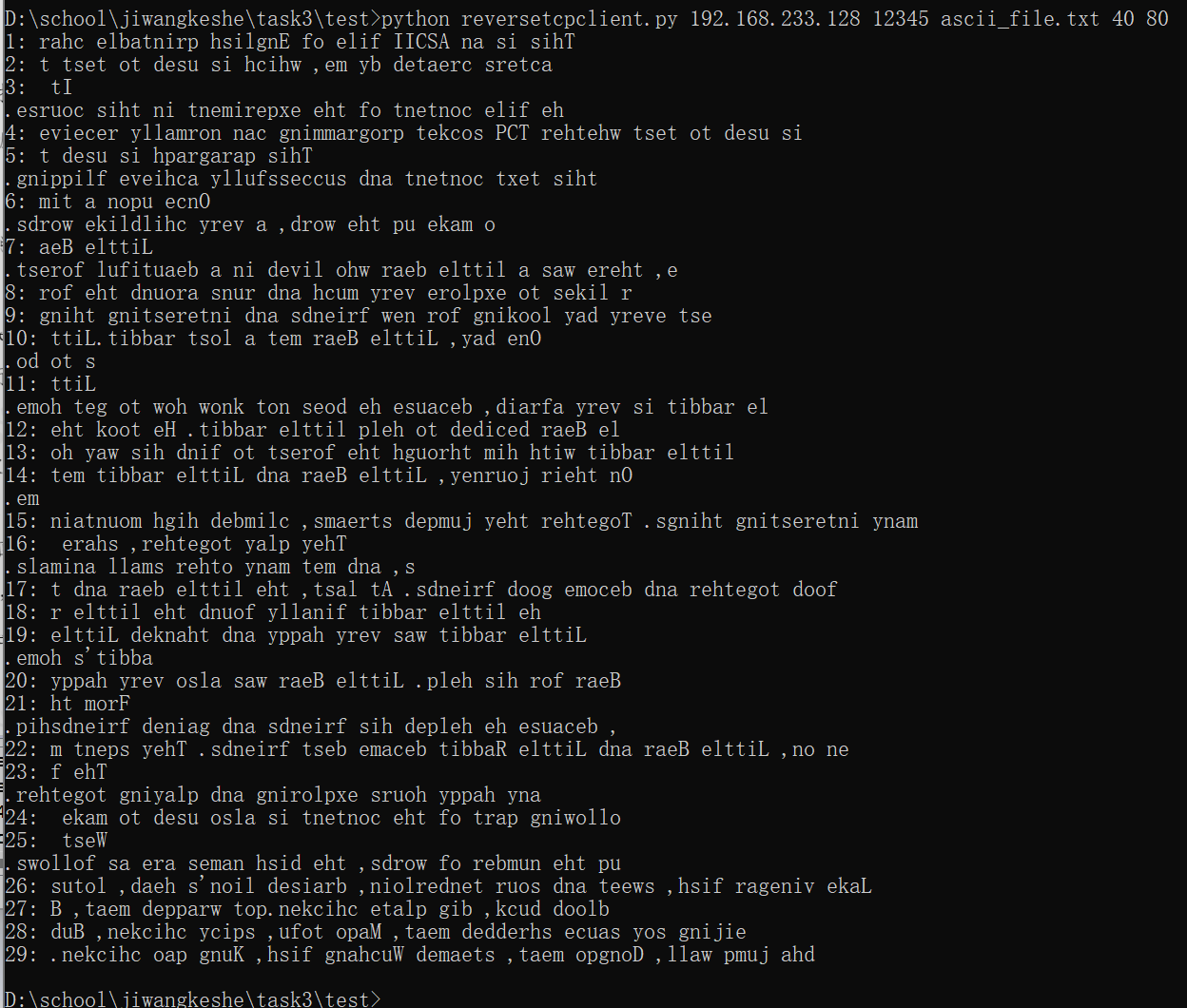


图1.4

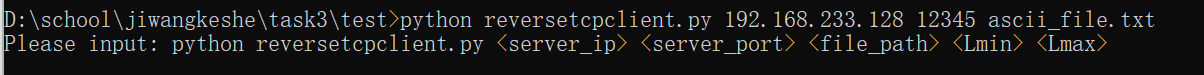


图1.4

1. client每次发送的reverseRequest中的长度都不一样(即不是例如每次固定发送500字节)，长度限定在一个[Lmin, Lmax]之间，最后一块除外。

