TCP

笔记本：	网络编程		
创建时间：	2018/5/10 14:39	**更新时间：**	2018/5/10 16:56
作者：	ly		
标签：	SOCK_STREAM, TCP		

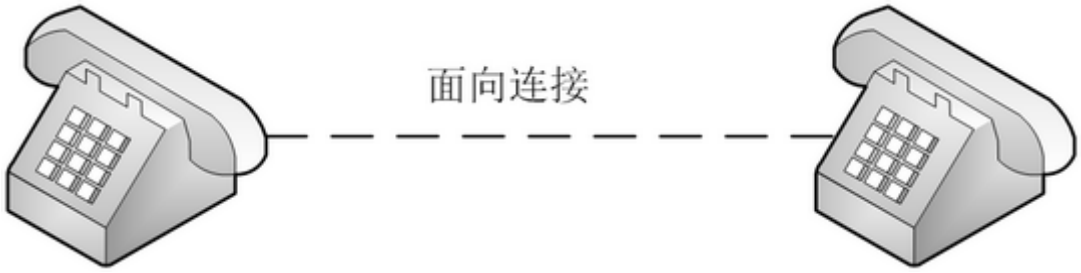
TCP

TCP简介

TCP（Transmission Control Protocol 传输控制协议）是一种面向连接的、可靠的、基于字节流的传输层通信协议。

TCP通信需要经过创建连接、数据传送、终止连接三个步骤。

TCP通信模型中，在通信开始之前，一定要先建立相关的链接，才能发送数据，类似于生活中打电话。

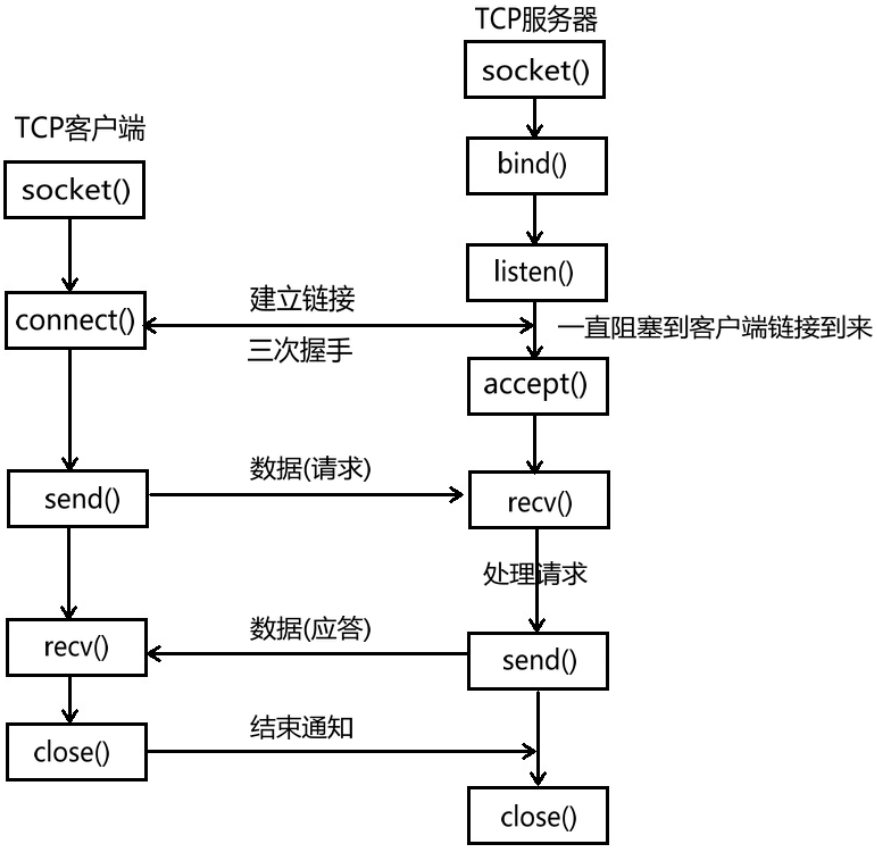


采用发送应答机制：TCP发送的每个报文必须得到对方**应答**才认为传输成功
超时重传：发送端发一个报文后就启动定时器，规定时间内没有收到应答，则重发
错误效验：用验证和函数来检查数据是否错误；发送和接收都会启动
流量控制和阻塞管理：用来避免主机发送得过快而使接收方来不及完全手下

