xargs原理及使用

<https://www.cnblogs.com/yorkyang/p/6294879.html>

# xargs简介

之所以能用到这个命令，是由于**很多 linux 命令不支持用管道传递参数**，

**管道是把一个命令的输出传递给另一个命令作为输入**。

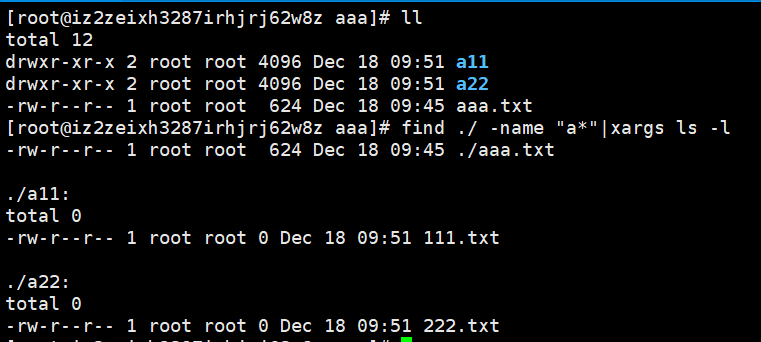
例如 find /sbin -perm +700 | ls -l　　 这个命令是错误的

　　 find /sbin -perm +700 |**xargs** ls -l 这样才是正确的

　　通常Linux命令可以用**|**首尾相连，**上一个命令的 stdout** 连接到**下一个命令的 stdin**。但是有些命令，比如ls、rm等，是从**命令行参数**接收输入的。这时候如果想把上一个命令的输出传给它们，就不好办了，所以就有了xargs。

　　简单而言，xargs可以把从 stdin 接收到的输入，按**空白符或回车符**分隔开，然后依次作为参数去调用**xargs后面的命令。**

xargs 默认的分隔符：**空格 或 回车**。



find ./ -name "a\*"|xargs ls –l

相当于分别找到了aaa.txt文件、a11目录、a22目录；

然后依次执行：ls –l aaa.txt ls –l a11 ls –l a22

# xargs命令与find的搭配使用

在使用find命令的-exec选项处理匹配到的文件时， find命令将所有匹配到的文件一起传递给exec执行。但有些系统对能够传递给exec的命令长度有限制，这样在find命令运行几分钟之后，就会出现溢出错误。错误信息通常是**“参数列太长”或“参数列溢出”**。这就是xargs命令的用处所在，特别是与find命令一起使用。find命令把匹配到的文件传递给xargs命令，而xargs命令每次只获取一部分文件而不是全部，不像-exec选项那样。这样它可以先处理最先获取的一部分文件，然后是下一 批，并如此继续下去。

在有些系统中，使用-exec选项会 为处理每一个匹配到的文件而发起一个相应的进程，并非将匹配到的文件全部作为参数一次执行；这样在有些情况下就会出现进程过多，系统性能下降的问题，因而 效率不高；而使用xargs命令则只有一个进程。另外，在使用xargs命令时，究竟是一次获取所有的参数，还是分批取得参数，以及每一次获取参数的数目 都会根据该命令的选项及系统内核中相应的可调参数来确定。

**管道是把一个命令的输出传递给另一个命令作为输入**，比如：command1 | command2但是command2仅仅把输出的内容作为输入参数。find . -name "install.log" -print打印出的是install.log这个字符串，如果仅仅使用管道，那么command2能够使用的仅仅是install.log这个字符串， 不能把它当作文件来进行处理。

xargs就是为了能够对find搜索到的文件进行操作而编写的，它能把管道传来的字符串当作文件交给其后的命令执行。

举个例子：  
$find . -name "install.log" -print | cat  
./install.log                      #显示从管道传来的内容，仅仅作为字符串来处理  
$find . -name "install.log" -print | xargs cat  
aaaaaa                        #将管道传来的内容作为文件，交给cat执行。也就是说，该命令执行的是如果存在install.log，那么就打印出这个文件的内容。  
来看看xargs命令是如何同find命令一起使用的，并给出一些例子。

### 1、在当前目录下查找所有用户具有读、写和执行权限的文件，并收回相应的写权限： # find . -perm -7 -print | xargs chmod o-w

### 2、查找系统中的每一个普通文件，然后使用xargs命令来测试它们分别属于哪类文件 # find . -type f -print | xargs file ./liyao: empty

### 3、尝试用rm 删除太多的文件，你可能得到一个错误信息：/bin/rm Argument list too long. 用xargs 去避免这个问题 $find ~ -name ‘\*.log’ -print0 | xargs -i -0 rm -f {}

### 4、查找所有的jpg 文件，并且压缩它 # find / -name \*.jpg -type f -print | xargs tar -cvzf images.tar.gz

### 5、拷贝所有的图片文件到一个外部的硬盘驱动  # ls \*.jpg | xargs -n1 -i cp {} /external-hard-drive/directory

# 参数使用

想把所有.jpg文件删除，当然你可以 rm \*.jpg，但是如果要递归操作所有子目录下的文件呢？

　　可以这样：

* **find . -name "\*.jpg" | xargs rm**

　　　这样，所有被find找到的文件名，都会作为参数来调用rm命令了

　　  对于大多数情况，这一行命令没有问题，但是如有些文件名中包含空格，就会有问题了。

　　　xargs默认以空白符分隔接受到的输入，所以一个含有空格的文件名会被当做多个参数，分别传给rm。所以在处理文件名这类命令时，通常要这样：

* **find . -name "\*.jpg" -print0 | xargs -0 rm**

       这里的 -print0 是告诉find命令，在每个输出后面以'\0'作为结束。-0是告诉xargs，使用'\0'来分隔输入，而不是空白符。这样就避免出现问题了。

      下面再考虑另一种情况，假设不是删除，而是想把符合要求的文件名都添加上后缀.bak怎么办？这时候需要这样：

* **find . -name "\*.jpg" -print0 | xargs -0 -I {} mv {} {}.bak**

　　其中的-I {}是告诉xargs，后面的命令中，用{}表示占位符，将会被实际的参数替代。这样就行了。

　　其他有用的参数还有：   
　　-n 用于指定每次传递几个参数   
　　-d 用于指定切分输入内容时，具体的分隔符   
　　其他更多的参数，就参考man xargs吧。

