Day06-网络编程

今日课程学习目标

```
常握 网络编程的基础知识：IP、端口、TCP协议、Socket
熟悉 TCP网络程序的开发流程
```

今日课程内容大纲

```
# 1. 网络编程基础知识
    IP地址
    端口和端口号
    TCP协议
    socket介绍
# 2. TCP网络程序开发
    TCP网络程序开发流程
    TCP网络程序开发示例1：基本流程
        服务端程序
        客户端程序
    TCP网络程序开发示例2：文件传输
        服务端程序
        客户端程序
```

知识点1：IP 地址介绍【重点】

思考题：

```
假设你在上海，你在自己电脑上打开了微信，你女朋友在北京，她在她的电脑上打开了微信，此时，你用微信给女朋友发送了一条消息，这条信息是怎么到你女朋友的微信上的？？？
```

首先，信息肯定是通过网络进行传输的，但是网络上有那么多台电脑，信息是怎么到你女朋友的电脑上呢?

> IP地址的作用：**标识网络中唯一的一台设备(比如你女朋友的电脑)**

IP地址的表现形式：

- `IPv4`：点分十进制，由32个二进制位组成

```
139.227.220.247
```

- `IPv6`：冒号十六进制，由128个二进制位组成

```
2345:0425:2CA1:0000:0000:0567:5673:23b5
```

查看电脑的 IP 地址：

- Windows：`ipconfig`
- Linux 和 Mac：ifconfig

检查电脑网络是否正常：

- `ping www.baidu.com` 是否能连接互联网
- `ping 127.0.0.1` 本地回环地址，检查网卡是否正常

知识点2：端口和端口号介绍【重点】

思考题：

```
你发送的微信消息，通过IP地址传递到你女朋友的电脑之后(PS：你女朋友电脑开启了很多程序：QQ、微信等)，那么这个消息是怎么传递给女朋友的微信程序，而不是QQ程序呢？
```

> 端口：标识一台设备的一个网络应用程序

一台计算机最多有65536个端口，为了区分端口，每个端口都有编号：`0-65535`，编写网络程序时，可以设置网络程序使用的端口号。

端口号分类：

- 知名端口号：范围是0到1023，系统预留的一些端口号，不建议使用
- 动态端口号：范围是1024到65535，建议使用
  - 一些动态端口号被一些常见程序占用了，自己写网络程序时，也不建议使用，比如：MySQL数据库 3306、redis数据库 6379、HDFS 9870等...
  - 如果程序员开发的网络程序没有设置端口号，操作系统会在动态端口号这个范围内随机生成一个给应用程序使用

