


# xargs 命令教程

作者：阮一峰

日期：2019年8月 8日

**xargs** 是 Unix 系统的一个很有用的命令，但是常常被忽视，很多人不了解它的用法。

本文介绍如何使用这个命令。



```
sanctuary — pi@tinypi: ~ — ssh pi@tinypi.local — 110x30
1 2 3
pi@tinypi:~$ ls | xargs rm
pi@tinypi:~$ ls
pi@tinypi:~$ ls
pi@tinypi:~$ printf "1\n2\n3\n" | xargs -i touch {}.txt
pi@tinypi:~$ ls
1.txt 2.txt 3.txt
pi@tinypi:~$ stat 1.txt
  File: '1.txt'
  Size: 0                Blocks: 0          IO Block: 4096   regular empty file
Device: b302h/45826d    Inode: 1986         Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/   pi)   Gid: ( 1000/   pi)
Access: 2017-04-10 22:54:36.410000000 -0700
Modify: 2017-04-10 22:54:36.410000000 -0700
Change: 2017-04-10 22:54:36.410000000 -0700
 Birth: -
pi@tinypi:~$ ls | xargs rm
pi@tinypi:~$ printf "1\t2\t3\t" | xargs -d "\t" touch {}
pi@tinypi:~$ ls
1 2 3
pi@tinypi:~$ stat 1
  File: '1'
  Size: 0                Blocks: 0          IO Block: 4096   regular empty file
Device: b302h/45826d    Inode: 1986         Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/   pi)   Gid: ( 1000/   pi)
Access: 2017-04-10 22:55:47.390000000 -0700
Modify: 2017-04-10 22:55:47.390000000 -0700
Change: 2017-04-10 22:55:47.390000000 -0700
 Birth: -
pi@tinypi:~$
```

## 一、标准输入与管道命令

Unix 命令都带有参数，有些命令可以接受"标准输入"（**stdin**）作为参数。

```
$ cat /etc/passwd | grep root
```

上面的代码使用了管道命令（**|**）。管道命令的作用，是将左侧命令（**cat /etc/passwd**）的标准输出转换为标准输入，提供给右侧命令（**grep root**）作为参数。

因为 **grep** 命令可以接受标准输入作为参数，所以上面的代码等同于下面的代码。

```
$ grep root /etc/passwd
```

但是，大多数命令都不接受标准输入作为参数，只能直接在命令行输入参数，这导致无法用管道命令传递参数。举例来说，`echo` 命令就不接受管道传参。

```
$ echo "hello world" | echo
```

上面的代码不会有输出。因为管道右侧的 `echo` 不接受管道传来的标准输入作为参数。

## 二、xargs 命令的作用

`xargs` 命令的作用，是将标准输入转为命令行参数。

```
$ echo "hello world" | xargs echo
hello world
```

上面的代码将管道左侧的标准输入，转为命令行参数 `hello world`，传给第二个 `echo` 命令。

`xargs` 命令的格式如下。

```
$ xargs [-options] [command]
```

真正执行的命令，紧跟在 `xargs` 后面，接受 `xargs` 传来的参数。

`xargs` 的作用在于，大多数命令（比如 `rm`、`mkdir`、`ls`）与管道一起使用时，都需要 `xargs` 将标准输入转为命令行参数。

```
$ echo "one two three" | xargs mkdir
```

上面的代码等同于 `mkdir one two three`。如果不加 `xargs` 就会报错，提示 `mkdir` 缺少操作参数。

## 三、xargs 的单独使用

`xargs` 后面的命令默认是 `echo` 。

```
$ xargs
# 等同于
$ xargs echo
```

大多数时候，`xargs` 命令都是跟管道一起使用的。但是，它也可以单独使用。

输入 `xargs` 按下回车以后，命令行就会等待用户输入，作为标准输入。你可以输入任意内容，然后按下 `Ctrl d`，表示输入结束，这时 `echo` 命令就会把前面的输入打印出来。

```
$ xargs
hello (Ctrl + d)
hello
```

再看一个例子。

```
$ xargs find -name
 "*.txt"
 ./foo.txt
 ./hello.txt
```

上面的例子输入 `xargs find -name` 以后，命令行会等待用户输入所要搜索的文件。用户输入 `"*.txt"`，表示搜索当前目录下的所有 **TEXT** 文件，然后按下 `Ctrl d`，表示输入结束。这时就相当执行 `find -name *.txt`。

## 四、-d 参数与分隔符

默认情况下，`xargs` 将换行符和空格作为分隔符，把标准输入分解成一个个命令行参数。

```
$ echo "one two three" | xargs mkdir
```

上面代码中，`mkdir` 会新建三个子目录，因为 `xargs` 将 `one two three` 分解成三个命令行参数，执行 `mkdir one two three`。

