实验报告（三）

李国楷 22  26

实验目的

理解 TCP 和 UDP 协议主要特点
掌握 socket 的基本概念和工作原理，编程实现 socket 通信

实验内容、结果、分析：

任务 1+任务 2

1.客户机运行情况

```
/home/kenlee/Desktop/lab3/cmake-build-debug/ciln localhost 7
Myself: hello
Server: hello
Myself: nihao
Server: nihao
Myself: kenlee
Server: kenlee
Myself: bye
Server: bye

进程已结束，退出代码为 0
```

2.不同 backlog 的情况下：
backlog=0:

此时只有 2 个客户机和服务器建立连接，剩下两个则处于 SYN_SENT

backlog=1:

```
kenlee@kenlee:~$ netstat -anp|grep "2000"
（并非所有进程都能被检测到，所有非本用户的进程信息将不会显示，如果想看到所有信息，则必须切换到 root 用户）
tcp        2      0 0.0.0.0:2000            0.0.0.0:*               LISTEN      7117/sev
tcp        0      0 127.0.0.1:34732         127.0.0.1:2000          ESTABLISHED 7125/ciln
tcp        0      0 127.0.0.1:2000          127.0.0.1:57964         ESTABLISHED -
tcp        0      0 127.0.0.1:57966         127.0.0.1:2000          ESTABLISHED 7139/ciln
tcp        0      0 127.0.0.1:52926         127.0.0.1:2000          TIME_WAIT   -
tcp        0      0 127.0.0.1:57964         127.0.0.1:2000          ESTABLISHED 7132/ciln
tcp        0      0 127.0.0.1:2000          127.0.0.1:57966         ESTABLISHED -
tcp        0      1 127.0.0.1:57970         127.0.0.1:2000          SYN_SENT    7154/ciln
tcp        0      0 127.0.0.1:2000          127.0.0.1:34732         ESTABLISHED 7117/sev
unix  3      [ ]         数据报   已连接     20007    -
unix  2      [ ]         数据报   已连接     20002    -
unix  3      [ ]         数据报   已连接     20008    -
unix  3      [ ]         数据报   已连接     20009    -
unix  3      [ ]         数据报   已连接     20006    -
```

此时有 3 个客户急和服务器建立连接，有 1 个客户急处于 SYN_SENT 状态

分析：

内核会为 socket 维护两个队列：未完成连接队列和已完成连接队列。而 listen 函数中的参数 backlog 则限制了已完成连接队列的长度。但是 backlog 参数的意义没有一个完全统一的标准，不同系统的解释不同。

backlog = 0 时：

当启动第一个服务端时，listen 接收到该服务端的连接后，一个已完成的连接会进入已完成连接队列，代码继续向下执行， accept 函数则会从该队列中取走该连接，所以此时队列为空。再运行第二个客户端，通过观察端口发现连接是 ESTABLISHED 状态，说明该连接处于已完成连接队列中。再运行第三个客户端，客户端处于 SYN_SENT 状态，连接未被建立，说明队列已满。因此，已完成连接队列的长度是 backlog + 1 .

backlog = 1 时的分析类似。

3.端口字节转换：

```
    server_addr.sin_port=*argv[1];
    //  server_addr.sin_port = htons(atoi(argv[1]));
```

```
/home/kenlee/Desktop/lab3/cmake-build-debug/sev 2000
```

```
/home/kenlee/Desktop/lab3/cmake-build-debug/ciln localhost 2000
connect: Connection refused

进程已结束，退出代码为 1
```