> 当运行一个网络程序默认会有一个端口号，当这个程序退出时，所占用的这个端口号就会被释放。

知识点3：TCP协议介绍【重点】

思考题：

```
你发送的微信消息，在网络上进行传输时，怎么能保证不会丢失，一定能发送到你女朋友的微信呢？
```

这就涉及到非常著名的**TCP网络协议**。

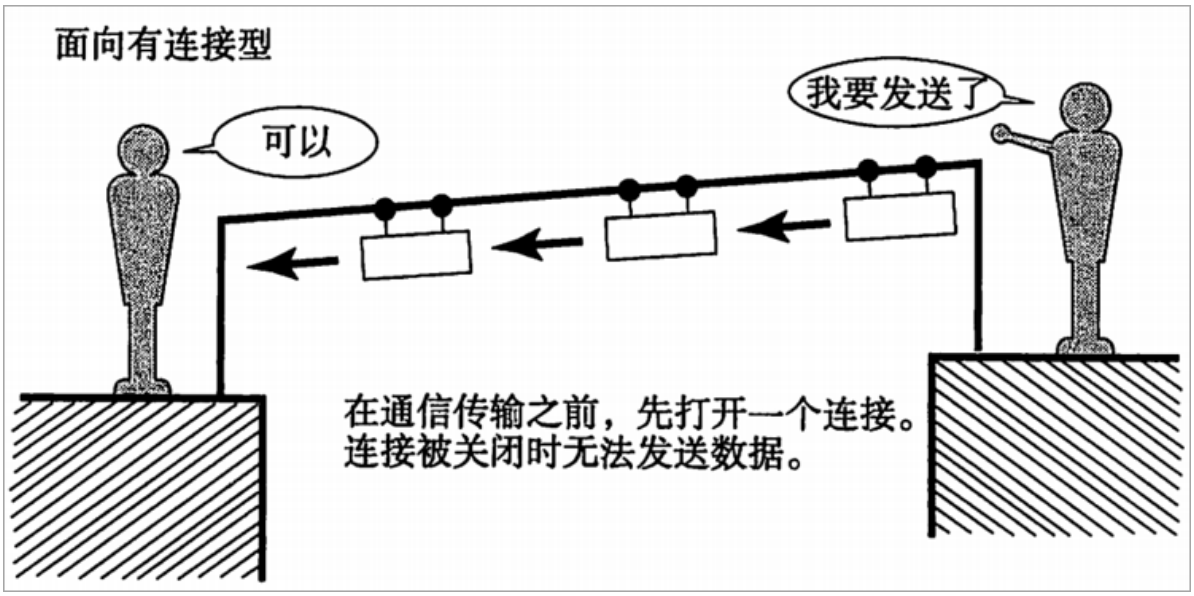
TCP概念：

> 概念：TCP(Transmission Control Protocol) 是一种网络传输控制协议，它是一种**面向连接的、可靠的、基于字节流的传输层通信协议**。

TCP 的特点：

- 面向连接

- 可靠传输：保证数据不丢包\不乱序



TCP 通信步骤：

- 创建连接
- 传输数据
- 关闭连接

知识点4：socket介绍【重点】

思考题：

```
1    知道了IP、端口、TCP协议的概念之后，如何自己编写一个网络程序呢？
```

这就涉及都 socket 网络编程接口。

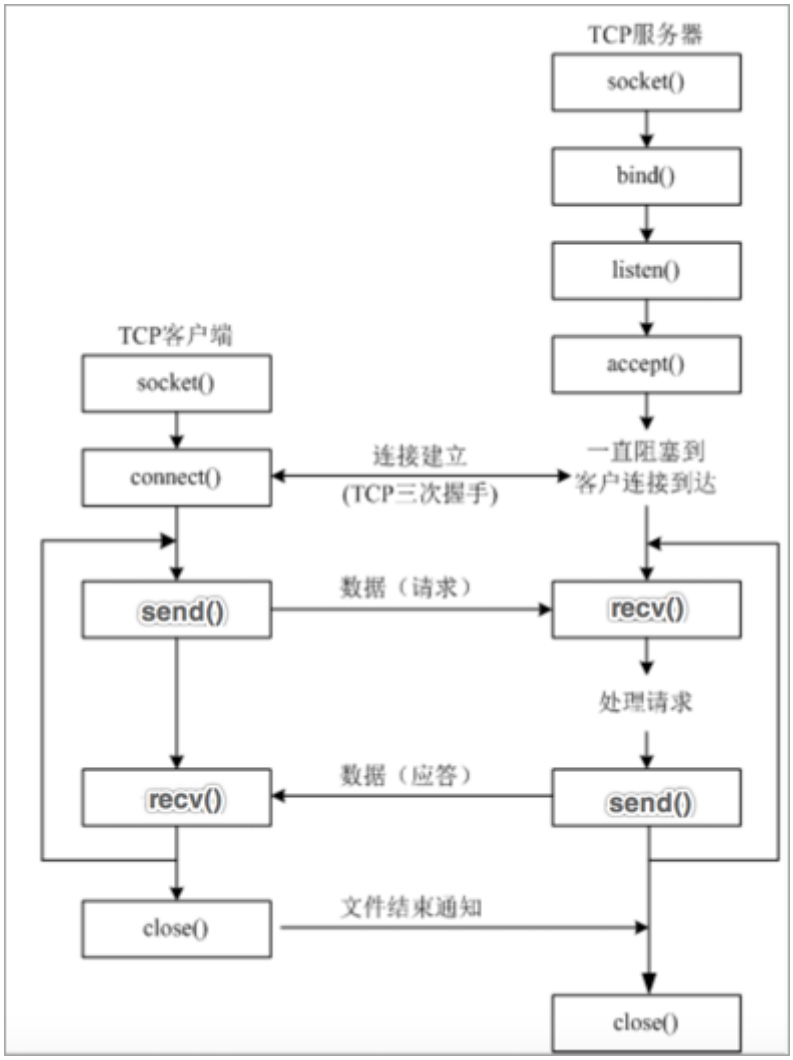
[**百度百科**] 所谓套接字(Socket)，就是对网络中不同主机上的应用进程之间进行双向通信的端点的抽象。一个套接字就是网络上进程通信的一端，提供了应用层进程利用网络协议交换数据的机制。从所处的地位来讲，套接字上联应用进程，下联网络协议栈，是应用程序通过网络协议进行通信的接口，是应用程序与网络协议栈进行交互的接口。

