

24 输入重定向和管道，随意流转

更新时间：2019-07-16 14:44:00



“

理想必须要人们去实现它，它不但需要决心和勇敢而且需要知识。

——吴玉章

”

内容简介

1. 前言
2. <, <<: 从文件或键盘读取
3. | : 管道
4. 总结

1. 前言

上一课流和输出重定向，心之所向，我们开始了“流，管道，重定向”的学习。

这一课我们学习新的内容：输入重定向和管道。

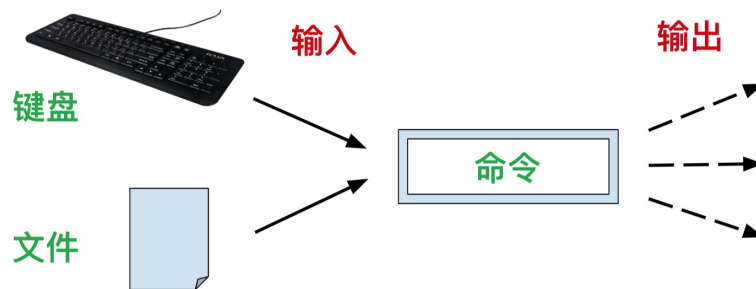
2. <, <<: 从文件或键盘读取

到目前为止，我们只讲了如何重定向命令的输出，也就是决定命令输出的信息的去向。那么接着我们可以做一点相反的事情：决定命令的输入来自哪里。

当然了，不是所有的命令都有输入，也不是所有的命令都有输出。

到目前为止，我们的命令的输入都来自于后面接的参数，这些参数有些是文件名，有些是目录名，等等。

但我们其实可以使命令的输入来自于文件或者键盘输入。如下图所示：



<: 从文件中读取

看到这个 < 符号，是否想到了之前的 > 符号呢？

是的，这对“孪生兄弟”，原理类似但是功能正相反，< 符号用于指定命令的输入。

我们用一个简单的例子来演示：

```
cat < notes.csv
```

```
oscar@oscar-laptop: ~/redirect$ cat < notes.csv
Mark,95 / 100,很不错
Matthew,30 / 100,跟平时一样水
Maria,70 / 100,有进步
Luke,54 / 100,接近平均分了
John,68 / 100,不错，但还可以更好
Samuel,100 / 100,总是那么完美
David,40 / 100,退步挺大呀
oscar@oscar-laptop:~/redirect$
```

如上图所示，`cat < notes.csv` 的运行结果和 `cat notes.csv`（不用重定向流符号）一模一样，都是在终端打印 `notes.csv` 的内容，那我们为什么需要 < 符号呢？

事实上，虽然 `cat < notes.csv` 的运行结果和 `cat notes.csv` 一样，但是原理却不一样：

- `cat notes.csv`：这种情况下，`cat` 命令接受的输入是 `notes.csv` 这个文件名，那么它要先打开 `notes.csv` 文件，然后打印出文件内容。
- `cat < notes.csv`：这种情况下，`cat` 命令接受的输入直接是 `notes.csv` 这个文件的内容，`cat` 命令只负责将其内容打印。而打开文件并将文件内容传递给 `cat` 命令的工作则交给 Shell 程序（也就是控制终端的程序）来完成。

所以，虽然结果看似一样，但是中间的过程却是不一样的。

<<: 从键盘读取

<< 符号的作用是将键盘的输入重定向为某个命令的输入，很多情况下都很有用。

我们用实例来说明：

```
sort -n << END
```

```
oscar@oscar-laptop: ~/r
File Edit View Search Terminal Help
oscar@oscar-laptop:~/redirect$ sort -n << END
> 
```

如上图所示，输入这条命令之后，按下回车，终端就进入了键盘输入模式。

看到那个 > 符号和其后闪动的光标了么？就是让你输入数据的。

我们知道 `sort -n` 的作用是将数值按照从小到大进行排列。那么我们就输入一些数值吧（每输入一个数值，用回车键来换行，接着输入下一个数值。输入 `END` 来结束输入，`END` 被称为结束字符串。当然了，你可以用其他字符串，比如 `haha`，`input`，不一定要用 `END`。我用 `END` 是因为 `end` 是“结束”的意思）：

```
oscar@oscar-laptop: ~/r
File Edit View Search Terminal Help
oscar@oscar-laptop:~/redirect$ sort -n << END
> 12
> 9
> 27
> 45
> 7
> 10
> 99
> 120
> 73
> END
7
9
10
12
27
45
73
99
120
oscar@oscar-laptop:~/redirect$ 
```

