聊聊优雅关闭连接 | Socket 选项之 SO_LINGER

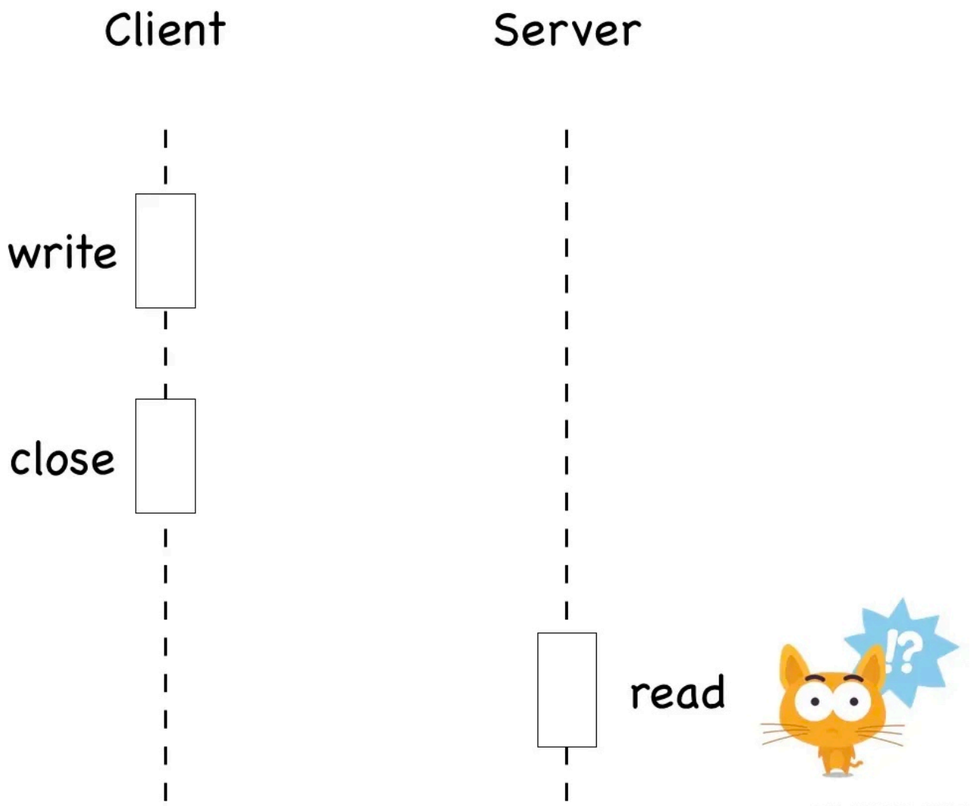
看下面的代码：

```
Socket socket = new Socket();
InetSocketAddress serverSocketAddress = new InetSocketAddress("10.0.0.3", 8080);
socket.connect(serverSocketAddress);

byte[] msg = getMessageBytes();
socket.getOutputStream().write(msg);

socket.close();
```