> socket (简称 套接字) 是**进程之间通信一个工具**，是操作系统提供的网络编程接口，屏蔽通过TCP/IP及端口进行网络数据传输的细节

- 作用：**进程之间的网络数据传输**
- 使用场景：**网络相关的应用程序或者软件都使用到了 socket**

知识点5：TCP网络程序开发流程【熟悉】

TCP网络程序开发流程：



1）客户端程序的开发流程：主动连接的

```
1. 创建套接字对象
2. 连接服务端
3. 发送数据
4. 接受数据
5. 关闭连接
```

2）服务端程序的开发流程：等待连接的

```
1. 创建监听套接字对象
2. 绑定端口
3. 开始监听
4. 等待连接
5. 发送数据
6. 接受数据
7. 关闭连接
```

知识点6：TCP服务端程序开发【熟悉】

TCP服务端程序开发：

```python
import socket

# 创建服务端监听套接字
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# 监听套接字绑定地址和端口
```

```python
server.bind(('127.0.0.1', 8080))

# 监听套接字开始监听，准备接收客户端的连接请求
server.listen(128)
print('服务器开始监听...')

# 接收客户端的连接请求
# service_client：专门和客户端通信的套接字
# ip_port：客户端的 IP 地址和端口号
service_client, ip_port = server.accept()
print(f'服务器接收到来自{ip_port}的请求')

# 接收客户端发送的消息，最多接收 1024 给字节
recv_msg = service_client.recv(1024) # 接收的消息为 bytes 类型
print('客户端发送的消息为：', recv_msg.decode())

# 给客户端发送响应消息
send_msg = input('请输入响应的消息：')
service_client.send(send_msg.encode())

# 关闭和客户端通信的套接字
service_client.close()
# 关闭服务器监听套接字
server.close()
```

知识点7：TCP客户端程序开发【熟悉】

TCP客户端程序开发：

```python
import socket

# 创建客户端 socket 套接字对象
# socket.AF_INET：表示 IPV4
# socket.SOCK_STRAM：表示 TCP 传输协议
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# 客户端请求和服务端程序建立连接
client.connect(('127.0.0.1', 8080))
print('客户端连接服务器成功！')

# 客户端向服务器发生消息
send_msg = input('请输入发送的消息：')
client.send(send_msg.encode()) # 注意：send 函数参数需要为 bytes 类型

# 客户端接收服务器响应的消息，最多接收 1024 个字节
recv_msg = client.recv(1024) # 接收的消息为 bytes 类型
print('服务器响应的消息为：', recv_msg.decode())

# 关闭客户端套接字
client.close()
```

知识点8：文件传输-服务端程序【熟悉】

需求：编写一个网络程序，实现将客户端的文件传递到服务端的电脑，并保存到服务器本地。

服务端程序代码：

```python
import socket
```

```python

# 创建服务端监听套接字
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# 监听套接字绑定地址和端口
server.bind(('127.0.0.1', 8080))

# 监听套接字开始监听，准备接收客户端的连接请求
server.listen(128)
print('服务器开始监听...')

# 接收客户端的连接请求
# service_client：专门和客户端通信的套接字
# ip_port：客户端的 IP 地址和端口号
service_client, ip_port = server.accept()
print(f'服务器接收到来自{ip_port}的请求')

# ① 创建文件对象
f = open("./dest/小电影.mp4", "wb")

# ② 循环接收客户端发送过来的数据，直到客户端发送完毕
while True:
    # ③ 每次接受 1024 个字节
    data = service_client.recv(1024)
    # ④ 将接收到的数据写入到文件中
    f.write(data)
    # ⑤ 读取结束，结束循环
    if len(data) == 0:
        break

# ③ 关闭文件
f.close()
print('服务器接收文件完成！')

# 关闭和客户端通信的套接字
service_client.close()
# 关闭服务器监听套接字
server.close()
```