* -a file 从文件中读入作为sdtin

　　　　$ cat 1.txt   
　　　　aaa  bbb ccc ddd  
　　　　a    b  
　　　　$ xargs -a 1.txt echo  
　　　　aaa bbb ccc ddd a b

* -e flag ，注意有的时候可能会是-E，flag必须是一个以空格分隔的标志，当xargs分析到含有flag这个标志的时候就停止

　　　　$ xargs -E 'ddd'  -a 1.txt echo  
　　　　aaa bbb ccc

　　　　$ cat 1.txt |xargs -E 'ddd' echo  
　　　　aaa bbb ccc

* -n num 后面加次数，表示命令在执行的时候一次用的argument的个数，默认是用所有的。

　　　　$ cat 1.txt |xargs -n 2 echo  
　　　　aaa bbb  
　　　　ccc ddd  
　　　　a b

* -p 操作具有可交互性，每次执行comand都交互式提示用户选择，当每次执行一个argument的时候询问一次用户

　　　　$ cat 1.txt |xargs -p echo  
　　　　echo aaa bbb ccc ddd a b ?...y  
　　　　aaa bbb ccc ddd a b  
　　　　$ cat 1.txt |xargs -p echo  
　　　　echo aaa bbb ccc ddd a b ?...n

* -t 表示先打印命令，然后再执行。

　　　　$ cat 1.txt |xargs -t echo  
　　　　echo aaa bbb ccc ddd a b   
　　　　aaa bbb ccc ddd a b

* -i 或者是-I，这得看linux支持了，将xargs的每项名称，一般是一行一行赋值给{}，可以用{}代替。

　　　　$ ls  
　　　　1.txt  2.txt  3.txt  log.xml  
　　　　$ ls \*.txt | xargs -t -i mv {} {}.bak  
　　　　mv 1.txt 1.txt.bak   
　　　　mv 2.txt 2.txt.bak   
　　　　mv 3.txt 3.txt.bak   
　　　　$ ls  
　　　　1.txt.bak  2.txt.bak  3.txt.bak  log.xml

　　注意，-I 必须指定替换字符　－i 是否指定替换字符-可选，例如：  
　　　　find . | xargs -I {} cp {} $D\_PATH  
　　　　与  
　　　　find . | xargs -i cp {} $D\_PATH

　　注意：cshell和tcshell中，需要将{}用单引号、双引号或反斜杠，否则不认识。bash可以不用。  
　　find /shell -maxdepth 2 -name a -print | xargs -t -i sed -i '1 i\111' ‘{}‘

* -r  no-run-if-empty 如果没有要处理的参数传递给xargsxargs 默认是带 空参数运行一次，如果你希望无参数时，停止 xargs，直接退出，使用 -r 选项即可，其可以防止xargs 后面命令带空参数运行报错。

　　　　$ echo ""|xargs -t mv  
　　　　mv   
　　　　mv: missing file operand  
　　　　Try `mv --help' for more information.  
　　　　$ echo ""|xargs -t -r mv         #直接退出  
　　　　-s num xargs后面那个命令的最大命令行字符数(含空格)   
　　　　$ cat 1.txt.bak |xargs  -s 9 echo  
　　　　aaa  
　　　　bbb  
　　　　ccc  
　　　　ddd  
　　　　a b  
　　　　$ cat 1.txt.bak |xargs  -s 4 echo  
　　　　xargs: can not fit single argument within argument list size limit      #length(echo)=4  
　　　　$ cat 1.txt.bak |xargs  -s 8 echo  
　　　　xargs: argument line too long      #length(echo)=4,length(aaa)=3,length(null)=1,total\_length=8

* -L  从标准输入一次读取num行送给Command命令 ，-l和-L功能一样

　　　　$ cat 1.txt.bak   
　　　　aaa bbb ccc ddd  
　　　　a b  
　　　　ccc  
　　　　dsds  
　　　　$ cat 1.txt.bak |xargs  -L 4 echo  
　　　　aaa bbb ccc ddd a b ccc dsds  
　　　　$ cat 1.txt.bak |xargs  -L 1 echo  
　　　　aaa bbb ccc ddd  
　　　　a b  
　　　　ccc  
　　　　dsds  
　　　　-d delim 分隔符，默认的xargs分隔符是回车，argument的分隔符是空格，这里修改的是xargs的分隔符

　　　　$ cat 1.txt.bak   
　　　　aaa@ bbb ccc@ ddd  
　　　　a b

　　　　$ cat 1.txt.bak |xargs  -d '@' echo  
　　　　aaa  bbb ccc  ddd  
　　　　a b

　　　　-x exit的意思，如果有任何 Command 行大于 -s Size 标志指定的字节数，停止运行 xargs 命令，-L -I -n 默认打开-x参数，主要是配合-s使用  
　　　　-P 修改最大的进程数，默认是1，为0时候为as many as it can 。