a.(推荐但非必须)Lmin、Lmax在命令行中指定。

b.在开始发送数据块前，要先随机确定待发送File中的各块的Bytes长度，再计算块数N，才能执行client->server的Initialization报文

1. 每次client收到来自server 的reverseAnswer 报文，都在命令行下打印出来“第x块:反转的文本”。如“8: yeknom elttil a”，如图1.5所示，其输出原文件的全部反转如图1.6所示：

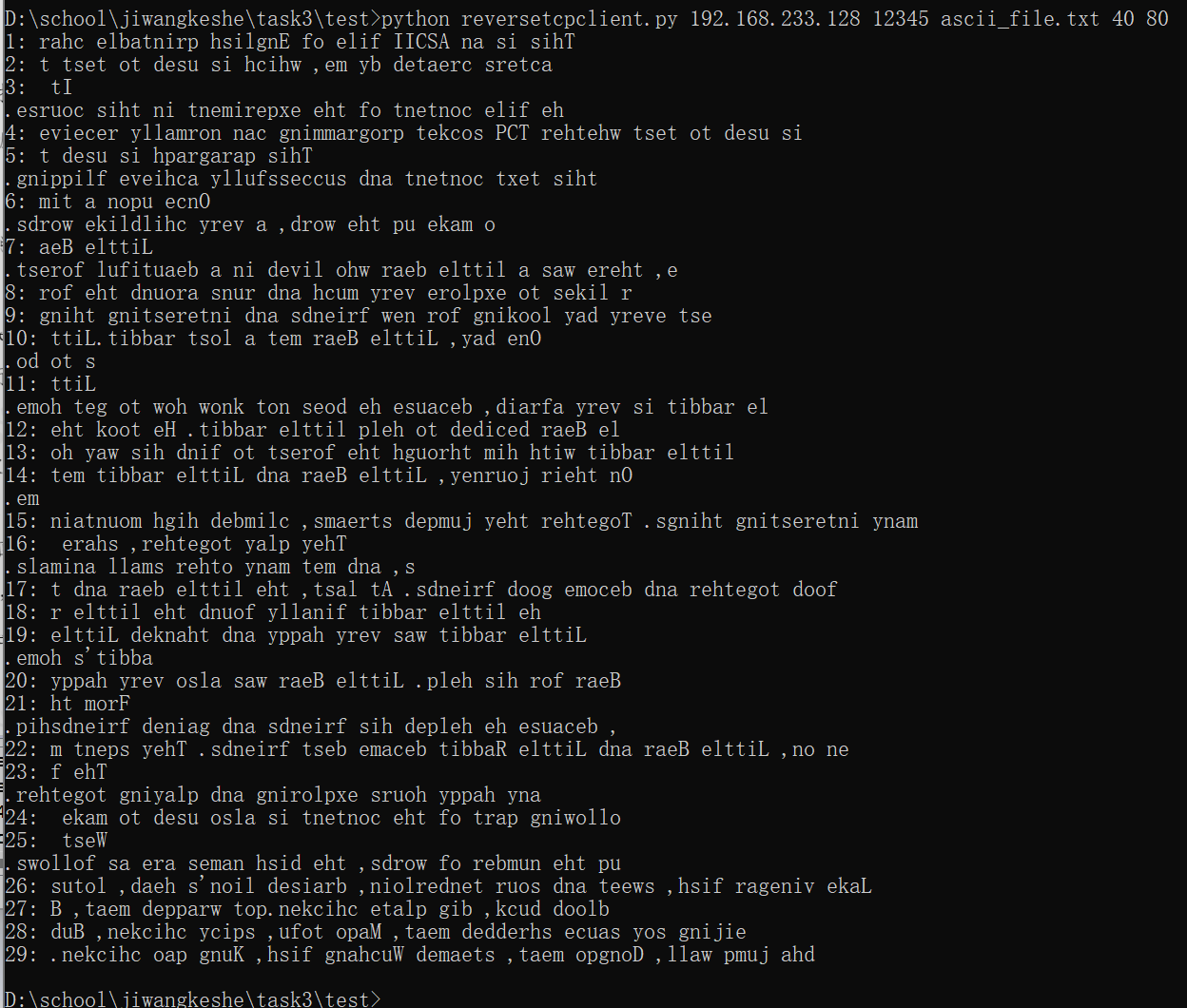


图1.5

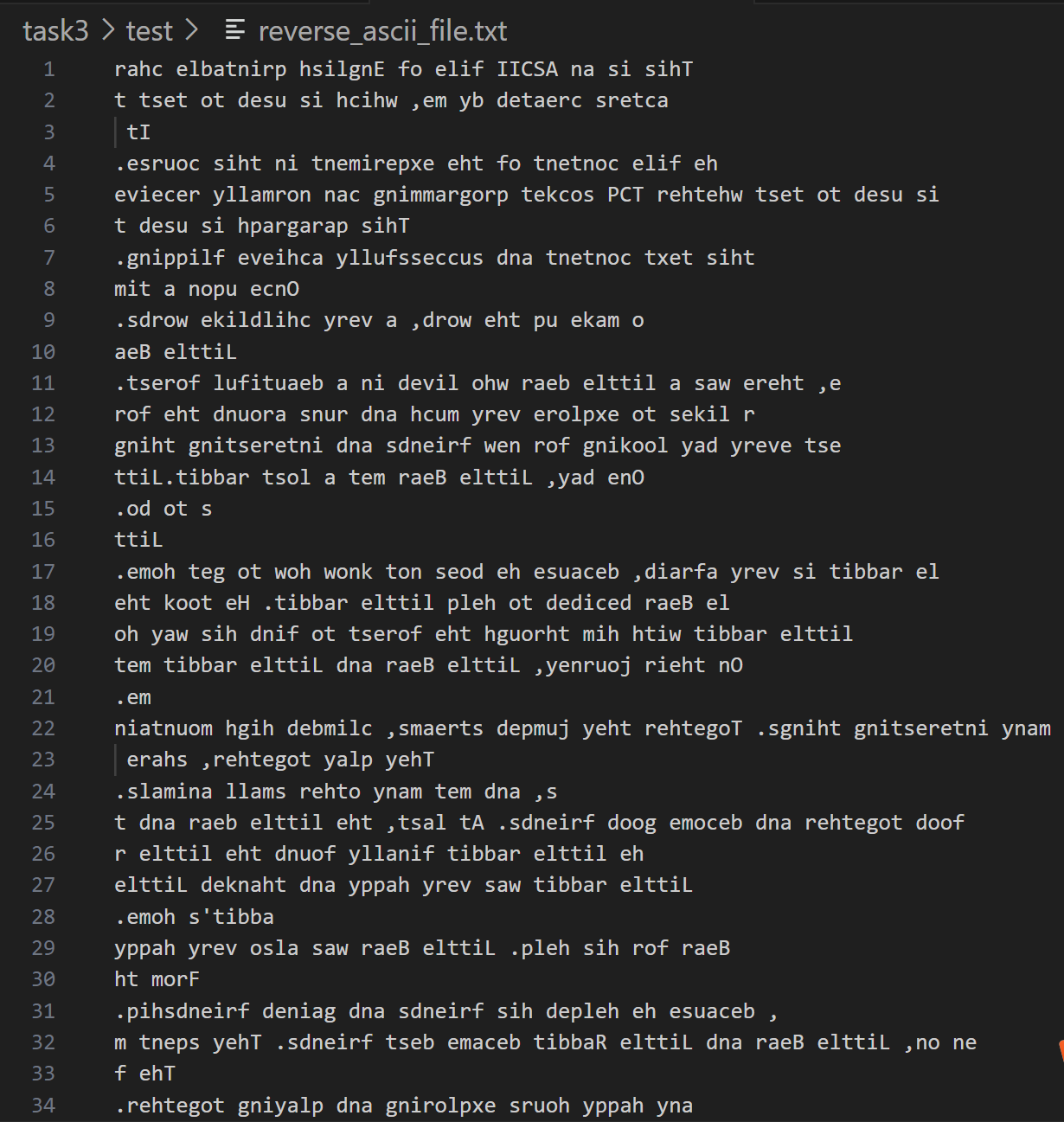


图1.6

1. (非必须实现，bonus 项:)server 能够同时处理2个及以上client的同时请求，即非阻塞，non-blocking。(处理non-blocking，有若干种实现方式，例如 select、epoll等。)，如图1.7所示：

