[**被readLine()折腾了一把**](http://blog.csdn.net/swingline/article/details/5357581)

标签： [string](http://www.csdn.net/tag/string)[null](http://www.csdn.net/tag/null)[buffer](http://www.csdn.net/tag/buffer)[dst](http://www.csdn.net/tag/dst)[input](http://www.csdn.net/tag/input)[user](http://www.csdn.net/tag/user)

2010-03-08 19:26 25867人阅读 [评论](http://blog.csdn.net/swingline/article/details/5357581#comments)(4) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/swingline/article/details/5357581#report)

版权声明：本文为博主原创文章，未经博主允许不得转载。

虽然写IO方面的程序不多，但BufferedReader/BufferedInputStream倒是用过好几次的，原因是：

* 它有一个很特别的方法：readLine()，使用起来特别方便，每次读回来的都是一行，省了很多手动拼接buffer的琐碎；
* 它比较高效，相对于一个字符/字节地读取、转换、返回来说，它有一个缓冲区，读满缓冲区才返回；一般情况下，都建议使用它们把其它Reader/InputStream包起来，使得读取数据更高效。
* 对于文件来说，经常遇到一行一行的，特别相符情景。

这次是在蓝牙开发时，使用两个蓝牙互相传数据(即一个发一个收)，bluecove这个开源组件已经把数据读取都封装成InputStream了，也就相当于平时的IO读取了，很自然就使用起readLine()来了。

发数据：

**[java]** [view plain](http://blog.csdn.net/swingline/article/details/5357581) [copy](http://blog.csdn.net/swingline/article/details/5357581) [print](http://blog.csdn.net/swingline/article/details/5357581)[?](http://blog.csdn.net/swingline/article/details/5357581)

1. BufferedWriter output = new BufferedWriter(new OutputStreamWriter(conn.openOutputStream()));
2. int i = 1;
3. String message = "message " + i;
4. while(isRunning) {
5. output.write(message+"/n");
6. i++;
7. }



读数据：

**[java]** [view plain](http://blog.csdn.net/swingline/article/details/5357581) [copy](http://blog.csdn.net/swingline/article/details/5357581) [print](http://blog.csdn.net/swingline/article/details/5357581)[?](http://blog.csdn.net/swingline/article/details/5357581)

1. BufferedReader input = new BufferedReader(new  InputStreamReader(m\_conn.openInputStream()));
2. String message = "";
3. String line = null;
4. while((line = m\_input.readLine()) != null) {
5. message += line;
6. }
7. System.out.println(message);



上面是代码的节选，使用这段代码会发现写数据时每次都成功，而读数据侧却一直没有数据输出(除非把流关掉)。经过折腾，原来这里面有几个大问题需要理解：

* 误以为readLine()是读取到没有数据时就返回null(因为其它read方法当读到没有数据时返回-1)，而实际上readLine()是一个阻塞函数，当没有数据读取时，就一直会阻塞在那，而不是返回null；因为readLine()阻塞后，System.out.println(message)这句根本就不会执行到，所以在接收端就不会有东西输出。要想执行到System.out.println(message)，一个办法是发送完数据后就关掉流，这样readLine()结束阻塞状态，而能够得到正确的结果，但显然不能传一行就关一次数据流；另外一个办法是把System.out.println(message)放到while循环体内就可以。
* readLine()只有在数据流发生异常或者另一端被close()掉时，才会返回null值。
* 如果不指定buffer大小，则readLine()使用的buffer有8192个字符。在达到buffer大小之前，只有遇到"/r"、"/n"、"/r/n"才会返回。

readLine()的实质(下面是从JDK源码摘出来的)：

**[java]** [view plain](http://blog.csdn.net/swingline/article/details/5357581) [copy](http://blog.csdn.net/swingline/article/details/5357581) [print](http://blog.csdn.net/swingline/article/details/5357581)[?](http://blog.csdn.net/swingline/article/details/5357581)

1. String readLine(boolean ignoreLF) throws IOException {
2. StringBuffer s = null;
3. int startChar;
4. synchronized (lock) {
5. ensureOpen();
6. boolean omitLF = ignoreLF || skipLF;
7. bufferLoop:
8. for (;;) {
9. if (nextChar >= nChars)
10. fill(); //在此读数据
11. if (nextChar >= nChars) { /\* EOF \*/
12. if (s != null && s.length() > 0)
13. return s.toString();
14. else
15. return null;
16. }
17. ......//其它
18. }
20. private void fill() throws IOException {
21. ..../其它
22. int n;
23. do {
24. n = in.read(cb, dst, cb.length - dst); //实质
25. } while (n == 0);
26. if (n > 0) {
27. nChars = dst + n;
28. nextChar = dst;
29. }
30. }



从上面看出，readLine()是调用了read(char[] cbuf, int off, int len) 来读取数据，后面再根据"/r"或"/n"来进行数据处理。

在Java I/O书上也说了：

public String readLine() throws IOException  
This method returns a string that contains a line of text from a text file. /r, /n, and /r/n are assumed to be line breaks and are not included in the returned string. This method is often used when reading user input from System.in, since most platforms only send the user's input to the running program after the user has typed a full line (that is, hit the Return key).  
readLine() has the same problem with line ends that DataInputStream's readLine() method has; that is, **the potential to hang on a lone carriage return that ends the stream** . This problem is especially acute on networked connections, where readLine() should never be used.

小结，使用readLine()一定要注意：

1. 读入的数据要注意有/r或/n或/r/n
2. 没有数据时会阻塞，在数据流异常或断开时才会返回null
3. 使用socket之类的数据流时，要避免使用readLine()，以免为了等待一个换行/回车符而一直阻塞