知识点9：文件传输-客户端程序【熟悉】

客户端程序代码：

```python
import socket

# 创建服务端监听套接字
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# 监听套接字绑定地址和端口
server.bind(('127.0.0.1', 8080))

# 监听套接字开始监听，准备接收客户端的连接请求
server.listen(128)
print('服务器开始监听...')

# 接收客户端的连接请求
# service_client：专门和客户端通信的套接字
# ip_port：客户端的 IP 地址和端口号
```

```python
service_client, ip_port = server.accept()
print(f'服务器接收到来自{ip_port}的请求')

# ① 创建文件对象
f = open("./dest/小电影.mp4", "wb")

# ② 循环接收客户端发送过来的数据，直到客户端发送完毕
while True:
    # ③ 每次接受 1024 个字节
    data = service_client.recv(1024)
    # ④ 将接收到的数据写入到文件中
    f.write(data)
    # ⑤ 读取结束，结束循环
    if len(data) == 0:
        break

# ③ 关闭文件
f.close()
print('服务器接收文件完成！')

# 关闭和客户端通信的套接字
service_client.close()
# 关闭服务器监听套接字
server.close()
```

知识点10：文件传输-recv方法的阻塞和解阻塞【熟悉】

当通过 socket 套接字调用 recv 方法接收数据时，如果对端没有发送过来数据时，recv 方法会一直阻塞。

recv 方法解阻塞的 3 种情况：

1）对端 发送过来了数据

```
socket套接字对象.send(发送数据)
```

2）对端 socket 套接字被关闭

```
scoket套接字对象.close()
```

3）对应 socket 套接字关闭了输出流

```
socket套接字对象.shutdown(socket.SHUT_WR)
```

知识点11：文件传输-客户端vs服务器程序recv阻塞处理【熟悉】

需求：在文件传输服务器和客户端程序的代码基础上，增加如下功能：

1）服务器接收文件内容完毕之后，给客户端回复消息：接收完成！

2）客户端发送文件内容完毕之后，接收服务器的回复消息

服务器代码：

```python
import socket

# 创建服务端监听套接字
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# 监听套接字绑定地址和端口
server.bind(('127.0.0.1', 8080))
```

```python

# 监听套接字开始监听，准备接收客户端的连接请求
server.listen(128)
print('服务器开始监听...')

# 接收客户端的连接请求
# service_client：专门和客户端通信的套接字
# ip_port：客户端的 IP 地址和端口号
service_client, ip_port = server.accept()
print(f'服务器接收到来自{ip_port}的请求')

# ① 创建文件对象
f = open("./dest/小电影.mp4", "wb")

# ② 循环接收客户端发送过来的数据，直到客户端发送完毕
while True:
    # ③ 每次接受 1024 个字节
    data = service_client.recv(1024)
    # ④ 将接收到的数据写入到文件中
    f.write(data)
    # ⑤ 读取结束，结束循环
    if len(data) == 0:
        break

# ⑥ 关闭文件
f.close()
print('服务器接收文件完成！')

# ⑦ 给客户端回复消息
service_client.send('接收完成'.encode())

# 关闭和客户端通信的套接字
service_client.close()
# 关闭服务器监听套接字
server.close()
```

客户端代码：

```python
import socket

# 创建客户端 socket 套接字对象
# socket.AF_INET：表示 IPV4
# socket.SOCK_STRAM：表示 TCP 传输协议
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# 客户端请求和服务端程序建立连接
client.connect(('127.0.0.1', 8080))
print('客户端连接服务器成功！')

# ① 创建文件对象
f = open("./src/小电影.mp4", "rb")

# ② 循环读取文件中的数据
while True:
    # ③ 每次从文件中读取1024个字节
    data = f.read(1024)
    # ④ 将读取的数据发送给服务器
    client.send(data)
```

```
        # ⑤ 文件读取结束，结束循环
        if len(data) == 0:
            # 客户端关闭输出流，让服务端的 recv 解阻塞
            # 注意：如果客户端关闭输出流，会导致服务端和客户端程序的 recv 都阻塞
            client.shutdown(socket.SHUT_WR)
            break

    # ⑥ 关闭文件
    f.close()
    print('客户端发送文件完成！')
    # ⑦ 接收客户端回复的消息
    res_msg = client.recv(1024)
    print('服务器回复：', res_msg.decode())

    # 关闭客户端套接字
    client.close()
```