XY

昵称： sunmmi
园龄： 8年6个月
粉丝： 14
关注： 21
关注成功

< 2023年8月 >

日	一	二	三	四	五	六
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

搜索
找找看
谷歌搜索

常用链接
我的随笔
我的评论
我的参与
最新评论
我的标签

随笔档案
2017年1月(2)
2016年8月(1)

文章分类
docker(43)
elk(17)
fastdfs(14)
git/svn(20)
haproxy(3)
更多

阅读排行榜
1. Linux系统性能优化思路和方法(5124)
2. web故障排查常用命令(666)

使用find和sed递归重命名文件

1.查找文件并重名文件，前面加字符

```
# 查找jar包，并把文件名yx替换成prod-yx

#方法1
find ./target/ -type f -name *.jar -exec bash -c 'mv $0 \${0/yx/prod-yx}' {} \;"


# 方法2
find . -name *.jar | sed  -e "p;s/yx/prod-yx/" | xargs -n2 mv


# 方法3，失败
find ./ -type f -name *.jar | sed 's#.*/##' |awk  '{print "prod-" $0}'
```

我想浏览一堆目录并将所有以_test.rb结尾的文件重命名为_spec.rb结尾。这是我从未想过如何处理bash的事情所以这次我认为我会付出一些努力来实现它。到目前为止，我做得很短，我最大的努力是：

```
find spec -name "*_test.rb" -exec echo mv {} `echo {} | sed s/test/spec/` \;
```

注意：在exec之后有一个额外的echo，这样在我测试时打印命令而不是运行。

当我运行它时，每个匹配文件名的输出是：

```
mv original original
```

即sed的替换已经丢失。有什么诀窍？

发生这种情况是因为 `sed` name__接收字符串 `{}` 作为输入，可以通过以下方式验证：

```
find . -exec echo `echo "{}" | sed 's/./foo/g'` \;
```

它以递归方式为目录中的每个文件打印 `foofoo` name__。这种行为的原因是Shell在扩展整个命令时会执行一次管道。

没有办法引用 `sed` name__管道， `find` name__将为每个文件执行它，因为 `find` name__不通过Shell执行命令，也没有管道或反引号的概念。 GNU findutils手册解释了如何通过将管道放在单独的Shell脚本中来执行类似的任务：

```
#!/bin/sh
echo "$1" | sed 's/_test.rb$/_spec.rb/'
```

（在一个命令中使用 `sh -c` 和大量引号可能会有一些不正常的方法，但我不打算尝试。）

要以最接近原始问题的方式解决它可能使用xargs“args per command line”选项：

```
find . -name *_test.rb | sed -e "p;s/test/spec/" | xargs -n2 mv
```

它以递归方式查找当前工作目录中的文件，回显原始文件名（ `p` ），然后回显修改后的名称（ `s/test/spec/` ），并将其全部反馈到 `mv` （ `xargs -n2` ）。请注意，在这种情况下，路径本身不应包含字符串 `test` 。

你可能想要考虑其他方式

```
for file in $(find . -name "*_test.rb")
do
  echo mv $file `echo $file | sed s/_test.rb$/_spec.rb/`
```

3. yum 源(262)

最新评论

1. Re:25-docker 不同宿主机容器互联

我在 daemon.json 中一加入 "cluster-store" ，然后重启 docker 的时候就会报错。。

--Chains朱朱

```
done
```

我发现这个更短

```
find . -name '*_test.rb' -exec bash -c 'echo mv $0 ${0/test.rb/spec.rb}' {} \;
```

...然后你可以在 **bash one-liner中实现批量重命名** 如下：

```
$ rename _test _spec **/*_test.rb
```

（`globstar` Shell选项将确保bash找到所有匹配的 `*_test.rb` 文件，无论它们嵌套在目录层次结构中有多深...使用 `help shopt` 查找如何设置选项）

如果你愿意，你可以在没有sed的情况下完成：

```
for i in `find -name '*_test.rb'` ; do mv $i ${i%%_test.rb}_spec.rb ; done
```

`${var%%suffix}` 从 `suffix` 的值中剥离 `var` 。

或者，使用sed来做：

```
for i in `find -name '*_test.rb'` ; do mv $i `echo $i | sed 's/test/spec/'` ; done
```

看一个例子：

```
$ tree
.
├── ab_testArb
├── a_test.rb
├── a_test.rb_test.rb
├── b_test.rb
├── c_test.hello
├── c_test.rb
└── mydir
    └── d_test.rb

$ while IFS= read -r file; do echo "mv $file ${file/_test.rb/_spec.rb}"; done < <(find -name "*_
mv ./b_test.rb ./b_spec.rb
mv ./mydir/d_test.rb ./mydir/d_spec.rb
mv ./a_test.rb ./a_spec.rb
mv ./c_test.rb ./c_spec.rb
```

