<https://www.cnblogs.com/sos-blue/p/6798810.html>

1、首先，bash中0，1，2三个数字分别代表STDIN\_FILENO、STDOUT\_FILENO、STDERR\_FILENO，即标准输入（一般是键盘），标准输出（一般是显示屏，准确的说是用户终端控制台），标准错误（出错信息输出）。

2、输入输出可以重定向，所谓重定向输入就是在命令中指定具体的输入来源，譬如 cat < test.c 将test.c重定向为cat命令的输入源。输出重定向是指定具体的输出目标以替换默认的标准输出，譬如ls > 1.txt将ls的结果从标准输出重定向为1.txt文本。有时候会看到如 ls >> 1.txt这类的写法，> 和 >> 的区别在于：> 用于新建而>>用于追加。即ls > 1.txt会新建一个1.txt文件并且将ls的内容输出到新建的1.txt中，而ls >> 1.txt则用在1.txt已经存在，而我们只是想将ls的内容追加到1.txt文本中的时候。

3、默认输入只有一个（0，STDIN\_FILENO），而默认输出有两个（标准输出1 STDOUT\_FILENO，标准错误2 STDERR\_FILENO）。因此默认情况下，shell输出的错误信息会被输出到2，而普通输出信息会输出到1。但是某些情况下，我们希望在一个终端下看到所有的信息（包括标准输出信息和错误信息），要怎么办呢？

       对了，你可以使用我们上面讲到的输出重定向。思路有了，怎么写呢？ 非常直观的想法就是2>1（将2重定向到1嘛），行不行呢？试一试就知道了。我们进行以下[**测试**](http://lib.csdn.net/base/softwaretest)步骤：

1）mkdir test && cd test                ; 创建test文件夹并进入test目录

2）touch a.txt b.c c                          ; 创建a.txt b.c c 三个文件

3）ls > 1                                           ; 按我们的猜测，这句应该是将ls的结果重定向到标准输出，因此效果和直接ls应该一样。但是实际这句执行后，标准输出中并没有任何信息。

4）ls                                                  ; 执行3之后再次ls，则会看到test文件夹中多了一个文件1

5）cat 1                                            ; 查看文件1的内容，实际结果为：1 a.txt b.c c     可见步骤3中 ls > 1并不是将ls的结果重定向为标准输出，而是将结果重定向到了一个文件1中。即1在此处不被解释为STDOUT\_FILENO，而是文件1。

4、到了此时，你应该也能猜到2>&1的用意了。不错，2>&1就是用来将标准错误2重定向到标准输出1中的。此处1前面的&就是为了让bash将1解释成标准输出而不是文件1。至于最后一个&，则是让bash在后台执行。