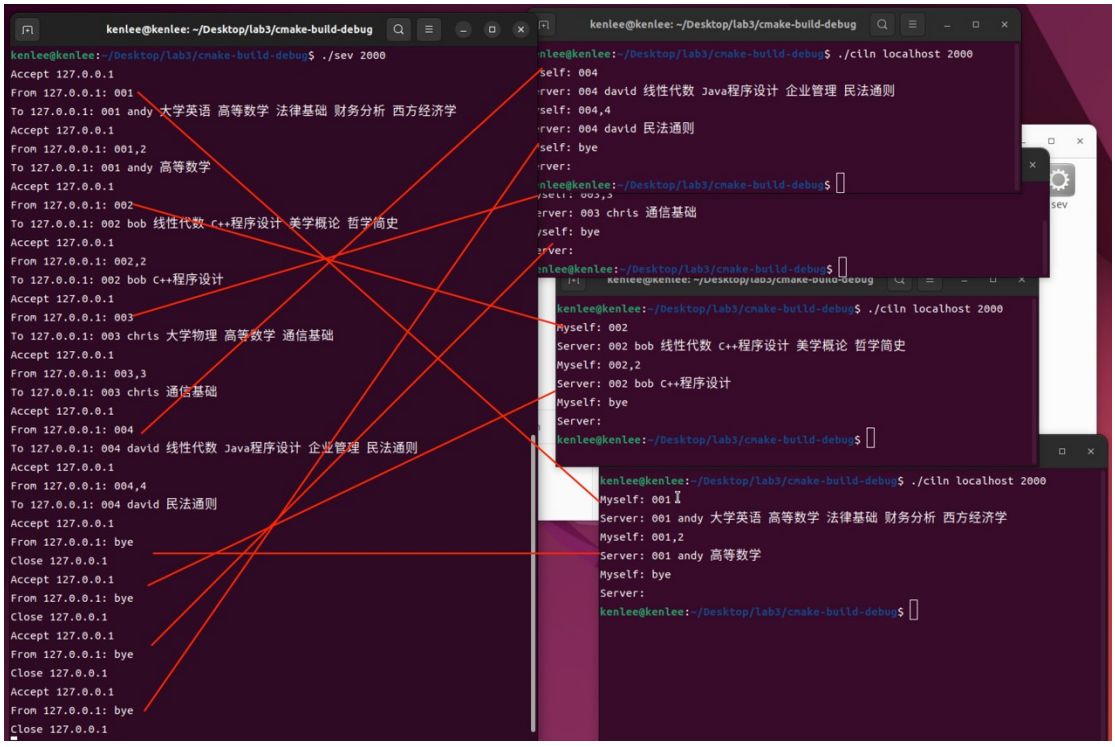
```
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:53255           0.0.0.0:*               LISTEN      4682/sev
```

不进行字节转换之后服务器的监听端口变成了 53255 而不是 2000
原因：

在计算机网络中，字节顺序转换涉及到大端序（Big-Endian）和小端序（LittleEndian）之间的转换。不同的系统使用不同的字节顺序，因此为了确保正确的数据传输，需要进行字节顺序的转换。由于主机和网络之间可能具有不同的字节顺序。为了协调这种差异，需要进行字节顺序转换，本实验中我们需要对端口进行字节顺寻转换，确保端口号能够被正确地解释。若不进行字节顺序转换，可能导致接收方解释端口号时发生错误。

任务 3： 课程表查询服务器(TCP 并发)

