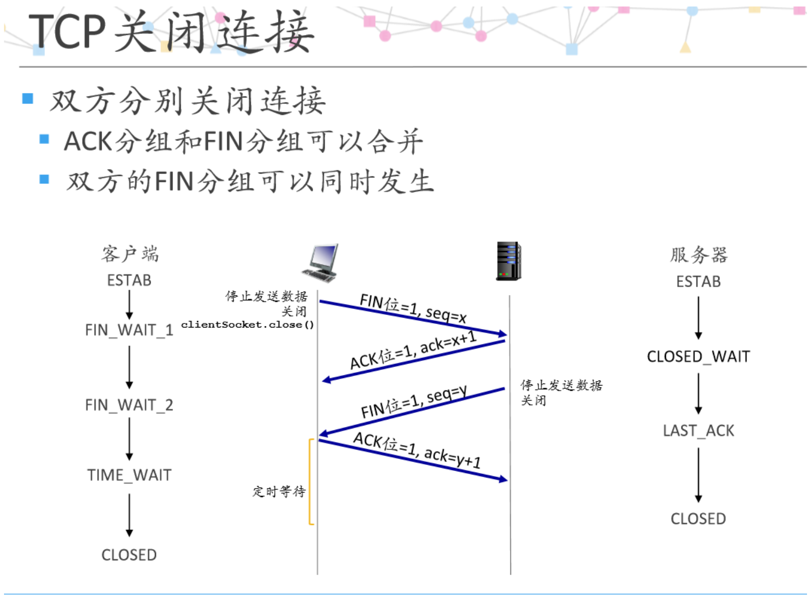
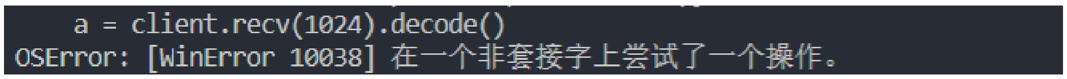
# 关于 TCP 关闭连接的答疑

谢瑞桃

同学提问：老师，这个地方如果客户端先关闭套接字，那服务器还是可以单方面发送数据的对吧？但是我在编程的时候，如果 clientsocket 调用了 close（），然后再用 recv（）接收数据的话，会出现一个 error。这样子 clientsocket 关闭之后，client 怎么接收数据嘞？见下图。


TCP关闭连接

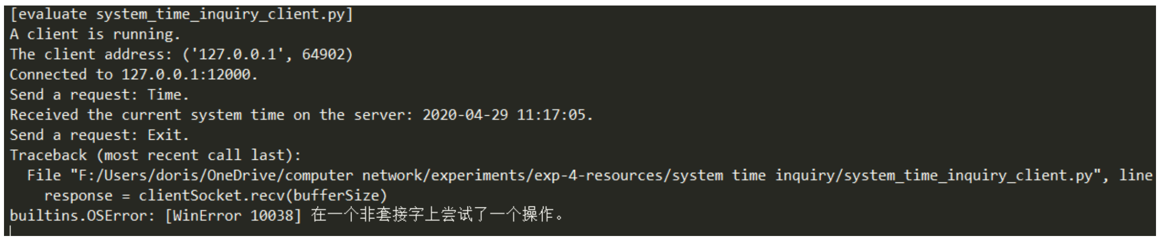
- 双方分别关闭连接
  - ACK分组和FIN分组可以合并
  - 双方的FIN分组可以同时发生


```
    a = client.recv(1024).decode()
OSError: [WinError 10038] 在一个非套接字上尝试了一个操作。
```

答疑：

0、这个问题非常好，感谢同学的提问。

1、我们用实验四里请求系统时间的例子做一下测试。我们在 clientSocket 发送完最后一个指令后，并且在 clientSocket 接收数据之前，调用 clientSocket.close()。客户端的运行结果如下，看到错误信息，的确会出现这个问题。


```
[evaluate system_time_inquiry_client.py]
A client is running.
The client address: ('127.0.0.1', 64902)
Connected to 127.0.0.1:12000.
Send a request: Time.
Received the current system time on the server: 2020-04-29 11:17:05.
Send a request: Exit.
Traceback (most recent call last):
  File "F:/Users/doris/OneDrive/computer network/experiments/exp-4-resources/system time inquiry/system_time_inquiry_client.py", line
    response = clientSocket.recv(bufferSize)
builtins.OSError: [WinError 10038] 在一个非套接字上尝试了一个操作。
```

我们找一下原因，查阅 socket 库的文档（https://docs.python.org/3.7/library/socket.html#socket.close），看看 close()函数的介绍。注意我用的是 python3.7，所以看对应版本的文档。如下：

socket.**close**()

Mark the socket closed. The underlying system resource (e.g. a file descriptor) is also closed when all file objects from `makefile()` are closed. Once that happens, all future operations on the socket object will fail. The remote end will receive no more data (after queued data is flushed).

Sockets are automatically closed when they are garbage-collected, but it is recommended to `close()` them explicitly, or to use a `with` statement around them.

