grep 过滤的命令，默认是过滤包含字符串的。

1.作用

Linux系统中grep命令是一种强大的文本搜索工具，它能使用正则表达式搜索文本，并把匹 配的行打印出来。grep全称是Global Regular Expression Print，表示全局正则表达式版本，它的使用权限是所有用户。

2.格式

grep [options]

3.主要参数

[options]主要参数：

－c：只输出匹配行的计数。

－I：不区分大 小写(只适用于单字符)。

－h：查询多文件时不显示文件名。

－l：查询多文件时只输出包含匹配字符的文件名。

－n：显示匹配行及 行号。

－s：不显示不存在或无匹配文本的错误信息。

－v：显示不包含匹配文本的所有行。

pattern正则表达式主要参数：

\： 忽略正则表达式中特殊字符的原有含义。

^：匹配正则表达式的开始行。

$: 匹配正则表达式的结束行。

\<：从匹配正则表达 式的行开始。

\>：到匹配正则表达式的行结束。

[ ]：单个字符，如[A]即A符合要求 。

[ - ]：范围，如[A-Z]，即A、B、C一直到Z都符合要求 。

。：所有的单个字符。

\* ：有字符，长度可以为0。

4.grep命令使用简单实例

$ grep ‘test’ d\*

显示所有以d开头的文件中包含 test的行。

$ grep ‘test’ aa bb cc

显示在aa，bb，cc文件中匹配test的行。

$ grep ‘[a-z]\{5\}’ aa

显示所有包含每个字符串至少有5个连续小写字符的字符串的行。

$ grep ‘w\(es\)t.\*\1′ aa

如果west被匹配，则es就被存储到内存中，并标记为1，然后搜索任意个字符(.\*)，这些字符后面紧跟着 另外一个es(\1)，找到就显示该行。如果用egrep或grep -E，就不用”\”号进行转义，直接写成’w(es)t.\*\1′就可以了。

5.grep命令使用复杂实例

假设您正在’/usr/src/Linux/Doc’目录下搜索带字符 串’magic’的文件：

$ grep magic /usr/src/Linux/Doc/\*

sysrq.txt:\* How do I enable the magic SysRQ key?

sysrq.txt:\* How do I use the magic SysRQ key?

其中文件’sysrp.txt’包含该字符串，讨论的是 SysRQ 的功能。

默认情况下，’grep’只搜索当前目录。如果 此目录下有许多子目录，’grep’会以如下形式列出：

grep: sound: Is a directory

这可能会使’grep’ 的输出难于阅读。这里有两种解决的办法：

明确要求搜索子目录：grep -r

或忽略子目录：grep -d skip

如果有很多 输出时，您可以通过管道将其转到’less’上阅读：

$ grep magic /usr/src/Linux/Documentation/\* | less

这样，您就可以更方便地阅读。

有一点要注意，您必需提供一个文件过滤方式(搜索全部文件的话用 \*)。如果您忘了，’grep’会一直等着，直到该程序被中断。如果您遇到了这样的情况，按 <CTRL c> ，然后再试。

