## *# 终止后台所有相关进程* ps -ef | grep -E 'nsq|redis|manage.py' | awk '{print $2}' | xargs kill

## find ./ -type f -print |xargs chmod 644 find ./ -type d -print|xargs chmod 755

**action**

通过前面俩步找到了想要的文件or目录，下面可以对找到的东西进行处理了，可以显示 删除   
等操作，反正你可以想到的命令都可以使用的哦。

-print #将路径名写入到标准输出 这个是默认的

-fprint file # 输出写入到文件

-ls #显示长目录列表

-fls file #写到文件

-delete #删除文件

-exec command {} \; #执行command，{}指示匹配的文件名 注意{} 和 \;之间的空格，没有会报错滴！

-ok command {} \; #和exec 一样，只是在做操作前需要确认 输入y 回车确定，直接回车什么也不做

做一个测试输出查找到的文件名使用echo

find -name 't\*' -exec echo {} \;

修改查找到的文件权限为 700

find -name 't\*' -exec chmod 700 {} \;

删除下面俩条语句都可以可以：

find -name 't\*' -exec rm {} \;

find -name 't\*' -delete

**xargs**

处理查找到的文件的时候使用 exec 处理command 有多个目标文件就会有执行command多少次，使用xargs可以   
把找到的文件通过管道，当做参数一次处理

xargs [-prt] [-istring] [command [argument...]]

find -name 't\*' | xargs echo {} # 看看输出你就会发现和 exec 的不同

{} . ./Linux过滤器.md ./深入理解计算机系统总结.md ./disamy-mbr.md

-p 每次action前 请求确认 y 是 不输则否

-t 输出命令本身意思是：tell me what are you doing

-r 告诉xargs 如果没有输入参数，则不运行命令（xargs 默认运行一次）

-i insert ，当你需要在xargs里面多次使用参数的时候比如

find -name 't\*' | xargs -i echo {} {}

find -name 't\*' | xargs -i mv {} ~/backup/{}.old

find命令是比较常用的命令，用来在特定目录下查找具有某种特征的文件。

一：find命令格式如下：

find [-path......] -options [-print -exec -ok]

path:要查找的目录路径（“~”表示$home目录；“.”表示当前目录；“/”表示根目录）。

print：将结果输出到标准输出。

exec：对匹配的文件执行该参数所给出的shell命令，形式为：command {} \,（{}和\,之间有空格）。

ok：与exec作用相同，区别是在执行命令之前，都会给用户提示，让用户确认是否执行。

linux上进程有5种状态:

[html] view plain copy

1. 运行(正在运行或在运行队列中等待)

2. 中断(休眠中, 受阻, 在等待某个条件的形成或接受到信号)

3. 不可中断(收到信号不唤醒和不可运行, 进程必须等待直到有中断发生)

4. 僵死(进程已终止, 但进程描述符存在, 直到父进程调用wait4()系统调用后释放)

5. 停止(进程收到SIGSTOP, SIGSTP, SIGTIN, SIGTOU信号后停止运行)

ps工具标识进程的5种状态码:

[html] view plain copy

D 不可中断 uninterruptible sleep (usually IO)

R 运行 runnable (on run queue)

S 中断 sleeping

T 停止 traced or stopped

Z 僵死 a defunct (”zombie”) process

[**Linux查看某个进程的线程**](https://www.cnblogs.com/EasonJim/p/8098217.html)

线程是现代操作系统上进行并行执行的一个流行的编程方面的抽象概念。当一个程序内有多个线程被叉分出用以执行多个流时，这些线程就会在它们之间共享特定的资源（如，内存地址空间、打开的文件），以使叉分开销最小化，并避免大量高成本的IPC（进程间通信）通道。这些功能让线程在并发执行时成为一个高效的机制。

