原创

Xargs的使用

xaverlin 2018-01-11 13:57:31 3898人阅读 0人评论

Xargs的使用

发现xargs是个很好用的命令，于是搜索了网上好多关于xargs的文章并做了相关的实验，现自己总结一下，好转化为自己的东西：

Xargs经常会与管道搞混淆，所以

首先我们先了解下管道的作用：

管道实现的是：将前面的stdout作为后面的stdin进行传递。但有些命令是不接受管道的传递方式，比如：ls （*#echo '/root'| ls* ）

， 管道传递过来的stdin只能作为内容而不是参数。或者说是管道无法正确传递到后面命令的参数位：（*#echo '--help'| cat ,区别于**#echo** '--help'| xargs cat* ）

鉴于管道功能上的局限，所以xargs的作用就显现出来了：

f Xargs实现的是将管道传输过来的stdin进行处理后，传递到命令的参数上。即xargs完成了两个动作：A.处理管道传输过来的stdin B.将处理后的stdin传递到正确的位置上

④ Xargs的作用不仅仅限于简单的stdin传递到命令的参数位。它还可以将stdin或者文件stdin分割成批，每个批中有很多分割片段，然后将这些片段按批交给xargs后面的命令进行处理。

⑤ Xargs默认其后面的命令是echo（***“echo ‘a.txt’ | xargs ” = ” echo ‘a.txt’ | xargs echo “***）而空格是其默认的定界符。

⑥ Xargs会把管道传递过来的stdin里的换行符，空白换成空格取代，形成一行作为整体输出：

[root@jmtom tmp]# cat c.txt

a.txt

b.txt

[root@jmtom tmp]# cat c.txt | xargs

a.txt b.txt

即：如果不指定分批选项（-i，-L，-n），Xargs的一整行结果将作为一个整体输出，而不是分隔开的！

分割选项：-d ，-0 | 分批选项： -n，-L，-i/-I | 使用-p或-t来观察命令的执行过程

分割选项 -d， -0：分割前会对stdin分批做一个分行动作，然后才进行分割

-d 指定分隔符

```
[root@jmtom testfile]# cat list.txt
a1.bak,a2.bak,a3.bak,a4.bak
[root@jmtom testfile]# cat list.txt | xargs -d','
a1.bak a2.bak a3.bak a4.bak
```

<——分割后输出会多出分行符，会对后续的命令有影响

```
[root@jmtom testfile]# cat list.txt | xargs -d',' cat
this is a1
this is a2
this is a2.
this is a3
this is a3.
this is a3..
cat: a4.bak
```

: 没有那个文件或目录<——这里就因为分割输出多出的那一空行导致**a4.bak**文件无法读出

```
[root@jmtom testfile]#
```

正确的命令应该为：`cat list.txt | xargs -d',' |xargs cat`

-0 指定固定的\0作为分隔符。

其实xargs -0就是特殊的xargs -d的一种，它等价于xargs -d"\0"。

（‘\0’表示字符串结束，即字符串都是以’\0’结束的）

如果使用xargs -0时不指定分批选项（-n、-L、-i），则处理后的结果将作为一个整体输出。

```
[root@jmtom testfile]# ls | xargs
a1.bak a2.bak a3.bak a4.bak a5.bak a8.bak a9.bak list1.txt list.txt
[root@jmtom testfile]# ls | xargs -0
a1.bak
a2.bak
a3.bak
a4.bak
a5.bak
a8.bak
a9.bak
list1.txt
list.txt
```

```
[root@jmtom testfile]#
```

如果指定了分批选项，但没有检测到null字符，则整个结果将称为一个不可分割整体，这时使用分批选项是完

全无意义的。

```
[root@jmtom testfile]# ls | xargs -0 -n 3
a1.bak
a2.bak
a3.bak
a4.bak
a5.bak
a8.bak
a9.bak
list1.txt
list.txt

[root@jmtom testfile]#
```

如果指定了分批选项，并且检测到了null字符，则以\0位的空格分段划批，这时使用-n、-L或-i的结果是一样的

```
[root@jmtom testfile]# ls |tr "\n" "\0"| xargs -0 -n 3
a1.bak a2.bak a3.bak
a4.bak a5.bak a8.bak
a9.bak list1.txt list.txt
[root@jmtom testfile]# ls |tr "\n" "\0"| xargs -0 -L 3
a1.bak a2.bak a3.bak
a4.bak a5.bak a8.bak
a9.bak list1.txt list.txt
[root@jmtom testfile]#
```