下面还有一些有意思的命令行参数：

grep -i pattern files ：不区分大小写地搜索。默认情况区分大小写，

grep -l pattern files ：只列出匹配的文件名，

grep -L pattern files ：列出不匹配的文件名，

grep -w pattern files ：只匹配整个单词，而不是字符串的一部分(如匹配’magic’，而不是’magical’)，

grep -C number pattern files ：匹配的上下文分别显示[number]行，

grep pattern1 | pattern2 files ：显示匹配 pattern1 或 pattern2 的行，

grep pattern1 files | grep pattern2 ：显示既匹配 pattern1 又匹配 pattern2 的行。

grep -n pattern files 即可显示行号信息

grep -c pattern files 即可查找总行数

这里还有些用于搜索的特殊符号：

\< 和 \> 分别标注单词的开始与结尾。

例如：

grep man \* 会匹配 ‘Batman’、’manic’、’man’等，

grep ‘\<man’ \* 匹配’manic’和’man’，但不是’Batman’，

grep ‘\<man\>’ 只匹配’man’，而不是’Batman’或’manic’等其他的字符串。

‘^’：指匹配的字符串在行首，

‘$’：指匹配的字符串在行 尾，

Grep 命令 用法大全

1、 参数：

-I ：忽略大小写

-c ：打印匹配的行数

-l ：从多个文件中查找包含匹配项

-v ：查找不包含匹配项的行

-n：打印包含匹配项的行和行标

2、RE（正则表达式）

\ 忽略正则表达式中特殊字符的原有含义

^ 匹配正则表达式的开始行

$ 匹配正则表达式的结束行

\< 从匹配正则表达式的行开始

\> 到匹配正则表达式的行结束

[ ] 单个字符；如[A] 即A符合要求

[ - ] 范围 ；如[A-Z]即A，B，C一直到Z都符合要求

. 所有的单个字符

\* 所有字符，长度可以为0

3、举例

# ps -ef | grep in.telnetd

root 19955 181 0 13:43:53 ? 0:00 in.telnetd

# more size.txt size文件的内容

b124230

b034325

a081016

m7187998

m7282064

a022021

a061048

m9324822

b103303

a013386

b044525

m8987131

B081016

M45678

B103303

BADc2345

# more size.txt | grep '[a-b]' 范围 ；如[A-Z]即A，B，C一直到Z都符合要求

b124230

b034325

a081016

a022021

a061048

b103303

a013386

b044525

# more size.txt | grep '[a-b]'\*

b124230

b034325

a081016

m7187998

m7282064

a022021

a061048

m9324822

b103303

a013386

b044525

m8987131

B081016

M45678

B103303

BADc2345

# more size.txt | grep 'b' 单个字符；如[A] 即A符合要求

b124230

b034325

b103303

b044525

# more size.txt | grep '[bB]'

b124230

b034325

b103303

b044525

B081016

B103303

BADc2345

# grep 'root' /etc/group

root::0:root

bin::2:root,bin,daemon

sys::3:root,bin,sys,adm

adm::4:root,adm,daemon

uucp::5:root,uucp

mail::6:root

tty::7:root,tty,adm

lp::8:root,lp,adm

nuucp::9:root,nuucp

daemon::12:root,daemon

# grep '^root' /etc/group 匹配正则表达式的开始行

root::0:root

# grep 'uucp' /etc/group

uucp::5:root,uucp

nuucp::9:root,nuucp

# grep '\<uucp' /etc/group

uucp::5:root,uucp

# grep 'root$' /etc/group 匹配正则表达式的结束行

root::0:root

mail::6:root

# more size.txt | grep -i 'b1..\*3' -i ：忽略大小写

b124230

b103303

B103303

# more size.txt | grep -iv 'b1..\*3' -v ：查找不包含匹配项的行

b034325

a081016

m7187998

m7282064

a022021

a061048

m9324822

a013386

b044525

m8987131

B081016

M45678

BADc2345

# more size.txt | grep -in 'b1..\*3'

1:b124230

9:b103303

15:B103303

# grep '$' /etc/init.d/nfs.server | wc -l

128

# grep '\$' /etc/init.d/nfs.server | wc –l 忽略正则表达式中特殊字符的原有含义

15

# grep '\$' /etc/init.d/nfs.server

case "$1" in

>/tmp/sharetab.$$

[ "x$fstype" != xnfs ] &&

echo "$path\t$res\t$fstype\t$opts\t$desc"

>>/tmp/sharetab.$$

/usr/bin/touch -r /etc/dfs/sharetab /tmp/sharetab.$$

/usr/bin/mv -f /tmp/sharetab.$$ /etc/dfs/sharetab

if [ -f /etc/dfs/dfstab ] && /usr/bin/egrep -v '^[ ]\*(#|$)'

if [ $startnfsd -eq 0 -a -f /etc/rmmount.conf ] &&

if [ $startnfsd -ne 0 ]; then

elif [ ! -n "$\_INIT\_RUN\_LEVEL" ]; then

while [ $wtime -gt 0 ]; do

wtime=`expr $wtime - 1`

if [ $wtime -eq 0 ]; then

echo "Usage: $0 { start | stop }"

# more size.txt

the test file

their are files

The end

# grep 'the' size.txt

the test file

their are files

# grep '\<the' size.txt

the test file

their are files

# grep 'the\>' size.txt

the test file

# grep '\<the\>' size.txt

the test file

# grep '\<[Tt]he\>' size.txt

the test file

==================================================================

1,简介

使用正则表达式的一个多用途文本搜索工具.这个php?name=%C3%FC%C1%EE" onclick="tagshow(event)" class="t\_tag">命令本来是ed行编辑器中的一个php?name=%C3%FC%C1%EE" onclick="tagshow(event)" class="t\_tag">命令/过滤器:

g/re/p -- global - regular expression - print.

基本格式

grep pattern [file...]

