第16章 网络编程-练习

1 套接字。面向连接和无连接有什么区别？

2 客户/服务器架构。用你自己的语言描述这个架构，并给出几个例子。

3 套接字。TCP和UDP中，哪一种服务器在接受连接后，把连接交给不同的套接字处理与客户的通讯。

4 客户。修改TCP（tsTclnt.py）和UDP（tsTclnt.py）客户端，让服务器的名字不要在代码里写死，要允许用户指定一个主机名和端口，只有在两个值都没有输入的时候，才使用默认值。

5 Intenet网络和套接字。找到Python Libaray Reference中7.2.2节的示例，TCP客户/服务器程序，实现它并让它运行起来。先运行服务器，然后是客户端。源代码的一个在线版本可以在这里找到：

<http://www.python.org/doc/current/lib/Socket_Example.html>

你认为这个服务器太无聊了，决定要修改服务器，让它能识别以下命令：

date 服务器将返回它的当前时间，i.e.，time.ctime(time.time())os得到操作系统的信息（os.name）

ls 得到当前目录的文件列表 (提示: os.listdir()可以得到目录列表，os.curdir 能得到当前目录) 附加题: 要能接受“ls dir”指令，并返回 dir 目录的文件列表。

做这个作业的时候，你不一定要有网络——你的机器可以跟自己通讯。注：在服务器退出后，要清除绑定后才能再次运行。否则，有可能得碰到“端口已经被使用”（“port already bound”）的错误信息。操作系统一般会在 5 分钟内清除绑定。所以，请耐心等待。

6. 日期时间服务。使用 socket.getservbyname()函数得到 UDP 协议中， “daytime”服务所 对 应 的 端 口 。 请 参 考 getservbyname() 函 数 的 文 档 ， 查 阅 详 细 的 语 法 。（ 即 ：socket.getservbyname.\_\_doc\_\_）。现在，写一个程序发送一个随便什么数据过去，等待回答。一旦你收到了服务器的信息，显示到屏幕上。

7. 半双工聊天。创建一个简单的，半双工的聊天程序。“半双工”的意思是当创建一个连接，服务启动的时候，只有一个人可以打字，另一个人只有在等到有消息通知他输入消息时，才能说话。一旦消息发送出去后，要等到有回复了才能发送下一条消息。一个人是服务端，另一个人是客户端。

8. 全双工聊天。修改你刚才的程序，改成全双工，即两个人可以独立地发送和接收消息。

9. 多用户全双工聊天。再次修改你的程序，把聊天服务改成支持多用户版本。

10. 多用户，多房间全双工聊天。现在把聊天服务改成支持多个用户，多个房间。

11. 网页客户。写一个 TCP 客户，连到你最喜欢的网站的 80 端口（去掉“http://”和其它的后缀信息，只用主机名）。一旦创建了一个连接，发送 HTTP 命令字符串“GET /\n”，把服务器返回的所有数据写到一个文件中。（GET 命令用于得到网页， “/”表示要得到的文件， “\n”把命令发送到服务器）。检查得到的文件的内容，它是什么？怎么检查你得到的数据是否正确？（注：你可能要在命令后加一个或是两个回车，一般来说，一个就可以了。 ）

12. 休眠服务器。创建一个“休眠”服务器，客户可以要求要“休眠”几秒钟。服务器就去做休眠的操作。休眠结束后，返回一个消息给客户，表示结束。客户在收到消息的时候，应该刚好等待了指定的时间。这就是一个简单的“远程过程调用”（“remote procedure call”），即客户发送一个指令，网络另一边的远程的机器执行这个命令。

13. 名字服务器。设计并实现一个名字服务器。这个服务器负责维护一个主机名－端口对的数据库，以及一个描述这个服务器提供的服务的字符串。 选择一个或几个服务器到你的名字服务器上进程“注册”。 （注意，这时，这些服务器是名字服务器的客户）。每一个客户在启动的时候，都不知道它们想要找的服务器的信息。名字服务器的客户也是这样。这些客户应该发送一个请求到名字服务器，说明它们想要得到什么服务。名字服务器返回一个主机名－端口对给客户，客户这时就可以连到合适的服务器来处理它的请求。

附加题：

(1) 在名字服务器中，加入对常用请求的缓冲。

(2) 在名字服务器中，加入日志功能，记录下哪个服务器注册了，客户在请求哪一个服务。

(3) 名字服务器应该周期性地“ping”这些注册了的服务器的对应端口，以确定这些服务器还在运行中。在连续数次 ping 失败后，就把这个服务器从列表中删除。你可以实现一些真实的服务，来注册到你的名字服务器上，或者也可以使用一些哑服务（它们根本不对请求做应答）。

14. 错误检查和优美的退出。 本章中， 我们所有客户和服务器的例子代码都没有做错误检查。我们没有检查用户是否按下了^C 来退出服务，或^D 来结束客户输入，也没有检查 raw\_input()函数得到的输入的合法性，也没有检查网络错误。由于这些弱点，我们很可能会在退出程序的时候，没有关闭套接字，也有可能会丢失数据。选择一对客户/服务器例子，加入足够的错误检查，让程序能正常的退出。比方说会关闭网络连接。

15. 异步和 SocketServer。选取 TCP 服务器例子，使用某一个混合类(mix-in)，让你的程序成为一个异步服务器。测试你的服务器，创建并同时运行多个客户，在服务器的输出里查看你的服务器是否在同时响应多个请求。

16. \*扩展 SocketServer 类。在 SocketServer TCP 服务器代码中，我们不能使用原来的 TCP客户，要做修改。这是因为 SocketServer 类在多个请求之间不保持连接。(a) 从 TCPServer 和StreamRequestHandler 中派生出新的类，重新设计服务器的架构，让服务器能为每个客户只使用一个连接。（而不是每个请求一个连接）(b) 把你之前的问题的解决方案应用到(a)部分，让多个客户的请求可以被并行地处理。