可以看到，`sort -n` 命令将我们输入的一串数值进行了由小到大的排序。

我们再试试其他命令与 << 符号的配合，这次我们用 `wc` 命令吧。`wc` 命令用于统计字符等，配合 `-m` 参数可以统计字符数。

```
wc -m << END
```

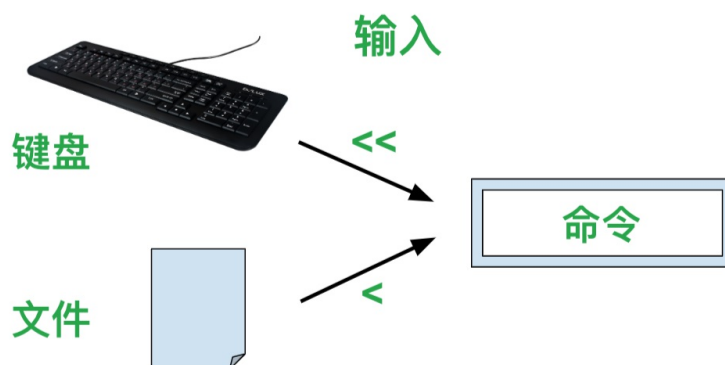
```
oscar@oscar-laptop: ~/redirect
File Edit View Search Terminal Help
oscar@oscar-laptop:~/redirect$ wc -m << END
> How many characters are there in this sentence ?
> END
49
oscar@oscar-laptop:~/redirect$ 
```

可以看到，“How many characters are there in this sentence ?”这句话中有49个字符。

这句话翻成中文就是“这句话中有多少个字符？”

小结

- <: 将命令的输入重定向为文件内容。
- <<: 将命令的输入重定向为键盘输入，以逐行输入的模式（回车键换行），所有输入的行都将在输入结束字符串（例如上面例子中的 END）之后发送给命令。



当然了，我们也可以将之前学习的输出重定向符号和这一节的输入重定向符号结合使用：

```
sort -n << END > numbers_sorted.txt 2>&1
```

```
oscar@oscar-laptop: ~/redirect
File Edit View Search Terminal Help
oscar@oscar-laptop:~/redirect$ sort -n << END > numbers_sorted.txt 2>&1
> 12
> 7
> 10
> 56
> 101
> 84
> END
oscar@oscar-laptop:~/redirect$ cat numbers_sorted.txt
7
10
12
56
84
101
oscar@oscar-laptop:~/redirect$
```

如上图所见，以上命令将 `sort -n` 命令对数值排序的结果都输入到 `numbers_sorted.txt` 文件中，如果有错误信息也输入。

3. |：管道

这最后一节要学的符号，将会成为你以后 Linux 生涯中最常用的符号之一，使用率绝对高过这两课学的其他符号，因此作为“压轴曲目”非常合适。

这个符号就是传说中的“管道符号”：

```
|
```

是的，就是美式键盘上位于反斜杠那个键上方位置的符号。要输入这个符号，需要使用 `Shift + \`。

| 符号既然被称为“管道符号”，那么其作用就是“建立命令管道”咯。

是的，还记得之前我们提到的那个比喻吗？一个命令的输出可以作为另一个命令的输入，就好像将两根水管接起来一样，前一根水管流出来的水就会流入后一根水管了。

管道也算是重定向的一种。

原理

将两个命令连成管道，这是什么意思呢？简单地说就是将一个命令的输出作为另一个命令的输入，如下图所示：



大致来说：命令 1 的输出立即会变成命令 2 的输入。使用这个原理，我们可以用 | 符号连接很多个命令，构成很长的命令管道。

管道符绝对使 Linux 命令的威力倍增。之前我们也说了，Linux 中的命令（很多都是从 Unix 时代承袭下来的），每一个的功能虽然有限，但是却在它们自己的岗位上“尽忠职守”，工作做得棒棒的。

所以单独一条 Linux 命令可能功能有限，但是一旦“铁索连环”，那可是会结合各个命令的功能，不难想见其强大。

实践

我们用几个实例来学习管道吧。

按学生名字排序

你应该还记得我们的 notes.csv 文件，其中有三部分，用逗号隔开的，第一部分是学生名字，第二部分是成绩，第三部分是评语。

之前我们用 `cut -d , -f 1 notes.csv` 命令来剪切了名字那一列。我们也学了 sort 命令，它用于排序文件内容。

那么为什么不把这两个命令用管道符连接起来呢？我们可以用 sort 命令对 cut 命令提取到的名字列进行排序：

```
cut -d , -f 1 notes.csv | sort
```