find -print0可以输出\0字符 ，所以会和xargs -0 结合一起来用

分批选项：-n,-L,-i/-I: 分批用于指定每次传递多少个分段(参数）。

-i选项优先级最高，-L选项次之，-n选项优先级最低。

多个分批选项同时指定时，高优先级的选项会覆盖低优先级的选项。也就是说这时候指定低优先级的选项是无意义的。

-n 该选项表示将xargs生成的命令行参数，每次传递几个参数给其后面的命令执行

-L 和-n选项类似，唯一的区别是-L永远是按段分段划批，而-n在和独立的xargs一起使用是按空格分段划批的。

-i 选项在逻辑上用于接收传递的分批结果。

-I 和-i是一样的，只是-i默认使用大括号作为替换符号，-I则可以指定其他的符号、字母、数字作为替换符号，但是必须用引号包起来。

实例：

```
[root@jmtom testfile]# ls
a1.bak  a2.bak  a3.bak  a4.bak  a5.bak  a8.bak  a9.bak  list1.txt  list.txt  my note.log
[root@jmtom testfile]# ls |xargs -n3 <--  这里-n以空格分段划批
a1.bak a2.bak a3.bak
```

```
a4.bak a5.bak a8.bak
a9.bak list1.txt list.txt
my note.log
[root@jmtom testfile]#
[root@jmtom testfile]# ls |xargs -d'k' -n3 <--  -n跟-d配合使用，按有‘k’段分批
a1.ba
a2.ba
a3.ba

a4.ba
a5.ba
a8.ba

a9.ba
list1.txt
list.txt
my note.log

[root@jmtom testfile]#
---------------------- ------------------ ------------- ----------------------
[root@jmtom testfile]# ls | xargs   -n1
a1.bak
a2.bak
a3.bak
a4.bak
a5.bak
a8.bak
a9.bak
list1.txt
list.txt
my                    <-- -n这里是以空格为段划批，这里的my不是存在的文件名
note.log            <-- “my   note.log”才是正确的文件名
[root@jmtom testfile]# ls | xargs   -L1
a1.bak
a2.bak
a3.bak
a4.bak
a5.bak
a8.bak
a9.bak
list1.txt
list.txt
my note.log      <-- -L这里是一直以段为段划批
[root@jmtom testfile]#
------ ----------   ---------------   ----------
```

-i 选项在逻辑上用于接收传递的分批结果。

即如果不使用-i，则默认是将分割后处理后的结果整体传递到命令的最尾部。而用了-i可以传递多个参数位，而不是仅仅最尾端的参数位。例如重命名备份的时候在每个传递过来的文件名加上后缀.bak，这需要两个参数位。

使用xargs -i时以大括号{}作为替换符号，传递的时候看到{}就将被结果替换。可以将{}放在任意需要传递的参数位上，如果多个地方使用{}就实现了多个传递。

```
[root@jmtom one]# touch day{1..7}
[root@jmtom one]# ls
day1   day2   day3   day4   day5   day6   day7
[root@jmtom one]# ls | xargs -i mv {}   {}.bak <-- 完成了两个参数位的传递
[root@jmtom one]# ls
```

```
day1.bak  day2.bak  day3.bak  day4.bak  day5.bak  day6.bak  day7.bak
[root@jmtom one]#
```

-I只是可以自定义其替换符号：

```
[root@jmtom one]# ls | xargs -Ix mv x x.bak <-- 跟上面的命令效果一样
[root@jmtom testfile]# ls | xargs  -L3
a1.bak a2.bak a3.bak
a4.bak a5.bak a8.bak
a9.bak list1.txt list.txt
my note.log
------ ------------ ------------
```

体会一下有-i和没有-i的区别：

```
[root@jmtom testfile]# ls | xargs  -L3  -i  cat {}
this is a1
this is a2
this is a2.
this is a3
this is a3.
this is a3..
this is a4
this is a4.
this is a4..
this is a4...
a1.bak,a2.bak
a3.bak,a4.bak
a5.bak,a8.bak
a9.bak
a1.bak,a2.bak,a3.bak,a4.bak
i am log of my note
[root@jmtom testfile]# ls | xargs -L3 cat
this is a1
this is a2
this is a2.
this is a3
this is a3.
this is a3..
this is a4
this is a4.
this is a4..
this is a4...
a1.bak,a2.bak
a3.bak,a4.bak
a5.bak,a8.bak
a9.bak
a1.bak,a2.bak,a3.bak,a4.bak
cat: my: 没有那个文件或目录
cat: note.log: 没有那个文件或目录   <-- 可见没有-i传递的参数会被误解分割了
[root@jmtom testfile]#
```


