python网络编程

套接字

套接字是一种具有通讯端点概念的计算机数据网络结构。

合法的端口号范围是0到65535，其中，小于1024的是系统保留端口。

套接字类型

- 面向连接

> 在通讯之前一定要建立一条连接，这种通讯方式也被称为虚电路或流套接字，面向连接的通讯方式是提供了顺序的，可靠的，不会重复的数据传输，而且也不会被加上数据边界，这也意味着，每一个要发送的信息，可能会被拆分成多份，每一份都会不多不少的正确到达目的地。然后被重新按顺序拼接起来，传给正在等待的应用程序。
>
> 实现这种连接的主要协议就是传输控制协议（TCP），要创建TCP套接字就得在创建的时候，指定套接字类型为SOCK_STREAM，由于这些套接字使用internet协议来查找网络中的主机，这样形成的整个系统，一般会有这两个协议（TCP和IP）来提及，即TCP/IP。

- 无连接

> 无需建立连接就可以通讯，数据到达的顺序，可靠性以及数据不可靠性就无法保证了，数据包会保留数据边界，这就表示，数据不会像面向连接的协议那样被拆分成小块。
>
> 实现这种连接的主要协议就是用户数据报协议，要创建UDP套接字就得在创建的时候，指定套接字类型为SOCK_DGRM。

套接字对象内建方法

服务器端套接字函数

函数	描述
s.bind()	绑定地址到套接字
s.listen()	开始TCP监听
s.accept()	被动接受TCP客户的连接，（阻塞式）等待连接的到来

客户端套接字函数

函数	描述
s.connect()	主动初始化TCP服务器连接
s.connect_ex()	connect()函数的扩展版本，出错时返回出错码，而不是抛异常

公共用途的套接字函数

函数	描述
s.recv()	接受TCP数据
s.send()	发送TCP数据
s.sendall()	完整发送TCP数据
s.recvfrom()	接受UDP数据
s.sendto()	发送UDP数据
s.getpeername()	连接到当前套接字的远端的地址
s.getsockname()	当前套接字的地址
s.getsockopt()	返回指定套接字的参数
s.setsockopt()	设置指定套接字的参数
s.close()	关闭套接字

Block-Oriented socket methods

函数	描述
s.setblocking()	设置套接字的阻塞与非阻塞模式
s.settimeout()	设置阻塞套接字操作的超时时间
s.gettimeout()	得到阻塞套接字操作的超时时间

面向文件的套接字函数

函数	描述
s.fileno()	套接字的文件描述符
s.makefile()	创建一个与该套接字关联的文件

论优美的推出和调用服务器的close()函数

友好的退出的一个方法就是把服务器的无限循环放在一个try-except语句的try子句当中，并捕获EOFError和KeyboardInterrupt异常，在异常处理子句中，调用close()函数关闭服务器的套接字。

twisted框架介绍

twisted是一个完全事件驱动的网络框架，它允许你使用和开发完全异步的网络应用程序和协议。

服务器实例

```
from twisted.internet import protocol, reactor
from time import ctime

PORT = 21567

class TSServProtocol(protocol.Protocol):
    def connectionMade(self):
        clnt = self.clnt = self.transport.getPeer().host
        print '...connected from:', clnt
    def dataReceived(self, data):
        self.transport.write('[%s] %s' % (
        ctime(), data))

factory = protocol.Factory()
factory.protocol = TSServProtocol
print 'waiting for connection...'
reactor.listenTCP(PORT, factory)
reactor.run()
```

客户端实例

```
#!/usr/bin/env python

from twisted.internet import protocol, reactor

HOST = 'localhost'
PORT = 21567

class TSClntProtocol(protocol.Protocol):
    def sendData(self):
        data = raw_input('> ')
        if data:
            print '...sending %s...' % data
            self.transport.write(data)
        else:
            self.transport.loseConnection()

    def connectionMade(self):
        self.sendData()

    def dataReceived(self, data):
        print data
        self.sendData()
```

```
class TSClntFactory(protocol.ClientFactory):
    protocol = TSClntProtocol
    clientConnectionLost = clientConnectionFailed = \
        lambda self, connector, reason: reactor.stop()

reactor.connectTCP(HOST, PORT, TSClntFactory())
reactor.run()
```