### 网络编程

### 引言

#### 什么是客户端/服务器机构？

服务器是一个软件或硬件，用于向一个或多个客户端提供所需的“服务”。

### 1.2 套接字：通信端点

#### 1.2.1 什么是套接字

应用层通过传输层进行数据通信时，TCP和UDP会遇到同时为多个应用程序进程提供并发服务的问题。多个TCP连接或多个应用程序进程可能需要 通过同一个TCP协议端口传输数据。为了区别不同的应用程序进程和连接，许多计算机操作系统为应用程序与TCP／IP协议交互提供了称为套接字 (Socket)的接口，**区分不同应用程序进程间的网络通信和连接**。

#### 1.2.2 套接字地址：主机与端口

#### 1.2.3 面向连接与无连接

顺序的、可靠的、不会重复的数据传输，传输控制协议（TCP）。

无需建立连接就可以通信，数据到达的顺序、可靠性及不重复性就无法保障，用户数据协议（UDP）。

### 1.3 Python中的网络编程

#### 1.3.1 socket()模块函数

使用socket.socket()函数来创建套接字：

socket.scoket(socket\_family, socket\_type, protocol=0)

tcp\_sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

udp\_sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

#### 1.3.2 套接字对象（内建）方法

|  |  |
| --- | --- |
| 函数 | 描述 |
| 服务器端套接字函数 | |
| bind() | 绑定地址（主机名，端口号对）到套接字 |
| listen() | 开始监听 |
| accept() | 被动接受TCP客户端连接，（阻塞式）等待连接的到来 |
| 客户端套接字函数 | |
| connect() | 主动初始化服务器连接 |
| connect\_ex() | connect()函数的扩展版本，出错时返回出错码，而不是抛出异常 |
| 公共用途的套接字函数 | |
| recv() | 接受tcp数据 |
| send() | 发送TCP数据 |
| sendall() | 完整发送TCP数据 |
| recvfrom() | 接收UDP数据 |
| sendto() | 发送UDP数据 |
| getpeername() | 连接到当前套接字的远端地址（TCP连接） |
| getsockname() | 当前套接字的地址 |
| getsockopt() | 返回指定套接字的参数 |
| setsockopt() | 设置指定套接字的参数 |
| close() | 关闭套接字 |
| 面向模块的套接字函数 | |
| setblocking() | 设置套接字的阻塞和非阻塞模式 |
| settimeout() | 设置阻塞套接字操作的超时时间 |
| gettimeout() | 得到阻塞套接字的超时时间 |
| 面向文件的套接字函数 | |
| fileno() | 套接字的文件描述符 |
| makefile() | 创建一个与该套接字关联的文件对象 |

#### 