使用-p或-t来观察命令的执行过程

```
[root@jmtom testfile]# ls | xargs -L3 -p cat
cat a1.bak a2.bak a3.bak ?...yes   <-- 询问是否这样执行命令，输入y或yes将执行
cat a4.bak a5.bak a8.bak ?...this is a1
this is a2
this is a2.
this is a3
this is a3.
this is a3..
this is a4
this is a4.
this is a4..
this is a4...
a1.bak,a2.bak
a3.bak,a4.bak
a5.bak,a8.bak
a9.bak
a1.bak,a2.bak,a3.bak,a4.bak
cat: my: 没有那个文件或目录
cat: note.log: 没有那个文件或目录
[root@jmtom testfile]#
```

```
this is a3
this is a3.
this is a3..
yes   <--- 同上
cat a9.bak list1.txt list.txt ?...this is a4
this is a4.
this is a4..
this is a4...
yes   <-- 同上
cat my note.log ?...a1.bak,a2.bak
a3.bak,a4.bak
a5.bak,a8.bak
a9.bak
a1.bak,a2.bak,a3.bak,a4.bak
yes   <--   同上
cat: my: 没有那个文件或目录   <-- 这里就看出来分批传递到这里后出问题了因为cat没法对“my   note.log”文件处理，它分解成了my和note.log两个不存在的文件，所以提示没有那个文件或者目录
cat: note.log: 没有那个文件或目录
[root@jmtom testfile]#
[root@jmtom testfile]# ls | xargs -L3 -i -p cat {}
cat a1.bak ?...y
cat a2.bak ?...this is a1
y
cat a3.bak ?...this is a2
this is a2.
y
cat a4.bak ?...this is a3
this is a3.
this is a3..
y
cat a5.bak ?...this is a4
this is a4.
this is a4..
this is a4...
y
cat a8.bak ?...y
cat a9.bak ?...y
cat list1.txt ?...y
cat list.txt ?...a1.bak,a2.bak
a3.bak,a4.bak
a5.bak,a8.bak
a9.bak
y
cat my note.log ?...a1.bak,a2.bak,a3.bak,a4.bak
y
i am log of my note
[root@jmtom testfile]#
```

这里# ls | xargs -L3 -i -p cat {}就可以看出来-i是按逻辑上一个一个传递参数（-i按分段来传递，无法按批来传递），这里-i的优先权大于-L，所以这里的-L是无意义的。

⑧ xargs主要是为find服务的

我们要删除一个带空格的文件：

```
[root@jmtom one]# touch a b c "my   note.log"
[root@jmtom one]# ls
a  b  c  my   note.log
```

```
[root@jmtom one]#
[root@jmtom one]# find . -name *.log
./my  note.log
[root@jmtom one]#  find . -name *.log | xargs rm -fr <- 这样是没法删除的，rm只是解读为删除./my 的文件，并不是./my note.log这个文件
[root@jmtom one]# ls
a  b  c  my  note.log
[root@jmtom one]#  find . -name *.log -print0| xargs -0 rm -fr  <-- 通常用find -print0来结合xargs -0 来处理，但没有通用性，别的命令无法用这个方法
[root@jmtom one]# ls
a  b  c
[root@jmtom one]# touch "my note.log"
[root@jmtom one]#  find . -name *.log | xargs -i rm -fr "{}"  <-- 这种方法相对会有通用性，别的命令也能使用
[root@jmtom one]# ls
a  b  c
[root@jmtom one]#
```

Ps：

Linux命令一般可以从两个地方读取要处理的内容，一个是通过命令行参数，二是通过标准输入。但是很多命令的设计是先从命令行参数中获取参数，然后才是从标准输入stdin中读取。（大多数命令有一个参数 “-”，来表示从标准输入中读取）

```
[root@jmtom tmp]# cat a.txt
this is a.txt
i love u !
u love me ?
[root@jmtom tmp]# echo "my love" | cat a.txt <- 只会先读取命令行参数所以只读了a.txt
this is a.txt
i love u !
u love me ?
[root@jmtom tmp]# echo "my love" | cat a.txt -   <- 这里多加了一个横杠，表stdin，所以先读了a.txt再读了echo的内容（注意先后顺序）
this is a.txt
i love u !
u love me ?
my love
[root@jmtom tmp]# echo "my love" | grep 'love'  a.txt
i love u !
u love me ?
[root@jmtom tmp]# echo "my love" | grep 'love'  a.txt -
a.txt:i love u !
a.txt:u love me ?
(标准输入):my love
[root@jmtom tmp]#
```


xargs 管道

1 收藏 分享

上一篇：ESXI上的新建虚机绑定已使用过... 下一篇：给目录增加容量

xaverlin
8篇文章，2W+人气，0粉丝

提问和评论都可以，用心的回复会被更多人看到和认可

Ctrl+Enter 发布 取消 发布

推荐专栏

全局视角看大型园区网
路由交换+安全+无线+优化+运维
共40章 | 51CTO夏杰
订 阅
￥51.00 510人订阅

网工2.0晋级攻略 ——零基础入门Python/A…
网络工程师2.0进阶指南
共30章 | 姜汁啤酒
订 阅
￥51.00 946人订阅

基于Python的DevOps实战
自动化运维开发新概念
共20章 | 抚琴煮酒
订 阅
￥51.00 181人订阅

负载均衡高手炼成记
高并发架构之路
共15章 | sery
订 阅
￥51.00 299人订阅

带你玩转高可用
前百度高级工程师的架构高可用实战
共15章 | 曹林华
订 阅
￥51.00 323人订阅

猜你喜欢

整合Apache与SVN，让客户端可以用浏览器浏览源码库
SVN从win迁移到Linux上
Gitlab+Harbor+Jenkins pipeline实现利用tag部署docker…
线上zabbix监控redis和redis集群
部署Ceph集群 - - jluocc
Gitlab+Gerrit+Ldap+nginx+mysql 之Gerrit搭建与配置(一)
Jenkins部署与备份恢复实战
Kafka学习之路 （五）Kafka在zookeeper中的存储

在线客服

1 0 分享 xaverlin