TCP和UDP的**不同点**：

面向连接（确认有创建三方交握，连接已创建才作传输。）
有序数据传输
重发丢失的数据包
舍弃重复的数据包
无差错的数据传输
阻塞/流量控制

TCP通信模型（C/S模型）



TCP客户端：

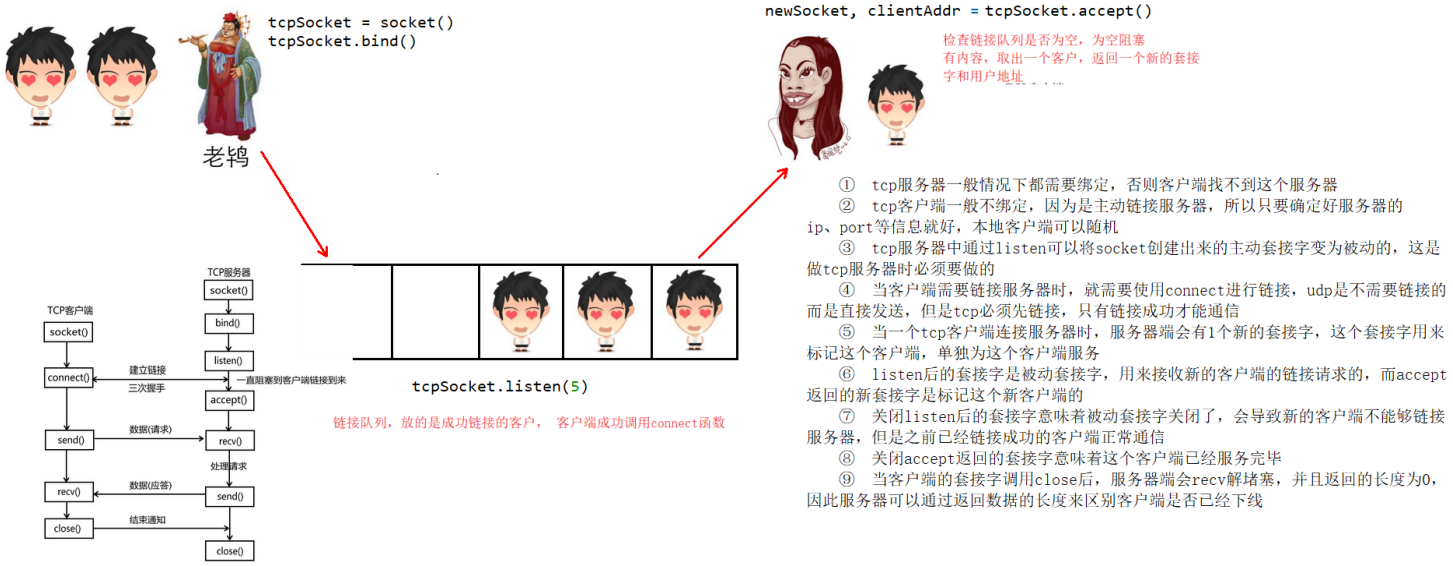
```
from socket import *     # 导入模块中所有函数

tcp = socket(AF_INET, SOCK_STREAM)    # 创建套接字

address = ("192.168.25.71", 8082)
tcp.connect(address)                    # 链接服务器 此处的ip,port为已经启动的服务器地址信息

sendData = input("请输入要发送的内容：")

tcp.send(sendData.encode("utf-8"))    # 因为上面已经绑定到了服务器，所以send()括号内直接为内容即可发送

recvData = tcp.recv(1024)    # 接收数据    接收到的recvData也为纯数据（没有地址信息）

print("接收数据为：", recveData.decode("utf-8")

tcp.close()
```