1.运行结果



2.应该关闭

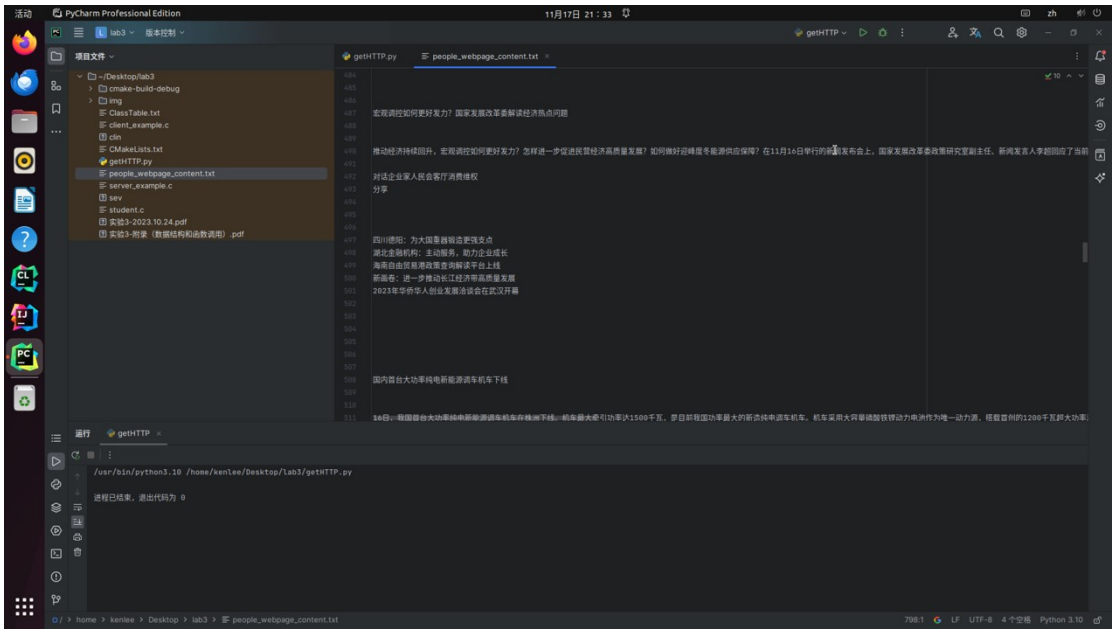
```
Accept 127.0.0.1
From 127.0.0.1: bye
Close 127.0.0.1
Accept 127.0.0.1
From 127.0.0.1: bye
Close 127.0.0.1
```

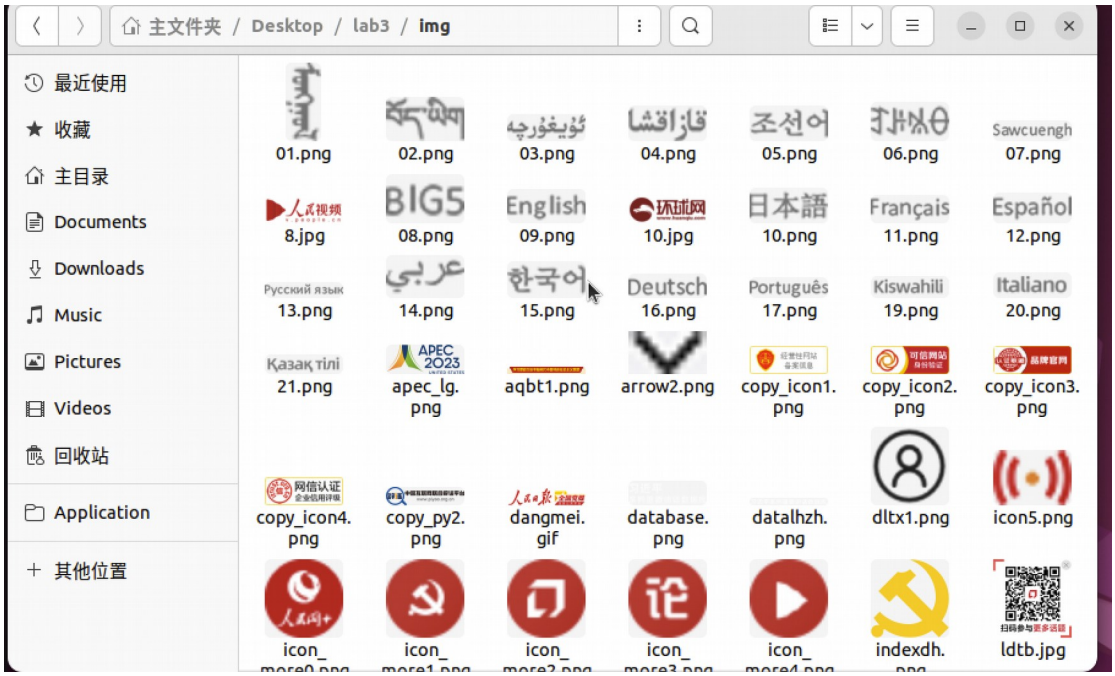
```
kenlee@kenlee:~/Desktop/lab3$ netstat -anp|grep "2000"
（并非所有进程都能被检测到，所有非本用户的进程信息将不会显示，如果想看到所有信息，则必须切换到 root 用户）
tcp        0      0 0.0.0.0:2000            0.0.0.0:*               LISTEN      6787/sev
tcp        0      0 127.0.0.1:56442         127.0.0.1:2000          ESTABLISHED 6811/ciln
tcp        0      0 127.0.0.1:2000          127.0.0.1:56440         ESTABLISHED 6787/sev
tcp        0      0 127.0.0.1:2000          127.0.0.1:56442         ESTABLISHED 6787/sev
tcp        0      0 127.0.0.1:56440         127.0.0.1:2000          ESTABLISHED 6800/ciln
unix  3      [ ]         流         已连接     22000    -                    /run/dbus/system_bus_socket
```

