四川大学
SICHUAN UNIVERSITY

题　　目 多线程服务器—开发过程文档

学生姓名 夏琦妮

课　　程 计算机网络

学　　号 2019141240178

专　　业 计算生物

指导教师 宋万忠

二〇二一　年　6 月 15 日

测试结果

1. 服务端

服务端运行正常，且能够实现多线程功能，在处理一个客户端请求的同时，能够对另一个客户端的请求进行响应，并执行响应的处理逻辑。效果如下图所示。

```
the length of request is 66
random number request run_index 1  reader 6
<socket.socket fd=968, family=AddressFamily.AF_INET, type=Soc
循环完成3次!
循环开始4次!
('127.0.0.1', 7897) comes
the length of request is 43
weather info beijing
the length of request is 66
random number request run_index 1  reader 7
('127.0.0.1', 7897) bye
the length of request is 66
random number request run_index 1  reader 8
the length of request is 66
```

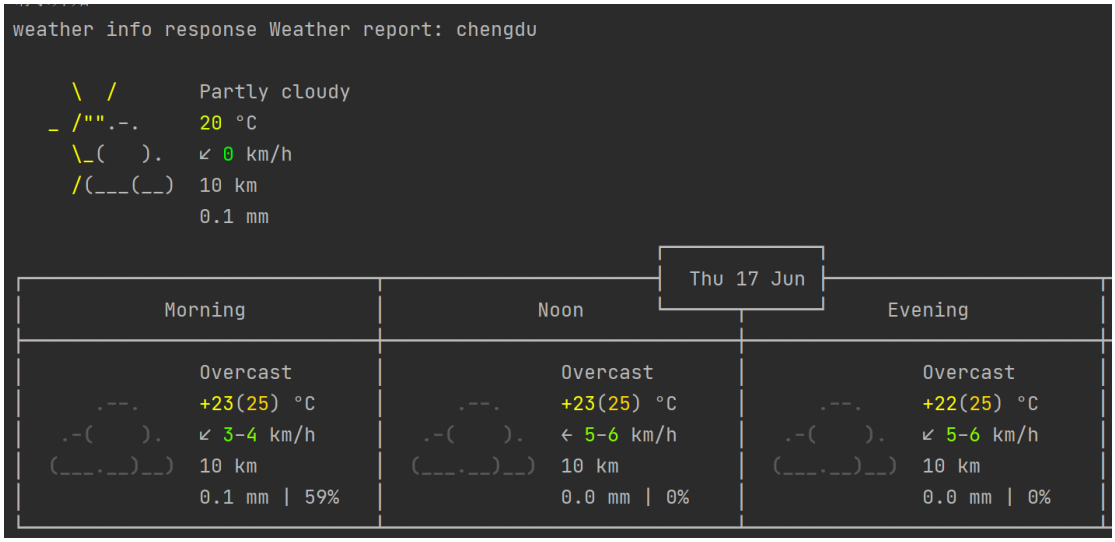
且查询天气功能与返回随机数功能正常，效果截图分别如下：

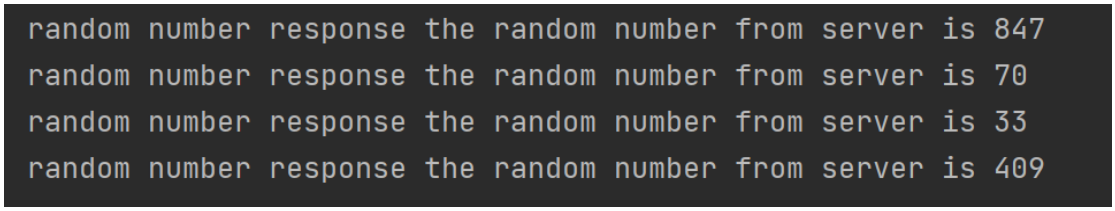
```
random number response the random number from server is 815
random number response the random number from server is 991
random number response the random number from server is 790
random number response the random number from server is 782
random number response the random number from server is 107
random number response the random number from server is 575
random number response the random number from server is 679
random number response the random number from server is 274
random number response the random number from server is 870
random number response the random number from server is 823
```

```
weather info response Weather report: beijing

              Mist
  _ - _ - _ -  +23(25) °C
   _ - _ - _   ↘ 15 km/h
  _ - _ - _ -  5 km
               0.1 mm

                                                       ┌─────────────┐
┌─────────────────────────────┬─────────────────────────┤  Thu 17 Jun ├─────────────────────────┐
│            Morning          │             Noon        └─────────────┘            Evening     │
├─────────────────────────────┼─────────────────────────────────┼─────────────────────────────┤
│     \   /       Partly cloudy │     \   /       Partly cloudy │     \   /       Sunny       │
