链滴

Java 困扰三周の问题：使用 byte[] 或 skip() 方法读取字节流 Stream 文件尾部多 / 少 / 缺字节解决方法

作者：adlered

原文链接：https://ld246.com/article/1568775939612

来源网站：链滴

前言

最近在造一个最强兼容性的FTP服务端轮子，但在使用InputStream、OutputStream及它的子类时我遇到了很奇怪也很严重的问题：**数据尾部随机缺少/多出数据。**这个问题简直太致命了。

问题复现

先出一段代码，我之前操作数据流的步骤如下（请注意看注释）：

```
// 获取文件输出流FileOutputStream和网络输入流InputStream
FileOutputStream fileOutputStream = new FileOutputStream(file);
InputStream inputStream = socket.getInputStream();
// 新建一个字节数组
byte[] bytes = new byte[8192];
// 从网络输入流inputStream，读取字节并写入到字节数组byte[] bytes
while ((inputStream.read(bytes)) != -1) {
    // 写入到本地文件中
    fileOutputStream.write(bytes);
}
// 以防万一，刷新一下缓存
fileOutputStream.flush();
// 关闭流
inputStream.close();
fileOutputStream.close();
```

无论使用何种字节流类，我使用new byte[8192]这种字节数组读取数据时，最终的结果总会多出来者少出来很多字节，非常奇怪，但在逻辑上，我以防万一，还将fileOutputStream中的缓存刷新了出，按理来讲应该没有问题了。

这个问题在我尝试了BufferedXxxStream、FileXxxStream、XxxStream了以后还是没有得到解决，无数次的尝试之后，浪费了我三个星期的时间去解决。但在之后的一次偶然write()方法的查阅中，我到了解决问题的线索。

问题解决过程

首先，我查阅了InputStream的read方法：

方法	用法
read()	从输入流中读取数据的下一个字节，返回到255范围内的int字节值。如果因为已经到达流末尾而没有可用的字节，则返回-1。在输入数据可用检测到流末尾或者抛出异常前，此方法一直阻塞。
read(byte[] b)	从输入流中读取一定数量的字节并将其存储在缓冲区数组 b 中。以整数形式返回实际读取的字节数。

解决思路

注：下面的解题思路中，read方法我只调用了一次，**忽略**了while，方便阅读理解。

1. 我使用read()方法逐个读取字节写入文件是 **没有问题**的，但是这样逐个读取**速度太慢、CPU爆满**：

```
int data;
data = inputStream.read();
// 此时data中应存储了输入流中的一个字节
```

2. 使用read(byte[] b)的方法，　**就**出现了**字节错乱**的问题，方法也是我一开始认为正确的：

```
byte[] bytes = new byte[8192];
inputStream.read(bytes);
// 此时bytes中应存储了输入流中的前8192字节（这里是我之前的误解，下面有解释）
```

3. 然后突然想到了一个问题：如果网络不好的话，　**输入流真的每次都能存满8192字节吗?**

4. 果然老师说的没错，不把题读完整，坑的一定是自己。看read(byte[] b)方法解释的后半段，　**以数形式返回实际读取的字节数**，也就是说，如果我用**int**接住它的返回值，就能得到本次read方法读到的**真实长度**？试试看：

```
byte[] bytes = new byte[8192];
int len = -1;
len = inputStream.read(bytes);
// 此时bytes中应存储了相应大小的字节，而len中存储的应是读取到的真实字节长度。
// 注意，len中存储的是长度，而不是read()不传参方法中的字节。
// 由于我省略掉了while，实际上下面的方法应该打印了很多次。
System.out.println("Length: " + len);
```

运行以后，我获得到的部分结果：

```
Length: 8192
Length: 8192
Length: 7000
Length: 8192
Length: 8092
```

果不其然啊，在第三次read方法时，**实际上只有7000个字节被写入了**，还剩下**1192个字节直接被填了0**，在多出了一堆数据的同时，文件数据也变得**不连贯**，直接导致损坏。

解决方案

问题找到了，其它的就简单了，只要我们不把它额外填充的字节写入就可以了。查阅一下OutputStream.write方法的使用：

```
write(byte[] b, int off, int len)
从指定的字节数组写入len个字节，从偏移off开始输出到此输出流。
```

Wow, AMAZING.

这就简单了，在已知长度n的条件下，我们只需要使用write(bytes, 0, n)就可以了：

```
byte[] bytes = new byte[8192];
int len = -1;
len = inputStream.read(bytes);
outputStream.write(bytes, 0, len);
```

从第0个字节开始，写入到len中指定的下标，OK。

至于为什么从0开始？别闹。

后语

在出现问题时，一定要**仔细翻阅官方文档**！这是次教训，也是个经验。