如图，如果不关闭，即使发送 bye 命令，服务器输出 close，但是在监听视角依然是建立连接的状态，而如果断开 socket 就不会出现这样的问题。断开 socket 会如下图显示 TIME_WAIT

```
kenlee@kenlee:~/Desktop/lab3$ netstat -anp|grep "2000"
（并非所有进程都能被检测到，所有非本用户的进程信息将不会显示，如果想看到所有信息，则必须切换到 root 用户）
tcp        0      0 0.0.0.0:2000            0.0.0.0:*               LISTEN      6940/sev
tcp        0      0 127.0.0.1:2000          127.0.0.1:38702         TIME_WAIT   -
tcp        0      0 127.0.0.1:2000          127.0.0.1:38706         TIME_WAIT   -
```

任务 4： 获取网页内容





方法：
1.用 requests.get()建立连接
用 soup = BeautifulSoup(response.content, 'html.parser', from_encoding='gbk')以 gbk 格式获取网页内容
用 webpage_content = soup.get_text() 去除 html 之类的标签，只要网页内容
然后写入本地的 txt 文件就行

2.用 img_links = [urljoin(url, img["src"]) for img in soup.find_all("img")]获取图片下载链接
然后下载图片到本地的 img 文件夹就行

实验小结、感想：

实验小结：
本次实验旨在深入理解 TCP 和 UDP 协议的特点，掌握 socket 的基本概念和工作原理，并通过编程实现 socket 通信。以下是针对实验任务的总结和观察结果：

1. 不同 backlog 值下的影响：
当设置 backlog 为 0 时，已完成连接队列为空，且当第三个客户端尝试连接时，队列已满，导致连接未建立。
设置 backlog 为 1 时，已完成连接队列能够容纳一个连接请求，第三个客户端的连接处于 SYN_SENT 状态，仍等待连接的建立。
结论：backlog 值决定了已完成连接队列的长度，影响着能够同时处理的等待连接数。

2. 端口字节转换的必要性：

在网络通信中，不同系统的字节顺序可能不同，为了确保正确传输数据，需要进行字节顺序的转换。

未进行字节转换可能导致接收端解释端口号时出现错误，因此对端口号进行字节顺序转换是必要的。

3. TCP 并发服务器的运行和关闭：

在服务器运行期间，即使发送了关闭命令，服务器仍然保持监听状态，需手动断开 socket 才能使服务器完全关闭。

若不断开 socket，服务器可能在监听视角仍显示建立连接的状态，但观察到的是 TIME_WAIT 状态，这可能是因为连接在服务器端需要一定时间被完全关闭和释放。

4. 获取网页内容和图片下载：

使用 requests 库建立连接获取网页内容，并通过 BeautifulSoup 解析内容，将网页内容写入本地 txt 文件，同时提取图片链接并下载到本地 img 文件夹。

感想：

通过本次实验，我深入了解了 TCP 和 UDP 协议的特点，学习了 socket 的基本概念和工作原理。实验中的不同 backlog 值对连接数量的影响以及端口字节转换的重要性让我更清楚地理解了网络通信的细节和实际运作。同时，通过构建 TCP 并发服务器和获取网页内容、图片下载等任务，我加深了对 socket 编程和网络数据传输的实际应用的认识，这些知识对于今后在网络通信和编程方面的应用将会起到重要的指导作用。

相关代码文档和文件记录

见附件