`-d` 参数可以更改分隔符。

```
$ echo -e "a\tb\tc" | xargs -d "\t" echo
a b c
```

上面的命令指定制表符 `\t` 作为分隔符，所以 `a\tb\tc` 就转换成了三个命令行参数。 `echo` 命令的 `-e` 参数表示解释转义字符。

## 五、-p 参数，-t 参数

使用 `xargs` 命令以后，由于存在转换参数过程，有时需要确认一下到底执行的是什么命令。

`-p` 参数打印出要执行的命令，询问用户是否要执行。

```
$ echo 'one two three' | xargs -p touch
touch one two three ?...
```

上面的命令执行以后，会打印出最终要执行的命令，让用户确认。用户输入 `y` 以后（大小写皆可），才会真正执行。

`-t` 参数则是打印出最终要执行的命令，然后直接执行，不需要用户确认。

```
$ echo 'one two three' | xargs -t rm
rm one two three
```

## 六、-o 参数与 find 命令

由于 `xargs` 默认将空格作为分隔符，所以不太适合处理文件名，因为文件名可能包含空格。

`find` 命令有一个特别的参数 `-print0`，指定输出的文件列表以 `null` 分隔。然后，`xargs` 命令的 `-0` 参数表示用 `null` 当作分隔符。

```
$ find /path -type f -print0 | xargs -0 rm
```

上面命令删除 `/path` 路径下的所有文件。由于分隔符是 `null`，所以处理包含空格的文件名，也不会报错。

还有一个原因，使得 `xargs` 特别适合 `find` 命令。有些命令（比如 `rm`）一旦参数过多会报错“参数列表过长”，而无法执行，改用 `xargs` 就没有这个问题，因为它对每个参数执行一次命令。

```
$ find . -name "*.txt" | xargs grep "abc"
```

上面命令找出所有 `TEXT` 文件以后，对每个文件搜索一次是否包含字符串 `abc`。

## 七、-L 参数

如果标准输入包含多行，`-L` 参数指定多少行作为一个命令行参数。

```
$ xargs find -name  
"*.txt"  
"*.md"  
find: paths must precede expression: `*.md'
```

上面命令同时将 `"*.txt"` 和 `*.md` 两行作为命令行参数，传给 `find` 命令导致报错。

使用 `-L` 参数，指定每行作为一个命令行参数，就不会报错。

```
$ xargs -L 1 find -name  
"*.txt"  
./foo.txt  
./hello.txt  
"*.md"  
./README.md
```

上面命令指定了每一行（`-L 1`）作为命令行参数，分别运行一次命令（`find -name`）。

下面是另一个例子。

```
$ echo -e "a\nb\nc" | xargs -L 1 echo  
a  
b  
c
```

上面代码指定每行运行一次 `echo` 命令，所以 `echo` 命令执行了三次，输出了三行。

## 八、-n 参数

`-L` 参数虽然解决了多行的问题，但是有时用户会在同一行输入多项。

```
$ xargs find -name  
"*.txt" "*.md"  
find: paths must precede expression: `*.md'
```

上面的命令将同一行的两项作为命令行参数，导致报错。

`-n` 参数指定每次将多少项，作为命令行参数。

```
$ xargs -n 1 find -name
```

上面命令指定将每一项（`-n 1`）标准输入作为命令行参数，分别执行一次命令（`find -name`）。

下面是另一个例子。

```
$ echo {0..9} | xargs -n 2 echo  
0 1  
2 3  
4 5  
6 7  
8 9
```

上面命令指定，每两个参数运行一次 `echo` 命令。所以，10个阿拉伯数字运行了五次 `echo` 命令，输出了五行。

## 九、-I 参数

如果 `xargs` 要将命令行参数传给多个命令，可以使用 `-I` 参数。

`-I` 指定每一项命令行参数的替代字符串。

```
$ cat foo.txt  
one  
two  
three
```

```
$ cat foo.txt | xargs -I file sh -c 'echo file; mkdir file'
one
two
three

$ ls
one two three
```

上面代码中，`foo.txt` 是一个三行的文本文件。我们希望对每一项命令行参数，执行两个命令（`echo` 和 `mkdir`），使用 `-I file` 表示 `file` 是命令行参数的替代字符串。执行命令时，具体的参数会替代掉 `echo file; mkdir file` 里面的两个 `file`。

## 十、--max-procs 参数

`xargs` 默认只用一个进程执行命令。如果命令要执行多次，必须等上一次执行完，才能执行下一次。

`--max-procs` 参数指定同时用多少个进程并行执行命令。`--max-procs 2` 表示同时最多使用两个进程，`--max-procs 0` 表示不限制进程数。

```
$ docker ps -q | xargs -n 1 --max-procs 0 docker kill
```

上面命令表示，同时关闭尽可能多的 Docker 容器，这样运行速度会快很多。

## 十一、参考链接

- [Linux and Unix xargs command tutorial with examples](#), George Ornbo
- [8 Practical Examples of Linux Xargs Command for Beginners](#), Himanshu Arora

（完）

### 文档信息

- 版权声明：自由转载-非商用-非衍生-保持署名（[创意共享3.0许可证](#)）
- 发表日期：2019年8月 8日