只需要建立一次连接之后，后面只需要发送send()内容、接收recv()即可，无需再附带地址信息。

TCP服务端

- socket创建套接字
- bind绑定ip和port　（服务端号码不变10086）
- listen使套接字变为被动链接，等待链接　（连接10086总机服务分配人工服务即：新的套接字）
- accept取出客户端链接　（直接与客服交流）
- recv/send接收客户端发送的数据

```
import socket    # 导入模块

tcp = socket.socket(socket.AF_INET, socket.SOCK_STREAM)    # 创建套接字

tcp.bind(("", 8888))    # 绑定本地网络信息

tcp.listen(128)    # 将套接字变为被动，可以接收别人的链接。创建链接队列  链接队列最大为128位

clientSocket, clientAddr = tcp.accept()    # 取出队列中的客户端，建立新的套接字，只能单个服务。后面一直用新的套接字

recvData = client.recv(1024)    # 接收对方发过来的数据

print(clientAddr, ">>>>>>>", recvData.deconde("utf-8"))

clientSocket.send("thanks".encode("utf-8"))    # 给客户端回数据

clientSocket.close()    # 关闭服务套接字，意味着不再为客户端服务了

tcp.close()    # 关闭监听套接字
```



TCP应用：服务器循环接收同一个用户的数据

```
import socket

tcp = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

tcp.bind(("", 8888))

tcp.listen(128)

new_socket, cli_addr = tcp.accept()

print(cli_addr, "链接上来咯")    # 当有客户链接上上来，就会打印这句

while True:
    recv_Data = new_socket.recv(1024)    # 重复接收数据
    if len(recv_data) > 0:                # 判断接收的数据是否为空
        print(cli_addr, ">>>>", recv_data.decode())
    else:
        print("对方已经下线了")        # 接收到的数据为空时，自动端口
        break
```

```
new_socket.close()
tcp.close()
```

TCP应用：服务循环为多个用户服务

```
import socket

# 创建TCP套接字(监听、链接套接字)
tcp_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# 绑定
tcp_socket.bind(("", 9988))

# 监听，将套接字变为被动，系统创建一个链接队列
tcp_socket.listen(128)

# 取出成功链接的客户，返回一个新的套接字（服务套接字），用户地址，如果没有客户连接，也会阻塞
while True:
    new_socket, cli_addr = tcp_socket.accept()
    print(cli_addr, "成功连接")

    # 接收客户端的数据，客户没有发送内容，阻塞，注意，使用服务套接字接收内容
    recv_data = new_socket.recv(1024)
    print(cli_addr, " >>>>>>>> ", recv_data.decode())

    # 给对方回复数据，使用新的套接字
    new_socket.send("ok".encode())

    # 关闭套接字
    new_socket.close()    # 每次循环都会关闭服务套接字

tcp_socket.close()
```

案例：文件下载器

功能：客户端发送要打开的文件名，接收服务端返回的文件内容，写入新的文件中
服务端接收文件名，打开文件内容，发送数据给客户端。

客户端：

```
import socket


def decend(recvInfo, fileName):
    """将内容写入新文件"""
    newName = "new" + fileName    # 根据源文件名生成新文件名

    print(fileName)
    print(recvInfo)

    with open(newName, "w") as f:
        f.write(rcvInfo)
    print("over")


tcp = socket.socket(socket.AF_INET, socke.SOCK_STREAM)
tcp.connect(("192.168.25.52", 8080))

while true:
    fileName = input("请输入要下载的文件名字：")    # 需要带后缀，以便后面生成文件
    if len(send) > 0 :
```

```
        tcp.send(send.encode("utf-8"))    # 发送文件名让服务器查找文件内容
    else:
        break

    recvInfo = tcp.recv(1024)    # 等待接收服务器发送过来的文件内容
    defend(recvInfo.decode(), fileName)    # 把内容写入新的文件
    print("服务器发回的数据：%s" % recvInfo.decode("utf-8"))

tcp.close()
```

服务端：

```
import socket


def findname(filename):
    """查找文件，读取文件内容"""
    try:
        with open(filename, "rb") as f:        # rb -- 以二进制的形式打开文件
            conten = f.read()
        print("查找到该文件了")
        return content            # 返回查找到的内容
    except:
        print("文件不存在")


tcp = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

tcp.bind()

tcp.listen(128)

while True:    # 可以让客户端重复连接上来

    newTcp, addr = tcp.accept()
    print("%s连接上来咯" % str(addr))

    while True:    # 可以让客户端重复查找文件内容
            recvdata = newTcp.recv(1024)        # 接收用户发送过来的文件名
            if len(recvdata.decode("utf-8")) > 0:
                filecontend = findname(recvdata)
                newTcp.send(filecontend)
            else:
                break
    newTcp.close()    # 用户不再发送文件，则关闭服务套接字
tcp.close()        # 用户断开连接，则关闭监听套接字 意味着不再接收任何客户端连接
```