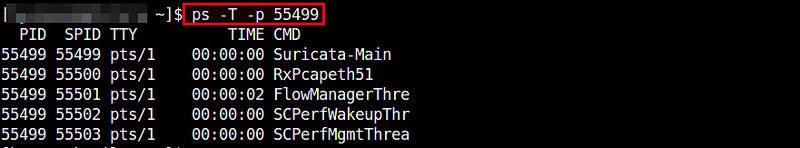
在Linux中，程序中创建的线程（也称为轻量级进程，LWP）会具有和程序的PID相同的“线程组ID”。然后，各个线程会获得其自身的线程ID（TID）。对于Linux内核调度器而言，线程不过是恰好共享特定资源的标准的进程而已。经典的命令行工具，如ps或top，都可以用来显示线程级别的信息，只是默认情况下它们显示进程级别的信息。

这里提供了在Linux上显示某个进程的线程的几种方式。

**方法一：PS**

在ps命令中，“-T”选项可以开启线程查看。下面的命令列出了由进程号为<pid>的进程创建的所有线程。

ps -T -p <pid>

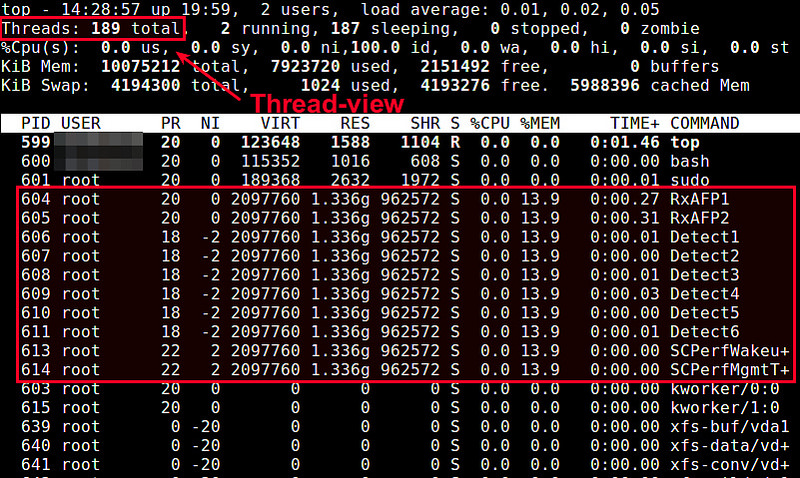
[](https://images2017.cnblogs.com/blog/417876/201712/417876-20171224154237818-2089734676.png)

“SID”栏表示线程ID，而“CMD”栏则显示了线程名称。

**方法二： Top**

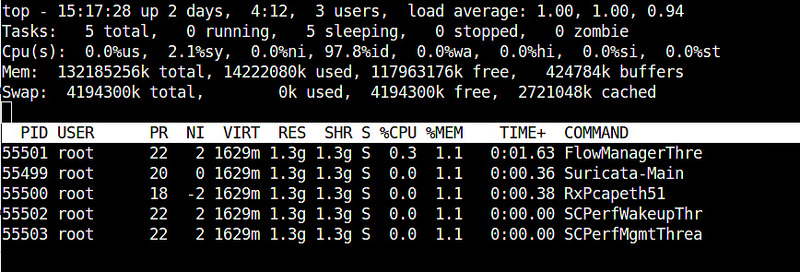
top命令可以实时显示各个线程情况。要在top输出中开启线程查看，请调用top命令的“-H”选项，该选项会列出所有Linux线程。在top运行时，你也可以通过按“H”键将线程查看模式切换为开或关。

top -H

[](https://images2017.cnblogs.com/blog/417876/201712/417876-20171224154304740-1853439090.png)

