NetworkingLab实验报告

实验完成度

编写了一对C语言服务端和客户端以及一对Python服务端和客户端，并成功通过了PartA的20个测试点。

实验思路

以C语言实现思路为例，python实现同理。

server.c

该程序实现了一个简单的TCP服务器，其核心流程如下：

1. 在main函数中，解析命令行参数，获取服务器端口号，调用server函数并传递服务器端口号作为参数。
2. 在server函数中，创建一个套接字（socket）并检查是否创建成功。如果创建失败，输出错误信息并返回-1。
3. 填充服务器地址结构体ServerAddr，包括地址族（AF_INET）、端口号（通过atoi函数将字符串形式的端口号转换为整数，并使用htons函数将其转换为网络字节序）和IP地址（使用INADDR_ANY表示服务器可以绑定到任意可用的网络接口）。
4. 使用bind函数将套接字绑定到服务器地址结构体。如果绑定失败，输出错误信息，关闭套接字并返回-1。
5. 使用listen函数开始监听连接请求。如果监听失败，输出错误信息，关闭套接字并返回-1。
6. 不断地循环使用accept函数接受连接请求，并创建一个新的文件描述符（文件描述符用于读取和写入连接上的数据）。如果接受失败，输出错误信息，并继续等待下一个连接请求。
7. 若accept成功，则使用recv函数接收客户端发送的数据，如果接收到的数据大于0，将其写入标准输出，并刷新输出缓冲区。
8. 接受消息完成后或引发异常则关闭对应的描述符，最后退出程序。

client.c

该程序实现了一个简单的客户端，其核心流程如下：

1. main函数是程序的入口函数，它解析命令行参数并调用client函数。
2. 在client函数内，首先使用socket函数创建一个套接字，指定地址族（AF_INET表示IPv4），套接字类型（SOCK_STREAM表示TCP套接字），协议（0表示默认协议）。
3. 若创建套接字成功，则创建struct sockaddr_in结构体，用于指定服务器的地址信息，包括地址族、端口号和IP地址。
4. 使用connect函数连接到服务器，并传递套接字描述符、服务器地址结构体和结构体的大小作为参数。如果连接失败，则使用perror函数输出错误信息，关闭套接字，然后返回-1表示失败。
5. 不断地从标准输入中读取消息，并使用send函数发送到服务器，如果发送消息失败，则输出错误信息，关闭套接字，然后返回-1表示失败。
6. 当从标准输入读取的消息长度为0时，表示消息发送完毕，关闭套接字，返回0表示成功。

遇到的一些有趣的问题

1. 在写发送长度时不能调用strlen函数，**因为生成的随机二进制数据中如果含有'\0'就会截断后面的内容**，这是个比较隐蔽的漏洞。
2. 最开始思考怎么并行处理10个请求想了很久，后来发现不用写并行。。。。。。

实验结果

```
16. TEST SHORT MESSAGE

SUCCESS: Message received matches message sent!
----------------------------------------

17. TEST RANDOM ALPHANUMERIC MESSAGE

SUCCESS: Message received matches message sent!
----------------------------------------

18. TEST RANDOM BINARY MESSAGE

SUCCESS: Message received matches message sent!
----------------------------------------

19. TEST SERVER INFINITE LOOP (multiple sequential clients to same server)

SUCCESS: Message received matches message sent!
----------------------------------------

20. TEST SERVER QUEUE (overlapping clients to same server)

SUCCESS: Message received matches message sent!

============================================================

TESTS PASSED: 20/20
vagrant@netruc:/vagrant/client_server$
```