标准 I/O

通过 FileI/O 的学习我们知道,当程序读写文件的时候,在程序执行完毕以后,都会及时的关闭输入流或者输出流。这些输入流或者是输出流。它们的生命周期是短暂的。不会存在程于序运行的整个生命周期中。然而对于某些应用程序，可能需要在程序运行的整个生命周期中，都是从同一个文件源中读取或者是向同一个数据源来汇集输出数据，也就是源数据源和目标数据源是始终不变的。在 JDK 的 java.lang.System 当中，提供了三个静态变量：

System.in、System.out 和 System.err。

- System.in，它为 InputStream 类型，它代表着标准输入流，默认的数据源为键盘，程序可以通过 System.in 读取标准输入流的数据。
- System.out 它为 PrintStream 类型,它代表的是标准输出流,默认的数据是控制台，程序可以通过 System.out 输出运行时的正常消息。System.err 为 PrintStream 类型，它代表标准错误输出流。

这以上三种流都是 Java 虚拟机创建的，它们存在于整个运行生命周期中，这些流始终处于打开状态，除非我们利用程序显式的来关闭它们，只要程序没有关闭这些流，在程序运行的任何时候都可以通过它们来输入或者是输出数据。这些就是标准输入输出的含义。

默认情况下，System.in、System.out 分别代表键盘和控制台或者说代表键盘和显示器，当程序通过 System.in 来获取输入时，实际上是从键盘读取输入，当程序试图通过 System.out 执行输出时，程序总是输出到显示器。有些时候就需要我们对这样的情况进行改变,也就是说重新将输入输出的默认改到文件或其他的位置,而不再是键盘和屏幕显示器。这就涉及到重定向标准输入输出。在 System 类里提供三个重定向标准输入和输出方法，如表-1 所示。

表-1 重定向标准输入和输出方法的常用方法

方法	说明
static void setErr(PrintStream err)	重定向标准错误输出流
static void setIn(InputStream in)	重定向标准输入流
static void setOut(PrintStream out)	重定向标准输出流

➢ 将输出重定向到文件输出，代码如示例 1 所示。

示例 1：

```
... ...
//创建 PrintStream 输出流
PrintStream ps =new PrintStream(new FileOutputStream("c:\\myDoc\\hello.txt"));
//重定向到文件
System.setOut(ps);
//向文件 print 内容
System.out.print("我的测试，重定向到文件 hello！");
System.out.println(new ReOut());
... ...
```

示例 1 代码中，创建输出流对象 ps，将这个 ps 对象指向 c 盘的 myDoc 下的 hello.txt 这个文件，然后调用 System.setOut 方法将标准输出重定向到刚才的文件 System.setOut 参数为输出流对象 ps，这样就将标准输出重定向到文件。System.out.print("我的测试，重定向到文件 hello！")是测试打印输出一行代码。打开 hello.txt 文本文件，内容如图 1 所示。

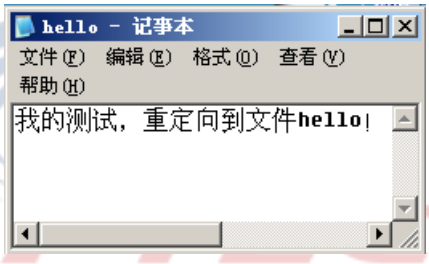

图 1 输出重定向到文件输出

通过效果图说明标准输出已经不再是控制台或者是屏幕了，而是输出到了 hello 文件。

➢ 重定向到标准输入的操作，代码如示例 1 所示。

示例 2：

```
... ...
//创建输入流
FileInputStream fis=new FileInputStream("c:\\myDoc\\hello.txt");
//重定向到文件
System.setIn(fis);
Scanner sc=new Scanner(System.in);
System.out.print("从文件读取到的内容为："+sc.next());
... ...
```

重定向标准输入，将标准输入由键盘改为文件 hello。

首先创建输入流 fis，输入文件对象指向 hello.txt 这个文本文件。然后调用 System.setIn 方法将标准输入重定向到文件。