要让top输出某个特定进程<pid>并检查该进程内运行的线程状况：

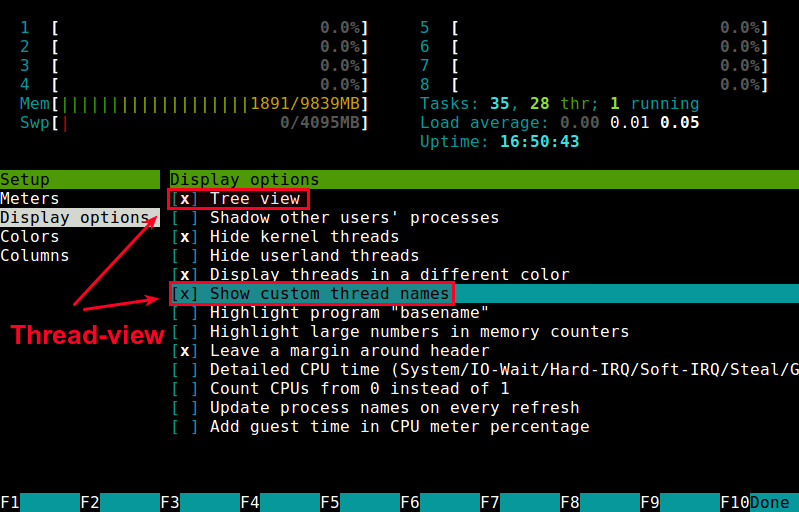
top -H -p <pid>

[](https://images2017.cnblogs.com/blog/417876/201712/417876-20171224154321475-969285560.png)

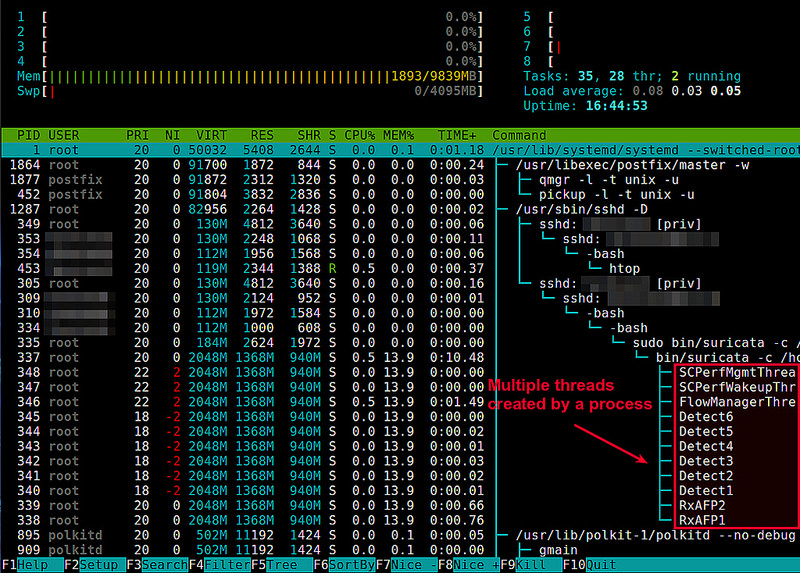
**方法三： Htop**

一个对用户更加友好的方式是，通过htop查看单个进程的线程，它是一个基于ncurses的交互进程查看器。该程序允许你在树状视图中监控单个独立线程。

要在htop中启用线程查看，请开启htop，然后按<F2>来进入htop的设置菜单。选择“设置”栏下面的“显示选项”，然后开启“树状视图”和“显示自定义线程名”选项。按<F10>退出设置。

[](https://images2017.cnblogs.com/blog/417876/201712/417876-20171224154341771-1447632241.png)

现在，你就会看到下面这样单个进程的线程视图。

[](https://images2017.cnblogs.com/blog/417876/201712/417876-20171224154352225-1958827666.png)

 Linux下使用xargs得到字符串作为参数进行输出、awk得到字符串作为参数进行输出

背景：

cat test.txt

1

3

4

实例：

1、xargs

cat test.txt | xargs -i echo {}

cat test.txt | xargs -I {} echo {}

大小写i的参数是不一样的效果，小写i默认参数为{}，且不能改变，而大写I参数是自定义的，可以是{}、a...等等。

2、awk

awk是变成语言，{}代码块中可以使用Shell进行编写，相当于动态语言。

cat test.txt | awk '{print $0}'

