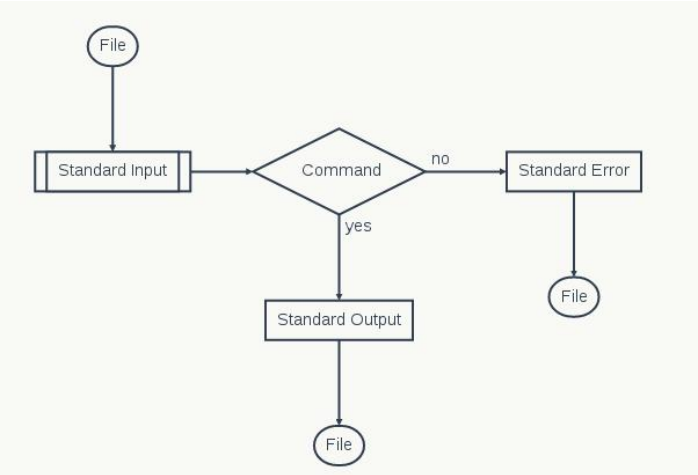
重定向简介

计算机最基础的功能是可以提供输入输出操作。对于Linux系统来说，通常以键盘为默认输入设备，又称标准输入设备；以显示器为默认的输出设备，又称标准输出设备。所谓重定向，就是将原本应该从标准输入设备（键盘）输入的数据，改由其他文件或设备输入，或将原本应该输出到标准输出设备（显示器）的内容，改由输出到其他文件或设备上。

文件标识符是重定向中很重要的一个概念，Linux使用0到9的整数指明了与特定进程相关的数据流，系统在启动一个进程的同时会为该进程打开三个文件：标准输入（stdin）、标准输出（stdout）、标准错误输出（stderr），分别用文件标识符0、1、2来标识。如果要为进程打开其他的输入输出，则需要从整数3开始标识。默认情况下，标准输入为键盘，标准输出和错误输出为显示器。

```
[root@sunday-test ~]# ll /dev/stdin
lrwxrwxrwx. 1 root root 15 Nov  6 08:50 /dev/stdin -> /proc/self/fd/0
[root@sunday-test ~]# ll /dev/stdout
lrwxrwxrwx. 1 root root 15 Nov  6 08:50 /dev/stdout -> /proc/self/fd/1
[root@sunday-test ~]# ll /dev/stderr
lrwxrwxrwx. 1 root root 15 Nov  6 08:50 /dev/stderr -> /proc/self/fd/2
```

对于一条命令的执行过程如下：



I/O重定向符号和用法

简单的说，I/O重定向可以将任何文件、命令、脚本、程序的输出重定向到另外一个文件、命令、程序或脚本。常见的重定向符号和功能如下图：

输出重定向：

Command > filename	把标准输出重定向到一个新文件中，当filename不存在时
Command >> filename	把标准输出重定向到一个文件中（追加）
Command > filename	把标准输出重定向到一个文件中
Command > filename 2>&1	把标准输出和错误一起重定向到一个文件中
Command 2 > filename	把标准错误重定向到一个文件中
Command 2 >> filename	把标准输出重定向到一个文件中（追加）
Command >> filename2>&1	把标准输出和错误一起重定向到一个文件（追加）

输入重定向

Command < filename > filename2	Command命令以filename文件作为标准输入，以filename2文件作为标准输出
Command < filename	Command命令以filename文件作为标准输入
Command << delimiter	从标准输入中读入，知道遇到delimiter分界符

绑定重定向

Command >&m	把标准输出重定向到文件描述符m中
Command < &-	关闭标准输入
Command 0>&-	同上

实例：

#显示当前目录文件 test.sh test1.sh test1.sh实际不存在

[root@sunday-test ~]# touch test.sh

[root@sunday-test ~]# ls test.sh test1.sh

ls: cannot access test1.sh: No such file or directory

test.sh

#正确输出与错误输出都显示在屏幕了，现在需要把正确输出写入suc.txt

1>可以省略，不写，默认所至标准输出

[root@sunday-test ~]# ls test.sh test1.sh > suc.txt

ls: cannot access test1.sh: No such file or directory

[root@sunday-test ~]# cat suc.txt

test.sh

#把错误输出，不输出到屏幕，输出到err.txt

[root@sunday-test ~]# ls test.sh test1.sh > suc.txt 2> err.txt

[root@sunday-test ~]# cat err.txt

ls: cannot access test1.sh: No such file or directory

```
[root@sunday-test ~]#
#继续追加把输出写入suc.txt err.txt  “>>”追加操作符
[root@sunday-test ~]# ls test.sh test1.sh >> suc.txt 2>> err.txt

#将错误输出信息关闭掉
[root@sunday-test ~]# ls test.sh test1.sh 2>&-
test.sh
[root@sunday-test ~]# ls test.sh test1.sh 2>/dev/null
test.sh
#&[n] 代表是已经存在的文件描述符，&1 代表输出 &2代表错误输出 &-代表关闭与它绑定的描述符
#/dev/null 这个设备，是linux 中黑洞设备，什么信息只要输出给这个设备，都会给吃掉

#关闭所有输出
[root@sunday-test ~]# ls test.sh test1.sh 1>&- 2>&-
#关闭 1 ，2 文件描述符
[root@sunday-test ~]# ls test.sh test1.sh 1>/dev/null 2>/dev/null
#将1,2 输出转发给/dev/null设备
[root@sunday-test ~]# ls test.sh test1.sh 1>/dev/null 2>&1
#将错误输出2 绑定给 正确输出 1，然后将 正确输出 发送给 /dev/null设备  这种常用
[root@sunday-test ~]# ls test.sh test1.sh &> /dev/null
#& 代表标准输出 ，错误输出 将所有标准输出与错误输出 输入到/dev/null文件

#使用标准输入，在a.txt文件中写入 "hello world"
[root@sunday-test ~]# cat > a.txt << EOF
> hello world
> EOF
[root@sunday-test ~]# cat a.txt
hello world
#在shell编程中，” EOF“通常与”<<“结合使用，“<<EOF“表示后续的输入作为子命令或子shell的输入，直到遇到”EOF“
```

管道符“|”

管道表现为输入输出重定向的一种方法，可以将一个命令的输出内容当做下一个命令的输入内容，可以通过多个简单命令的共同协作来完成较为复杂的工作。