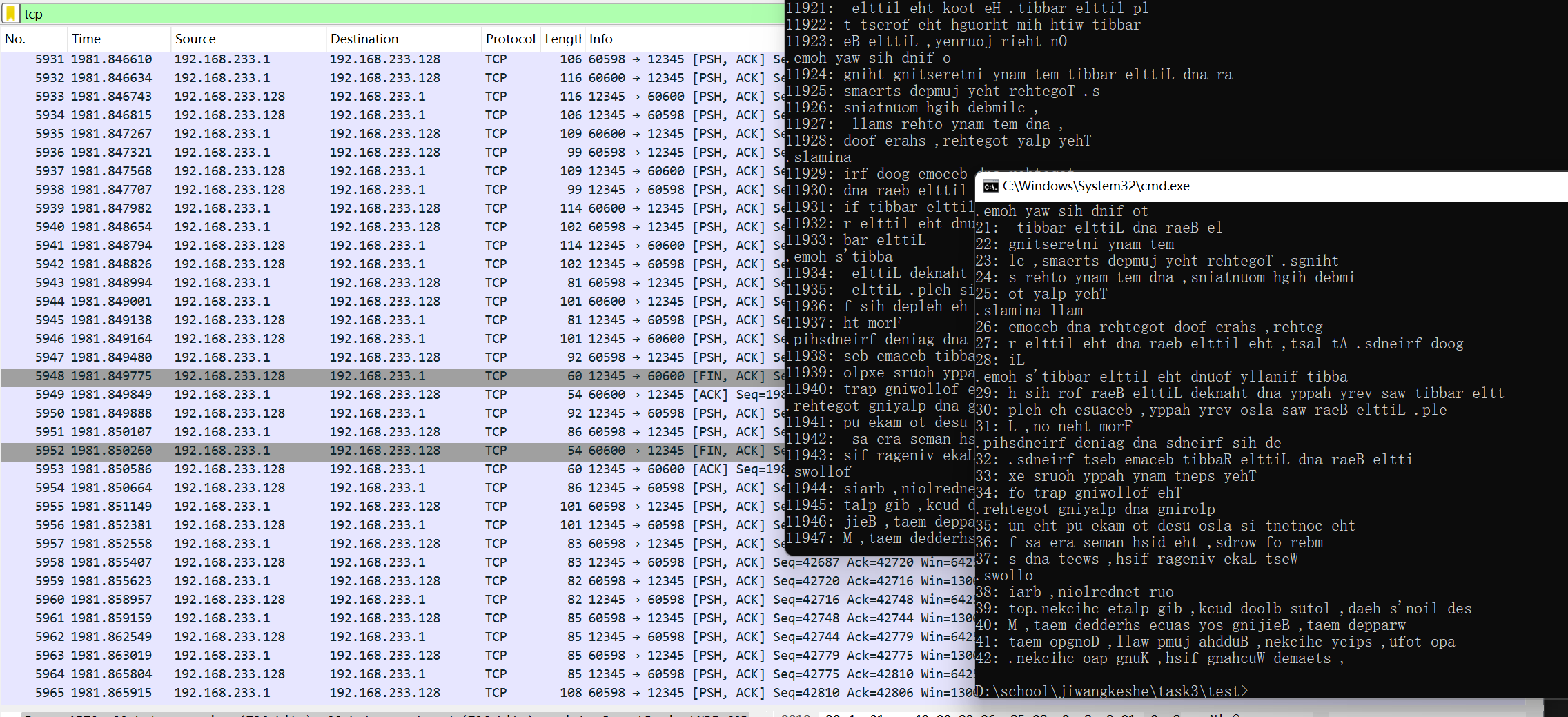


图1.7

1. Wireshark包捕获的截图如图1.8所示：

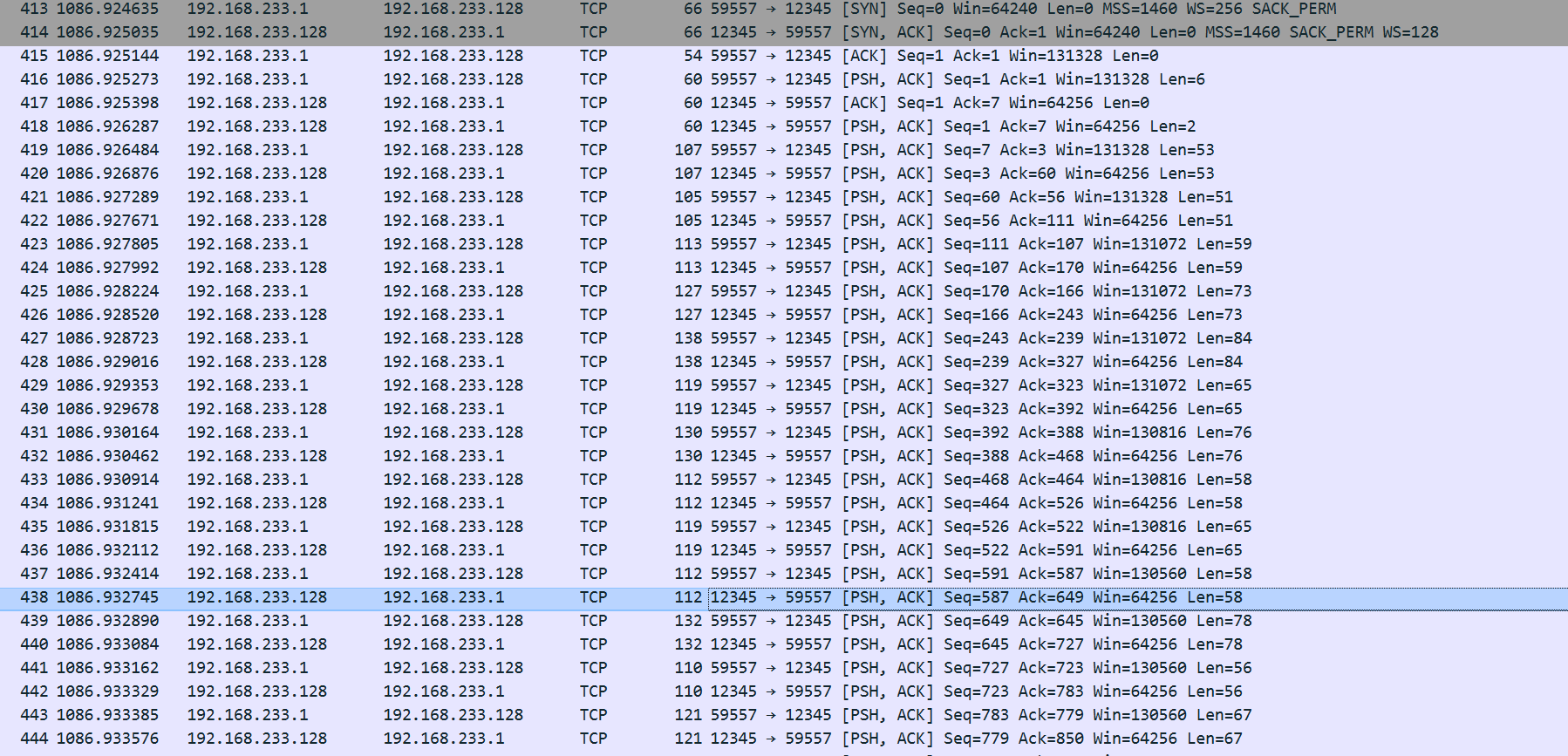


图1.8

413-415行为TCP连接建立的过程，如图1.9所示：

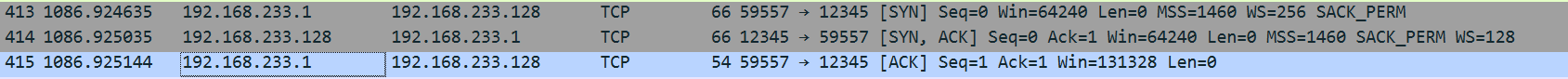


图1.9

416行为client首先发送Initialization给server，417行server回应接收成功，418行为server再发送agreement给client，如图1.10所示：