$n：当前记录的第n个字段，比如n为1表示第一个字段，n为2表示第二个字段。

$0：这个变量包含执行过程中当前行的文本内容。

1、将当前目录下包含jack串的文件中，jack字符串替换为tom

sed -i "s/jack/tom/g" `grep "jack" -rl ./`

2、将某个文件中的jack字符串替换为tom

sed -i "s/jack/tom/g" test.txt

命令： sed -i "s/oldstring/newstring/g" `grep "oldstring" -rl  path`

解释一下： -i 在文件中直接替换，而不是输出到终端

         "s/oldstring/newstring/g"  替换字符串的语法，后面的g表示替换所有的

          `grep "oldstring" -rl path` : 注意该命令两边的引号。

                 -r：在目录中递归查找

                -l: 输出找到包含oldstring 的文件名

其实就是一个sed 命令  ：  sed -i "s/oldstring/newstring/g" "包含oldstring的文件"

   而其中包含oldstring的文件是同过 grep "oldstring" -rl  path 这条命令找出来的

### [linux 批量替换文件内容及查找某目录下所有包含某字符串的文件（批量修改文件内容）](http://justcoding.iteye.com/blog/1948005)

**博客分类：**

* [Ubuntu / Mac / Github / Aptana / Nginx / Shell / Linux](http://justcoding.iteye.com/category/229050)

### 1. sed

**C代码  [收藏代码](javascript:void())**

1. grep -rl matchstring somedir/ | xargs sed -i 's/string1/string2/g'

例如：

对象：文件夹

**C代码  [收藏代码](javascript:void())**

1. grep -rl 'windows' ./path | xargs sed -i 's/windows/linux/g'

### 2. find

对象：文件

**C代码  [收藏代码](javascript:void())**

1. find -name 'test' | xargs perl -pi -e 's|windows|linux|g'

 这里使用了perl语言，使用-e加上一段代码，从而批量地将当前目录及所有子目录下的file.log文件中的string1替换成了string2; string支持正则表达式

### 3. awk

**C代码  [收藏代码](javascript:void())**

1. grep -i "windows" -r ./path | awk -F : '{print $1}' | sort | uniq | xargs sed -i 's/windows/linux/g'

 这里使用了shell命令，先查找出文件，再用awk分割（以：切分），再行替换！

注意：   grep可以使用正则，也可以使用\转义一些特殊字符，比如“等

sed -i 's/\"localhost\"/\"10.2.2.2\"/g' /home/my.conf

更多：

## 

## sed replace word / string syntax

The syntax is as follows:

**C代码  [收藏代码](javascript:void())**

1. sed -i 's/old-word/new-word/g' \*.txt

GNU sed command can edit files in place (makes backup if extension supplied) using the -i option. If you are using an old UNIX sed command version try the following syntax:

**C代码  [收藏代码](javascript:void())**

1. sed 's/old/new/g' input.txt > output.txt

You can use old sed syntax along with [bash for loop](http://www.cyberciti.biz/faq/bash-for-loop/):

**C代码  [收藏代码](javascript:void())**

1. #!/bin/bash
2. OLD="xyz"
3. NEW="abc"
4. DPATH="/home/you/foo/\*.txt"
5. BPATH="/home/you/bakup/foo"
6. TFILE="/tmp/out.tmp.$$"
7. [ ! -d $BPATH ] && mkdir -p $BPATH || :
8. **for** f in $DPATH
9. **do**
10. **if** [ -f $f -a -r $f ]; then
11. /bin/cp -f $f $BPATH
12. sed "s/$OLD/$NEW/g" "$f" > $TFILE && mv $TFILE "$f"
13. **else**
14. echo "Error: Cannot read $f"
15. fi
16. done
17. /bin/rm $TFILE
18. #从文件内容查找匹配指定字符串的行：
19. $ grep "被查找的字符串" 文件名
21. #从文件内容查找与正则表达式匹配的行：
22. $ grep –e “正则表达式” 文件名
24. #查找时不区分大小写：
25. $ grep –i "被查找的字符串" 文件名
27. #查找匹配的行数：
28. $ grep -c "被查找的字符串" 文件名
30. #从文件内容查找不匹配指定字符串的行：
31. $ grep –v "被查找的字符串" 文件名
33. #从根目录开始查找所有扩展名为.txt的文本文件，并找出包含"linux"的行
34. find . -type f -name "\*.txt" | xargs grep "linux"