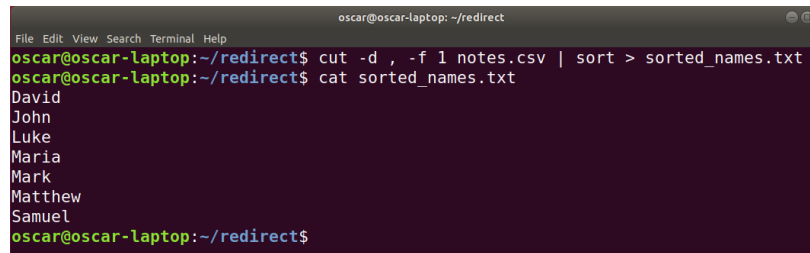
```
oscar@oscar-laptop: ~/redirect
File Edit View Search Terminal Help
oscar@oscar-laptop:~/redirect$ cut -d , -f 1 notes.csv | sort
David
John
Luke
Maria
Mark
Matthew
Samuel
oscar@oscar-laptop:~/redirect$
```

如上图所见，我们用 sort 命令对名字列按照首字母的字典顺序进行了排序。

怎么样，管道是不是很有意思？

如果我们将上面的命令再扩充一下，配合之前学习的输出重定向符号，就变成了这样：

```
cut -d , -f 1 notes.csv | sort > sorted_names.txt
```



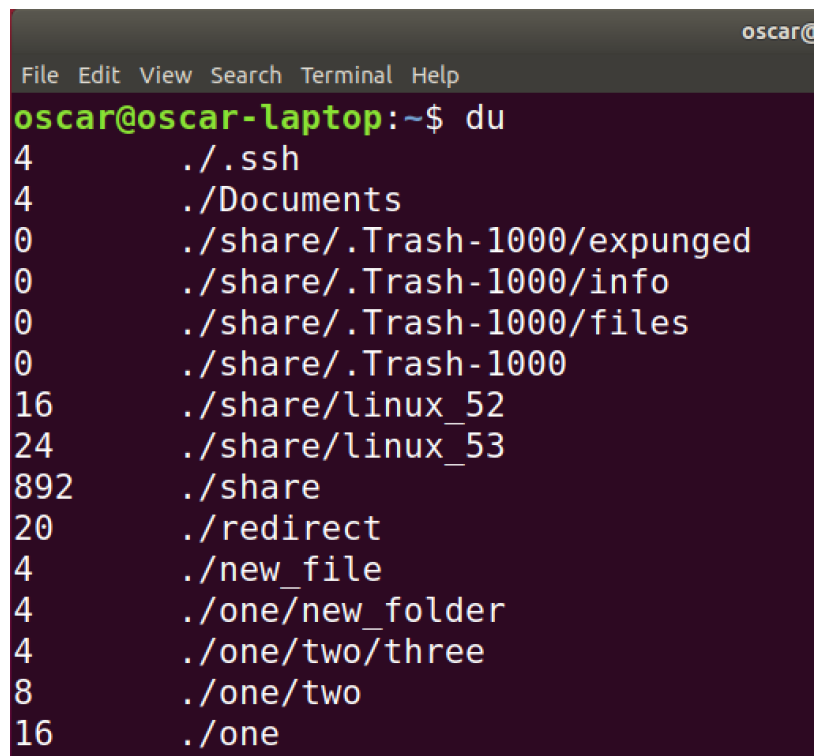
```
oscar@oscar-laptop: ~/redirect
File Edit View Search Terminal Help
oscar@oscar-laptop:~/redirect$ cut -d , -f 1 notes.csv | sort > sorted_names.txt
oscar@oscar-laptop:~/redirect$ cat sorted_names.txt
David
John
Luke
Maria
Mark
Matthew
Samuel
oscar@oscar-laptop:~/redirect$
```