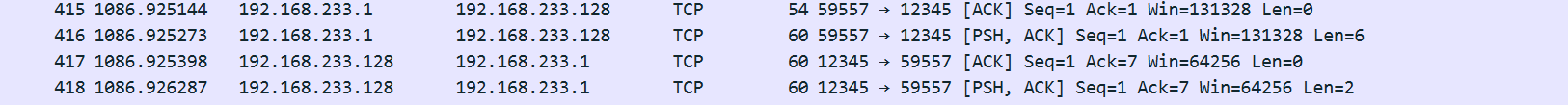


图1.10

Initialization的Type为0001（即1），数据块长度0000001d（即29），Agree的Type为0002（即2），如图1.11，图1.12所示：

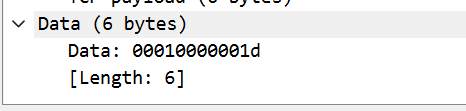


图1.11

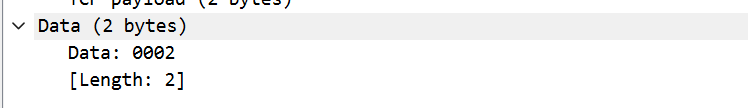


图1.12

419-420行为client发送reverseRequest1给server，server再发送reverseAnswer1给client，其length为6（Type和Data Length）+47（数据长度），如图1.13所示：

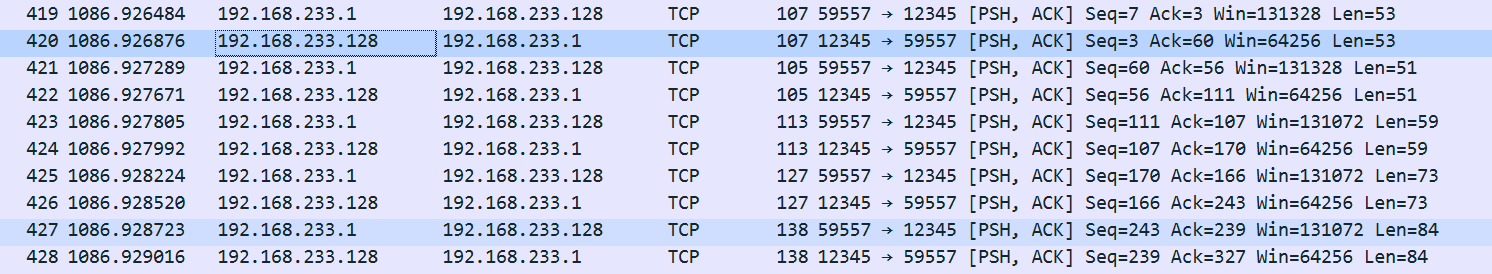


图1.13

reverseRequest1的Type为0003（即3），数据块长度0000002f（即47），reverseAnswer1的Type为0004（即4），数据块长度0000002f（即47），故总长度为Type（2）+data length （4）+ data（47） = 53，如图1.14，图1.15所示：