│   _ /"".-.      +28(27) °C    │   _ /"".-.      +31(29) °C    │       .-.       +34(32) °C  │
│     \_(   ).    ↘ 13-15 km/h  │     \_(   ).    ↘ 15-17 km/h  │    ― (   ) ―    → 14-16 km/h │
│     /(___(__)   10 km         │     /(___(__)   10 km         │       `-'       10 km       │
│               0.0 mm | 0%     │               0.0 mm | 0%     │      /   \      0.0 mm | 0%  │
└─────────────────────────────┴─────────────────────────────────┴─────────────────────────────┘
```

2. 客户端

客户端发送请求以及展示服务端返回结果功能正常。效果如下图所示





发现的问题及解决方法

2.1 问题一

服务端接受客户端请求不完整。客户端发送给服务端的请求数据，服务端有时只能接受到一部分，有时却能全部接受。代码截图如下所示，猜测原因是服务端进程从 socket 中读取数据时，还没有等读取完客户端就断开了连接。

```
while True:
    # 从连接中读取长度为10240的数据
    temp = conn.recv(10240)
    # 系统暂停一秒，等待数据发送完全
    time.sleep(1)
    # 对收到的数据进行解码  二进制数据转化为字符串
    request_body = temp.decode("utf-8")
```

解决方法:

在请求头中加入数据的长度,在服务端一侧读取确定长度的数据,修改后代码如下所示:

```
while True:  # 循环读写
    length_prefix = conn.recv(4)  # 请求长度前缀

    if not length_prefix:  # 连接关闭了
        print(addr, "bye")
        conn.close()
        break  # 退出循环，处理下一个连接
    length, = struct.unpack("I", length_prefix)  # 将二进制解析为数字
    print("the length of request is " + str(length))
    body = conn.recv(length)  # 请求消息体
```

2.2 问题二

单线程服务端在处理一个请求时，无法对新来的请求进行响应并做出处理逻辑。为解决此问题，服务端程序更改为多线程。主线程始终保持在监听端口，当有一个请求进入需要执行处理逻辑时，不是在主线程上运行处理逻辑，而是创建一个新线程进行处理并返回结果，处理完成后此线程会被操作系统销毁，更改后代码如下图所示:

```
"""
i = 1
while True:
    print("循环开始%d次！" % i)
    conn, addr = sock.accept()  # 接收连接
    print(conn)
    thread.start_new_thread(handle_conn, (conn, addr, handlers))  # 开启新线程进行
    print("循环完成%d次！" % i)
    i += 1
```

2.3 问题三

天气获取功能起初使用 wthrcdn.etouch.cn 网站的天气信息，虽然可以正常获取数据但是数据格式并不是很友好，且解析过程比较繁琐，经过多方比较与查询后，最终确定使用 wttr.in 网站的天气预报，获取方式简单且返回结果清晰明了，代码以及效果截图如下所以：

```
def send_weather_info(conn, city_name):
    url = "http://wttr.in/{}".format(city_name)
    response_txt = requests.get(url).text
    send_result(conn, "weather info response", response_txt)
```