可以看到，我们将 `sort` 命令排序的结果重定向到 `sorted_names.txt` 文件中了。

根据大小排序目录

之前我们学过，`du` 命令可以深入遍历当前目录下每个子目录，把所有文件的大小都做一个统计。

我们可以用 `cd` 命令来回到我们的家目录，然后运行 `du` 命令。



```
oscar@oscar-laptop: ~$ du
4      ./ssh
4      ./Documents
0      ./share/.Trash-1000/expunged
0      ./share/.Trash-1000/info
0      ./share/.Trash-1000/files
0      ./share/.Trash-1000
16     ./share/linux_52
24     ./share/linux_53
892    ./share
20     ./redirect
4      ./new_file
4      ./one/new_folder
4      ./one/two/three
8      ./one/two
16     ./one
```

单独用 `du` 命令的缺点我们也有目共睹了：一是可能要运行很久（如果文件很多的话），二是显示结果没有排序，杂乱无章。

但如果我们这样做就会清爽很多：

```
du | sort -nr | head
```

还记得 `head` 命令的用法么？如果不用 `-n` 参数指定显示行数，那么 `head` 会默认显示前 10 行。

```
File Edit View Search Terminal Help
oscar@oscar-laptop: ~$ du | sort -nr | head
40744 .
20348 ~/.cache
17480 ~/.mozilla
17468 ~/.mozilla/firefox
17444 ~/.mozilla/firefox/jm1m5hs9.default
17432 ~/.cache/mozilla
17428 ~/.cache/mozilla/firefox
17424 ~/.cache/mozilla/firefox/jm1m5hs9.default
8424 ~/.cache/mozilla/firefox/jm1m5hs9.default/startupCache
6924 ~/.cache/mozilla/firefox/jm1m5hs9.default/safebrowsing
oscar@oscar-laptop: ~$
```

所以上命令的作用是：

- `du`：深入遍历当前目录下每个子目录，把所有文件的大小都做一个统计。
- `sort -nr`：`sort` 命令的 `-n` 参数是以数值（此处是文件大小）排序，默认是小的在前；`-r` 参数是倒序排列，有了 `-r` 参数，排序结果就变成大的数值在前了。
- `head`：列出前 10 行。

列出包含关键字的文件

还记得我们的好朋友 `grep` 命令吗？这个命令很强大，可以在文件中查找关键字，并且显示关键字所在的行。但有时，我们会觉得 `grep` 显示的信息太冗长了，每一行不仅有文件名，还有关键字出现的那一行文本，等等。

我们可以运行以下命令试试：

```
sudo grep log -Ir /var/log | cut -d : -f 1 | sort | uniq
```

```
File Edit View Search Terminal Help
oscar@oscar-laptop: ~$ sudo grep log -Ir /var/log | cut -d : -f 1 | sort | uniq
/var/log/alternatives.log
/var/log/apt/history.log
/var/log/apt/term.log
/var/log/auth.log
/var/log/auth.log.1
/var/log/bootstrap.log
/var/log/dpkg.log
/var/log/gpu-manager.log
/var/log/installer/partman
/var/log/installer/syslog
/var/log/kern.log
/var/log/kern.log.1
/var/log/syslog
/var/log/syslog.1
oscar@oscar-laptop: ~$
```

这个命令做了什么呢？让我们逐步来分解：

- `sudo grep log -Ir /var/log`：遍历 `/var/log` 这个目录及其子目录，列出所有包含 `log` 这个关键字的行。`-I` 参数用于排除二进制文件。`-r` 参数用于递归遍历。`sudo` 命令是为了以 `root` 身份查找系统文件夹 `/var/log`。
- `cut -d : -f 1`：从 `sudo grep log -Ir /var/log` 的输出结果中只剪切出文件名那一列（由冒号分隔的第一个区域）。
- `sort`：将文件名的列以首字母的字典顺序进行排序。
- `uniq`：去掉重复的文件名。

小结

小结很简单，因为管道符的基本作用比较简单，就是将一个命令的输出重定向为另一个命令的输入。如下图所示：



命令 1 的输出，命令 2 的输入

记住这个基本原理就好了。

这部分内容终于结束了。上一课和这一课的新概念比较多，如果觉得还没怎么掌握，那需要再读一遍，也可以参考一些课外读物，自己在终端上多练习。

尽情发挥你的想象力，来创造出各种命令的组合吧。

相信经过这两课的“洗礼”，您的“三观”应该更正了吧。

小结

1. < 符号重定向命令的输入为文件内容；<< 符号重定向命令的输入为键盘输入。
2. 管道符号 | 可以将命令连接起来，好像一个个对接的管道一样，前一个命令的输出成为后一个命令的输入。

今天的课就到这里，一起加油吧！