UDP编程

服务端例子

```
import socket

HOST = '127.0.0.1'
PORT = 8001

s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)  # 创建套接字时类型得选择SOCK_DGRAM
s.bind((HOST, PORT))  # 绑定套接字到本地IP与端口
# UDP不需要监听连接

print(f'Server start at: {HOST}:{PORT}')

while 1:  # 服务器可以无限循环的接收内容
    data, addr = s.recvfrom(1024)  # 不需要接收连接，直接接收数据
    print(f'Received from {addr}')
    print(data)
    s.sendto(bytes(f'Server received {data}', 'utf-8'), addr)  # 给客户端也发送数据
s.close() # 关闭服务器套接字
```

客户端例子

```
import socket

HOST = '127.0.0.1'
PORT = 8001

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# 不需要用connect方法连接到服务器

while 1:
    cmd = input('Please input msg:')
    s.sendto(bytes(cmd, 'utf-8'), (HOST, PORT))
    data = s.recvfrom(1024)
    print(data)
```

1. 启动服务端

```
❯ python udp_server.py
Server start at: 127.0.0.1:8001
Received from ('127.0.0.1', 52454)
b'test'
Received from ('127.0.0.1', 52454)
b'exit'
```

2. 启动客户端

```
❯ python udp_client.py
Please input msg:test
(b"Server received b'test'", ('127.0.0.1', 8001))
Please input msg:exit
(b"Server received b'exit'", ('127.0.0.1', 8001))
```

Socket 的方法

```
# 服务端使用方法

s.bind(address) 将套接字绑定到地址，在AF_INET下，以tuple(host, port)的方式传入，
                如s.bind((host, port))
s.listen(backlog) 开始监听TCP传入连接，backlog指定在拒绝链接前，操作系统可以挂起的最大连接数，
                  该值最少为1，大部分应用程序设为5就够用了
s.accept() 接受TCP链接并返回（conn, address），其中conn是新的套接字对象，可以用来接收和发送数据，
           address是链接客户端的地址。

# 客户端使用方法

s.connect(address) 链接到address处的套接字，一般address的格式为tuple(host, port)，
                   如果链接出错，则返回socket.error错误
s.connect_ex(address) 功能与s.connect(address)相同，但成功返回0，失败返回errno的值
```

```
s.recv(bufsize[, flag]) 接受TCP套接字的数据，数据以字符串形式返回，buffsize指定要接受的最大数据量，
                        flag提供有关消息的其他信息，通常可以忽略
s.send(string[, flag]) 发送TCP数据，将字符串中的数据发送到链接的套接字，返回值是要发送的字节数量，
                       该数量可能小于string的字节大小
s.sendall(string[, flag]) 完整发送TCP数据，将字符串中的数据发送到链接的套接字，但在返回之前尝试发送所有数据。
                          成功返回None，失败则抛出异常
s.recvfrom(bufsize[, flag]) 接受UDP套接字的数据，与recv()类似，但返回值是tuple(data, address)。
                            其中data是包含接受数据的字符串，address是发送数据的套接字地址
s.sendto(string[, flag], address) 发送UDP数据，将数据发送到套接字，address形式为tuple(ipaddr, port)，
                                  指定远程地址发送，返回值是发送的字节数
s.close() 关闭套接字
s.getpeername() 返回套接字的远程地址，返回值通常是一个包含ipaddr, port的元组
s.getsockname() 返回套接字自己的地址，返回值通常是一个包含ipaddr, port的元组
s.setsockopt(level, optname, value) 设置给定套接字选项的值
s.getsockopt(level, optname[, buflen]) 返回套接字选项的值
s.settimeout(timeout) 设置套接字操作的超时时间，timeout是一个浮点数，单位是秒，值为None则表示永远不会超时。
                      一般超时期应在刚创建套接字时设置，因为他们可能用于连接的操作，如s.connect()
s.gettimeout() 返回当前超时值，单位是秒，如果没有设置超时则返回None
s.fileno() 返回套接字的文件描述
s.setblocking(flag) 如果flag为0，则将套接字设置为非阻塞模式，否则将套接字设置为阻塞模式（默认值）。
                    非阻塞模式下，如果调用recv()没有发现任何数据，或send()调用无法立即发送数据，
                    那么将引起socket.error异常。
s.makefile() 创建一个与该套接字相关的文件
```

延伸阅读

1. 《CCNA学习指南中文版》 第2 - 3章
2. http://man7.org/linux/man-pages/man2/socket.2.html