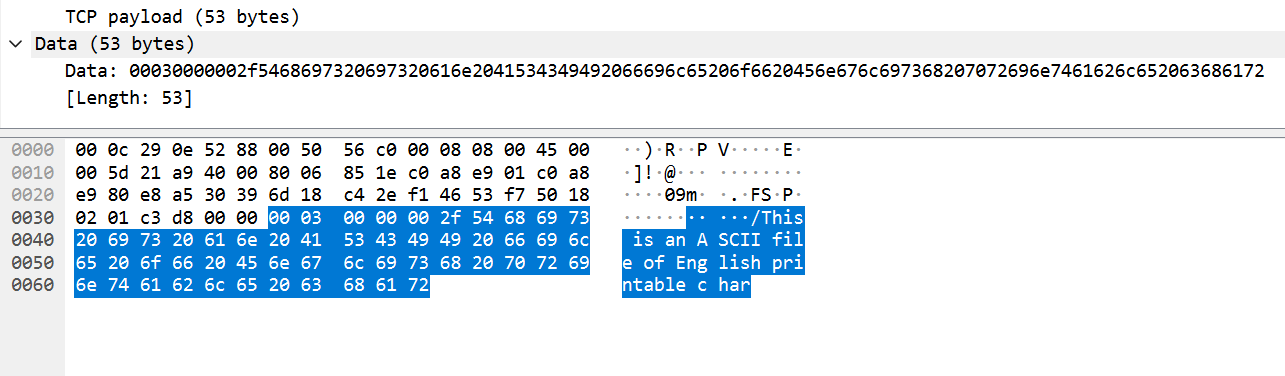


图1.14

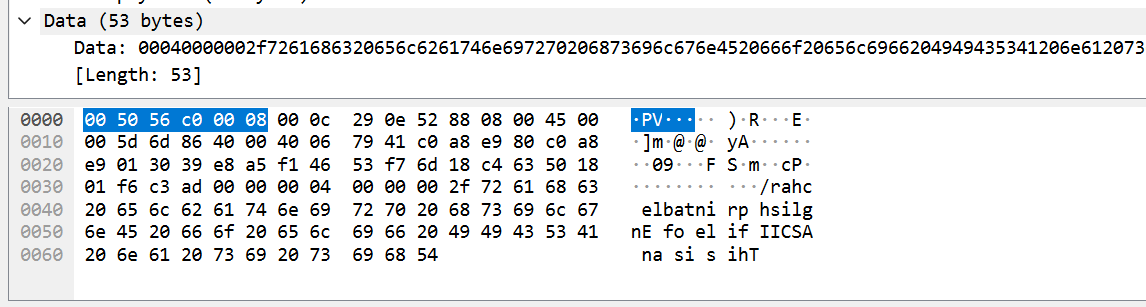


图1.15

477-480行为TCP连接释放的过程，如图1.16所示：

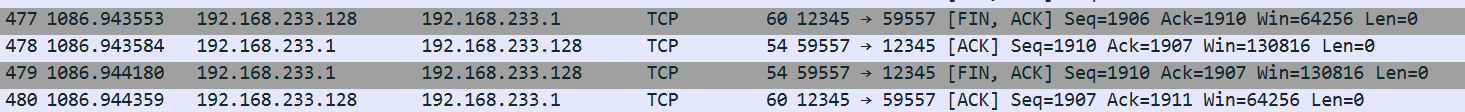


图1.16

**Part 2:**

**实现的关键点和对应的（代码）解决方案**

1. 关键点一：接收数据包。

解决方案：

定义了一个receive\_packet函数，用于接收固定长度的数据包，它循环接收数据直到达到指定的长度，如图2.1所示：

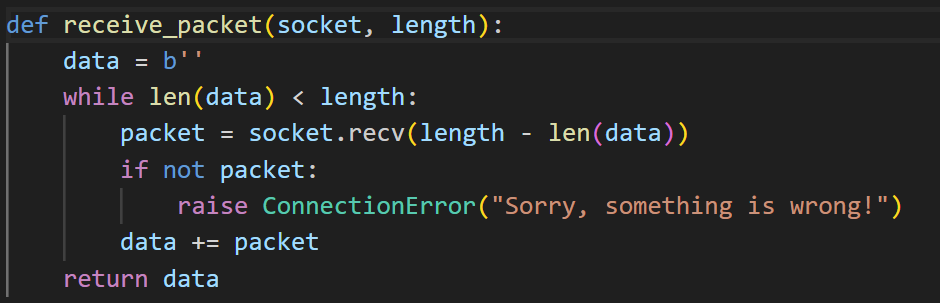


图2.1

1. 关键点二，数据块的反转与发送。

解决方案：

服务器接收到数据块后，将其从字节串解码为ASCII字符串，然后使用切片操作[::-1]进行反转，最后再编码回字节串；使用struct模块来构造反转回答(reverseAnswer)数据包。首先，它将数据包类型和反转后数据块的长度打包成一个字节串，然后将反转后的数据块追加到这个字节串后面，最后再使用sendall方法将构造好的反转回答数据包发送回客户端，如图2.2所示：