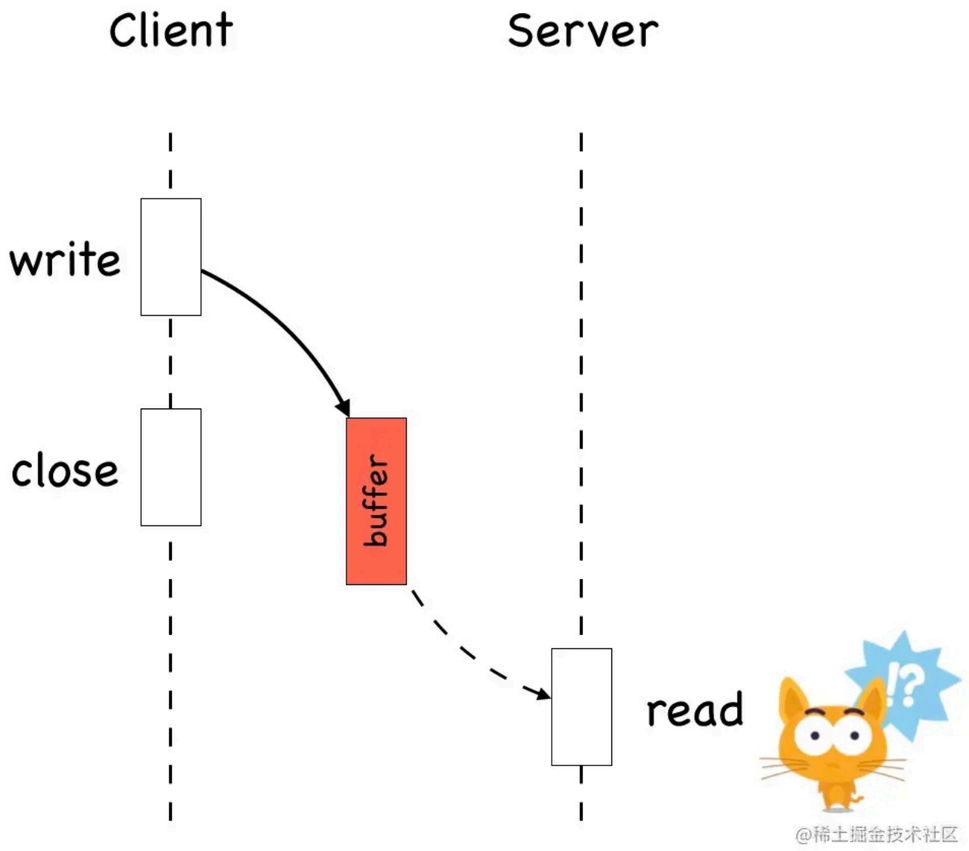
会发生什么：

1. 服务器收到 msg 所有内容
2. 服务器会收到 msg 部分内容
3. 服务器会抛出异常

图示



往套接字写东西，实际上是往内核写东西



关闭套接字的常见方式

- FIN：优雅关闭，发送 FIN 包表示自己这端所有的数据都已经发送出去了，后面不会再发送数据
- RST：强制连接重置关闭，无法做出什么保证

当调用 socket.close() 的时候会发生什么呢?

正常情况下

- 操作系统等所有的数据发送完才会关闭连接
- 因为是主动关闭，所以连接将处于 TIME_WAIT 两个 MSL

SO_LINGER

linger ☆ ⋯

英 ['lɪŋgə(r)] 美 ['lɪŋgər]

v. 继续存留，缓慢消失；流连，逗留；持续看（或思考）；苟延残喘；消磨，缓慢度过

【名】（Linger）（法）兰热，（德、捷、瑞典）林格（人名）

第三人称单数 lingers

现在分词 lingering

过去式 lingered

过去分词 lingered

SO_LINGER 参数是一个 linger 结构体，代码如下

```
struct linger {
    int l_onoff;    /* linger active */
    int l_linger;   /* how many seconds to linger for */
};
```

- l_onoff 用来表示是否启用 linger 特性，非 0 为启用，0 为禁用 ， linux 内核默认为禁用。这种情况下 close 函数立即返回，操作系统负责把缓冲队列中的数据全部发送至对端
- l_linger 在 l_onoff 为非 0 （即启用特性）时才会生效。
  - 如果 l_linger 的值为 0，那么调用 close，close 函数会立即返回，同时丢弃缓冲区内所有数据并立即发送 RST 包重置连接
  - 如果 l_linger 的值为非 0，那么此时 close 函数在阻塞直到 l_linger 时间超时或者数据发送完毕，发送队列在超时时间段内继续尝试发送，如果发送完成则皆大欢喜，超时则直接丢弃缓冲区内容 并 RST 掉连接。

实验环节

案例分析：从 Nginx Connection reset by peer 看网络分析

(104: Connection reset by peer) while sending request to upstream