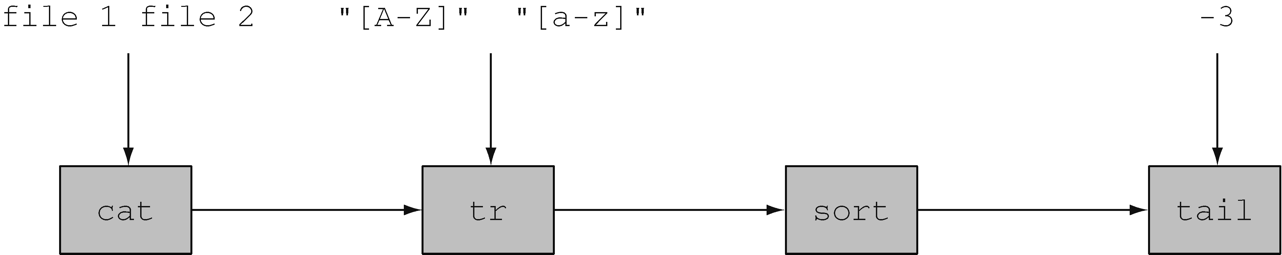
第一个编程概念是流处理。流是一系列数据项，一次只生成一项。程序可以从输入流中一个一个读取数据项，然后以同样的方式将数据项写入输出流。一个程序的输出流很可能是另一个程序的输入流。

一个实际的例子是在Unix或Linux中，很多程序都从标准输入（Unix和C中的stdin，Java中的System.in）读取数据，然后把结果写入标准输出（Unix和C中的stdout，Java中的System.out）。首先，Unix的cat命令会把两个文件连接起来创建一个流，tr命令会转换流中的字符，sort命令会对流中的行进行排序，而tail -3给出流的最后三行。Unix命令行允许这些程序通过管道（|）连接在一起，比如

cat file1 file2 | tr “[A-Z]” “[a-z]” | sort | tail -3

会先把字母转换成小写字母，然后打印出按照词典排序出现在最后的三个单词。sort把一个行流作为输入，产生了另一个行流(进行排 序)作为输出，如图所示。请注意在Unix中，命令(cat、tr、sort和tail)是同时执行的，这样sort就可以在cat或tr完成前先处理头几行。就像汽车组装流水线一样，汽车排队进入加工站，每个加工站会接收、修改汽车，然后将之传递给下一站做进一步的处理。尽管流水线实际上是一个序列，但不同加工站的运行一般是并行的。



基于这一思想，Java 8在java.util.stream中添加了一个Stream API；Stream<T>就是一系列T类型的项目。你现在可以把它看成一种比较花哨的迭代器。Stream API的很多方法可以链接起来形成一个复杂的流水线。

推动这种做法的关键在于，现在你可以在一个更高的抽象层次上写Java 8程序了：思路变成了把这样的流变成那样的流（就像写数据库查询语句时的那种思路），而不是一次只处理一个项目。另一个好处是，Java 8可以透明地把输入的不相关部分拿到几个CPU内核上去分别执行你的Stream操作流水线——这是几乎免费的并行，用不着去费劲搞Thread了。