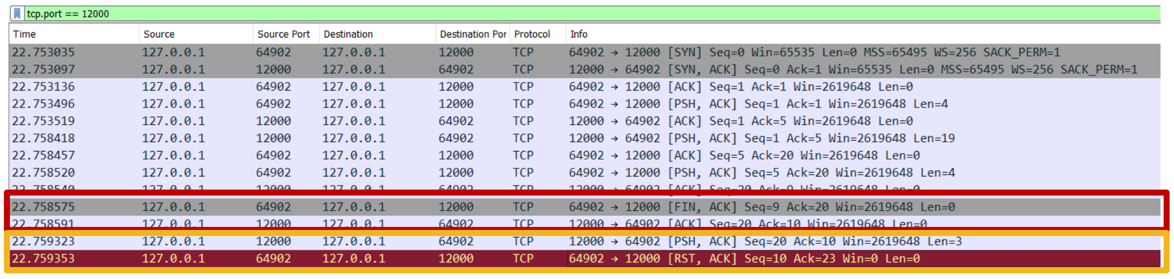
*Changed in version 3.6:* `OSError` is now raised if an error occurs when the underlying `close()` call is made.

**Note**

`close()` releases the resource associated with a connection but does not necessarily close the connection immediately. If you want to close the connection in a timely fashion, call `shutdown()` before `close()`.

文档清晰地讲了一旦调用该函数以后，该 socket 对象上的所有操作都将失效。因为，调用该函数将会释放其连接的所有资源，但不会立即关闭连接。并且，该文档建议如果想以实时地关闭连接，就使用 shutdown 函数。

2、我们分析一下上述过程中的数据包。



```
tcp.port == 12000
Time       Source     Source Port  Destination  Destination Por  Protocol  Info
22.753035  127.0.0.1  64902        127.0.0.1    12000            TCP       64902 → 12000 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
22.753097  127.0.0.1  12000        127.0.0.1    64902            TCP       12000 → 64902 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
22.753136  127.0.0.1  64902        127.0.0.1    12000            TCP       64902 → 12000 [ACK] Seq=1 Ack=1 Win=2619648 Len=0
22.753496  127.0.0.1  64902        127.0.0.1    12000            TCP       64902 → 12000 [PSH, ACK] Seq=1 Ack=1 Win=2619648 Len=4
22.753519  127.0.0.1  12000        127.0.0.1    64902            TCP       12000 → 64902 [ACK] Seq=1 Ack=5 Win=2619648 Len=0
22.758418  127.0.0.1  12000        127.0.0.1    64902            TCP       12000 → 64902 [PSH, ACK] Seq=1 Ack=5 Win=2619648 Len=19
22.758457  127.0.0.1  64902        127.0.0.1    12000            TCP       64902 → 12000 [ACK] Seq=5 Ack=20 Win=2619648 Len=0
22.758520  127.0.0.1  64902        127.0.0.1    12000            TCP       64902 → 12000 [PSH, ACK] Seq=5 Ack=20 Win=2619648 Len=4
22.758540  127.0.0.1  12000        127.0.0.1    64902            TCP       12000 → 64902 [ACK] Seq=20 Ack=9 Win=2619648 Len=0
22.758575  127.0.0.1  64902        127.0.0.1    12000            TCP       64902 → 12000 [FIN, ACK] Seq=9 Ack=20 Win=2619648 Len=0
22.758591  127.0.0.1  12000        127.0.0.1    64902            TCP       12000 → 64902 [ACK] Seq=20 Ack=10 Win=2619648 Len=0
22.759323  127.0.0.1  12000        127.0.0.1    64902            TCP       12000 → 64902 [PSH, ACK] Seq=20 Ack=10 Win=2619648 Len=3
22.759353  127.0.0.1  64902        127.0.0.1    12000            TCP       64902 → 12000 [RST, ACK] Seq=10 Ack=23 Win=0 Len=0
```