当文件名中包含空格时，这对我有用。下面的示例递归地将所有.dar文件重命名为.Zip文件：

```
find . -name "*.dar" -exec bash -c 'mv "$0" "`echo \"$0\" | sed s/.dar/.Zip/`"' {} \;
```

注意： 上面的 `function` 可能需要 `GNU` 版本的 `sed` 和 `find` 来正确处理 `find printf` 和 `sed -z -e` 和 `:;recursive regex test;t` 调用。如果您无法使用这些功能，则可能会通过一些小的调整来复制功能。

这应该从头到尾完成您想要的一切，并且非常小心。我用 `fork` 做了 `sed` ，但是我也在练习一些 `sed` 递归分支技术，这就是为什么我在这里。我觉得这有点像在理发学校打折。这是工作流程：

```
rm -rf ${UNNECESSARY}

    我故意遗漏任何可能删除或破坏任何类型数据的函数调用。你提到./app可能是不需要的。删除它或事先将其移动到其他位置，

_mvnfind "${@}"

    声明其参数并调用worker函数。 ${sh_io}特别重要，因为它保存了函数的返回值。 ${sed_sep}紧随其后;这是一个任意

mv -n $1 $2

    整棵树从一开始就被移动了。它会省去很多头痛;相信我。您想要做的其余部分 - 重命名 - 只是文件系统元数据的问题。例

read -R SED <<HEREDOC

```

```
    我在这里找到所有sed的命令以节省逃避麻烦并将它们读入变量以供给下面的sed。说明如下。

find . -name ${OLD} -printf

    我们开始find进程。使用find，我们只搜索需要重命名的任何内容，因为我们已经使用函数的第一个命令执行了所有的放置m

%dir-depth :tab: 'mv '%path-to-${SRC}' '${sed_sep}'%path-again :null delimiter:'

    在find找到我们需要的文件之后，直接构建并打印出（most）我们处理重命名所需的命令。 %dir-depth添加到每行的开头

sort -general-numerical -zero-delimited

    我们根据%directory-depth对所有find的输出进行排序，以便最先处理与$ {SRC}关系最近的路径。这避免了将mving文

sed -ex :rcrs;srch|(save${sep}*til)${OLD}|\saved${SUBSTNEW}|;til ${OLD=0}

    我认为这是整个脚本中唯一的循环，它只循环遍历为每个字符串打印的第二个%Path，以防它包含多个可能需要替换的$ {OLD
    所以基本上sed在这里搜索$ {sed_sep}，然后，找到它，保存它和它遇到的所有字符，直到它找到$ {OLD}，然后用$ {NE
    这避免了必须解析整个字符串，并确保mv命令字符串的前半部分（当然需要包含$ {OLD}）确实包含它，并且后半部分被更改

sed -ex...-ex search|%dir_depth(save*)${sed_sep}|(only_saved)|out

    这里发出的两个-exec调用没有第二个fork。首先，正如我们所见，我们根据需要修改mv的-printf函数命令所提供的findi
```

这是一个很好的oneliner，可以解决这个问题。 Sed无法处理此权限，尤其是当xargs使用-n 2传递多个变量时.bash替换可以轻松地处理这个：

```
find ./spec -type f -name "*_test.rb" -print0 | xargs -0 -I {} sh -c 'export file={}; mv $file ${
```

添加-type -f会将移动操作限制为仅限文件，-print 0将处理路径中的空白空间。

删除“.txt.txt”扩展名如下 -

```
cd ~/practice

find . -name "*txt" -execdir sh -c 'mv "$0" `echo "$0" | sed -r 's/\.[[:alnum:]]+\.[[:alnum:]]+$/
```

如果你用+代替;为了在批处理模式下工作，上面的命令将只重命名第一个匹配的文件，而不是'find'重命名的整个文件匹配列表。

```
find . -name "*txt" -execdir sh -c 'mv "$0" `echo "$0" | sed -r 's/\.[[:alnum:]]+\.[[:alnum:]]+$/
```

https://www.it-swarm.dev/zh/bash/%E4%BD%BF%E7%94%A8find%E5%92%8Csed%E9%80%92%E5%BD%92%E9%87%8D%E5%91%BD%E5%90%8D%E6%96%87%E4%BB%B6/972000968/

分类: shell

支持 Markdown

自动补全

提交评论 退出 订阅评论 我的博客

[Ctrl+Enter快捷键提交]

编辑推荐：
· 服务端不回应客户端的 syn 握手，连接建立失败原因排查
· 超强的 Anchor Positioning 锚点定位
· [MAUI] 在 .NET MAUI 中实现可拖拽排序列表
· 使用 MediatR 实现 CQRS
· 记一次 .NET某报关系统 非托管泄露分析

阅读排行：
· 弹尽粮绝，会员救园：会员上线，命悬一线
· 使用C#创建安装Windows服务程序(干货)
· 记录一次内网渗透过程
· 【技术积累】Linux中的命令行【理论篇】【九】
· 服务端不回应客户端的syn握手，连接建立失败原因排查