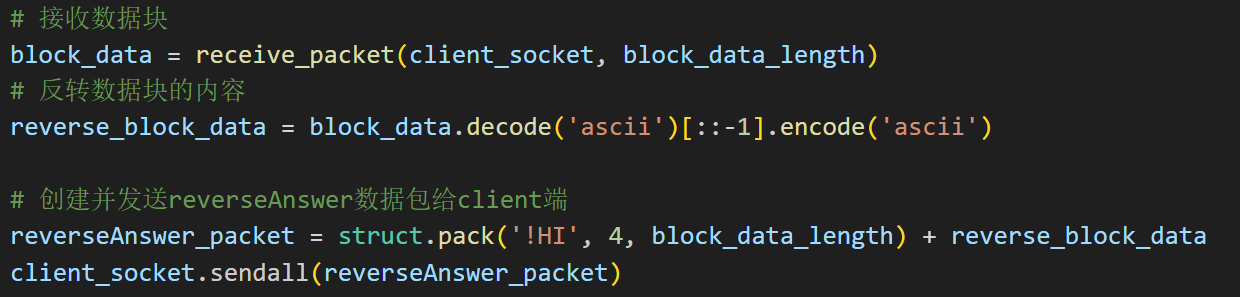


图2.2

1. 关键点三，文件读取与分块。

解决方案：

使用open函数打开指定的文件，并读取文件的全部内容，初始化一个空列表blocks来存储分割后的数据块，以及一个变量current\_start\_pos来记录当前数据块的起始位置。同时，获取文件的长度以便后续的分割操作；再使用一个while循环来分割数据块，直到current\_start\_pos达到文件的长度。在每次循环中，生成一个随机长度block\_length，该长度介于指定的Lmin和Lmax之间，根据当前数据块的起始位置和随机生成的数据块长度，计算数据块的结束位置，使用min函数确保不会超出文件的总长度；从文件内容中提取出从current\_start\_pos到current\_end\_pos的部分，并将其作为一个新的数据块添加到blocks列表中，将current\_start\_pos更新为下一个数据块的起始位置，即当前数据块的结束位置，在循环结束后，返回包含所有分割后的数据块的列表，如图2.3所示：



图2.3

1. 关键点四，反转后数据块的收集与保存。

解决方案：

客户端收集所有反转后的数据块，并将它们保存到一个新的文件中，如图2.4所示：

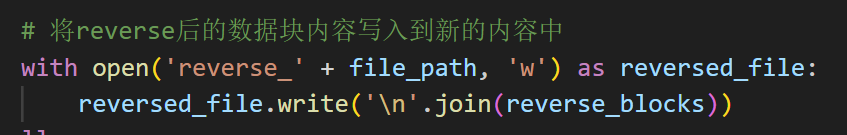


图2.4

**Part 3:**

**运用并掌握的知识点**

在本次的课设实践中，我不仅重新巩固复习了TCP的连接建立和连接释放过程，通过这一过程，我对使用Python进行TCP套接字编程，包括创建、绑定、监听和接受连接等操作以及数据传输的细节有了更清晰的认识。

同时，还可以了解如何使用多线程处理多个客户端连接，提高程序的并发处理能力。通过掌握struct.pack和struct.unpack的使用，我学会了如何确保数据在网络传输中的字节顺序一致性，以及如何通过设置特定的字段来识别和处理传输的数据信息和类型。

此外，在文件操作方面，我进一步了解和学习了如何读取文件内容并将其分割成多个数据块进行处理，并将反转后的数据重新保存在一个新的文件中。在此基础之上，我对python的字符串切片和反转有了更深刻的理解，如何正确使用切片语法来获取字符串的子串以及步长来实现字符串的反转。

总之，通过本次的课设实践，我对以往在课堂上所学习到的TCP相关的知识理解更加透彻，对知识点的学习和理解进一步加深，通过上网查询相关的资料以及翻阅课本、笔记，复习巩固了知识点，并加以运用，从理论到实践，让我对python编程和网络编程的各个方面都有了更加全面和深入的认识，所学的知识能够融合汇聚。

**Part 4:**

**git的url如下：**