上图中，红色为客户端发给服务器端（端口号为 12000）的 FIN 报文以及与其对应的确认报文。黄色为服务器发给客户端的数据报文以及与其对应的确认。的确，如

socket 文档介绍，客户端调用 close 函数以后，发了 FIN 报文，连接还是半开的，tcp 还是接收到了数据。只是套接字异常了。

3、我们利用 socket 库中另一个函数 shutdown 可以实现连接半开放时的数据接收。

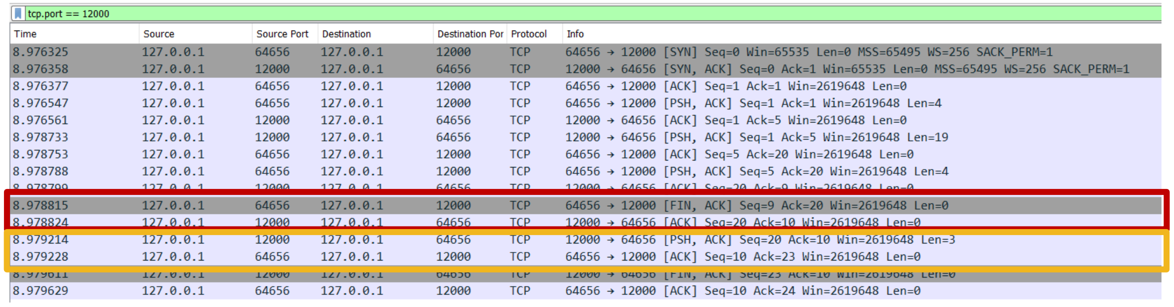
`socket.`**`shutdown`**`(`*how*`)`

Shut down one or both halves of the connection. If *how* is `SHUT_RD`, further receives are disallowed. If *how* is `SHUT_WR`, further sends are disallowed. If *how* is `SHUT_RDWR`, further sends and receives are disallowed.

举例：我们在 clientSocket 发送完最后一个指令后，并且在 clientSocket 接收数据之前，调用 clientSocket.shutdown(SHUT_WR)。clientSocket 会先发 FIN 分组，再收到数据。客户端运行结果如下图。

```
[evaluate system_time_inquiry_client.py]
A client is running.
The client address: ('127.0.0.1', 64656)
Connected to 127.0.0.1:12000.
Send a request: Time.
Received the current system time on the server: 2020-04-29 10:57:41.
Send a request: Exit.
Received a response: Bye.
```

我们再看看这次的数据包。



tcp.port == 12000

| Time | Source | Source Port | Destination | Destination Por | Protocol | Info |
|---|---|---|---|---|---|---|
| 8.976325 | 127.0.0.1 | 64656 | 127.0.0.1 | 12000 | TCP | 64656 → 12000 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1 |
| 8.976358 | 127.0.0.1 | 12000 | 127.0.0.1 | 64656 | TCP | 12000 → 64656 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1 |
| 8.976377 | 127.0.0.1 | 64656 | 127.0.0.1 | 12000 | TCP | 64656 → 12000 [ACK] Seq=1 Ack=1 Win=2619648 Len=0 |
| 8.976547 | 127.0.0.1 | 64656 | 127.0.0.1 | 12000 | TCP | 64656 → 12000 [PSH, ACK] Seq=1 Ack=1 Win=2619648 Len=4 |
| 8.976561 | 127.0.0.1 | 12000 | 127.0.0.1 | 64656 | TCP | 12000 → 64656 [ACK] Seq=1 Ack=5 Win=2619648 Len=0 |
| 8.978733 | 127.0.0.1 | 12000 | 127.0.0.1 | 64656 | TCP | 12000 → 64656 [PSH, ACK] Seq=1 Ack=5 Win=2619648 Len=19 |
| 8.978753 | 127.0.0.1 | 64656 | 127.0.0.1 | 12000 | TCP | 64656 → 12000 [ACK] Seq=5 Ack=20 Win=2619648 Len=0 |
| 8.978788 | 127.0.0.1 | 64656 | 127.0.0.1 | 12000 | TCP | 64656 → 12000 [PSH, ACK] Seq=5 Ack=20 Win=2619648 Len=4 |
| 8.978799 | 127.0.0.1 | 12000 | 127.0.0.1 | 64656 | TCP | 12000 → 64656 [ACK] Seq=20 Ack=9 Win=2619648 Len=0 |
| 8.978815 | 127.0.0.1 | 64656 | 127.0.0.1 | 12000 | TCP | 64656 → 12000 [FIN, ACK] Seq=9 Ack=20 Win=2619648 Len=0 |
| 8.978824 | 127.0.0.1 | 12000 | 127.0.0.1 | 64656 | TCP | 12000 → 64656 [ACK] Seq=20 Ack=10 Win=2619648 Len=0 |
| 8.979214 | 127.0.0.1 | 12000 | 127.0.0.1 | 64656 | TCP | 12000 → 64656 [PSH, ACK] Seq=20 Ack=10 Win=2619648 Len=3 |
| 8.979228 | 127.0.0.1 | 64656 | 127.0.0.1 | 12000 | TCP | 64656 → 12000 [ACK] Seq=10 Ack=23 Win=2619648 Len=0 |
| 8.979611 | 127.0.0.1 | 12000 | 127.0.0.1 | 64656 | TCP | 12000 → 64656 [FIN, ACK] Seq=23 Ack=10 Win=2619648 Len=0 |
| 8.979629 | 127.0.0.1 | 64656 | 127.0.0.1 | 12000 | TCP | 64656 → 12000 [ACK] Seq=10 Ack=24 Win=2619648 Len=0 |

上图中，红色为客户端发给服务器端的 FIN 报文以及与其对应的确认报文。黄色为服务器发给客户端的数据报文以及与其对应的确认。之后，服务器关闭连接。

4、授之以鱼，不如授之以渔。课堂上讲解的内容只是我们这个领域非常少的一部分基础内容，同学们以后在实践中会遇到非常多的问题，怎么办呢？希望同学们能学习遇到问题以后解决问题的方法。