(1)grep 搜索字符串 [filename]

(2)grep 正则表达式 [filename]

在文件中搜索所有 pattern 出现的位置, pattern 既可以是要搜索的字符串,也可以是一个正则表达式.

注意：在输入要搜索的字符串时最好使用双引号/而在模式匹配使用正则表达式时，注意使用单引号

2,grep的选项

-c 只输出匹配行的计数

-i 不区分大小写（用于单字符）

-n 显示匹配的行号

-v 不显示不包含匹配文本的所以有行

-s 不显示错误信息

-E 使用扩展正则表达式

更多的选项请查看：man grep

3,常用grep实例

(1)多个文件查询

grep "sort" \*.doc #见文件名的匹配

(2)行匹配:输出匹配行的计数

grep -c "48" data.doc #输出文档中含有48字符的行数

(3)显示匹配行和行数

grep -n "48" data.doc #显示所有匹配48的行和行号

(4)显示非匹配的行

grep -vn "48" data.doc #输出所有不包含48的行

(4)显示非匹配的行

grep -vn "48" data.doc #输出所有不包含48的行

(5)大小写敏感

grep -i "ab" data.doc #输出所有含有ab或Ab的字符串的行

4, 正则表达式的应用

(1)正则表达式的应用 (注意：最好把正则表达式用单引号括起来)

grep '[239].' data.doc #输出所有含有以2,3或9开头的，并且是两个数字的行

(2)不匹配测试

grep '^[^48]' data.doc #不匹配行首是48的行

(3)使用扩展模式匹配

grep -E '219|216' data.doc

(4) ...

这需要在实践中不断应用和总结，熟练掌握正则表达式。

5, 使用类名

可以使用国际模式匹配的类名：

[[:upper:]] [A-Z]

[[:lower:]] [a-z]

[[:digit:]] [0-9]

[[:alnum:]] [0-9a-zA-Z]

[[:space:]] 空格或tab

[[:alpha:]] [a-zA-Z]

11、seq 打印数字序列

用于产生从某个数到另外一个数之间的所有整数

例一：

# seq 1 10

结果是1 2 3 4 5 6 7 8 9 10

例二：

#!/bin/bash

for i in `seq 1 10`;

do

echo $i;

done

或者用

for i in $(seq 1 10)

也可以

seq

-f, --format=FORMAT use printf style floating-point FORMAT (default: %g)

-s, --separator=STRING use STRING to separate numbers (default: \n)

-w, --equal-width equalize width by padding with leading zeroes

-f 选项 指定格式

#seq -f"%3g" 9 11

9

10

11

% 后面指定数字的位数 默认是"%g",

"%3g"那么数字位数不足部分是空格

#sed -f"%03g" 9 11 这样的话数字位数不足部分是0

% 前面制定字符串

seq -f "str%03g" 9 11

str009

str010

str011

-w 指定输出数字同宽 不能和-f一起用

seq -w -f"str%03g" 9 11

seq: format string may not be specified when printing equal width strings

seq -w 98 101

098

099

100

101

输出是同宽的

-s 指定分隔符 默认是回车

seq -s" " -f"str%03g" 9 11

str009 str010 str011

要指定\t 做为分隔符号

seq -s"`echo -e "\t"`" 9 11

指定\n\n作为分隔符号

seq -s"`echo -e "\n\n"`" 9 11

19293949596979899910911

得到的是个错误结果

不过一般也没有这个必要 它默认的就是回车作为分隔符

几个例子

awk 'BEGIN { while (num < 10 ) printf "dir%03d\n", ++num ; exit}' | xargs mkdir

mkdir $(seq -f 'dir%03g' 1 10)

for i in `seq -f '%02g' 1 20`

do

if ! wget -P $HOME/tmp -c [img]http://www.xxxsite.com/photo/$i.jpg[/img] ; then

wget -P $HOME/tmp -c $\_

fi

done

seq 是Linux 中一個預設的外部命令,一般用作一堆數字的簡化寫法，如

seq 1 10

便會出現

1

2

3

4

5

6

7

8

9

10

它還有三個選項

-f, --format=FORMAT use printf style floating-point FORMAT (default: %g)

-s, --separator=STRING use STRING to separate numbers (default: \n)

-w, --equal-width equalize width by padding with leading zeroes

-f 最常用, 例如一次制做10 個名dir001 , dir002 .. dir010 的目錄，它便很有用途,我們可以

這樣下一個命令便可了

seq -f 'dir%03g' 1 10 | xargs mkdir

或

mkdir $(seq -f 'dir%03g' 1 10)

它用的是printf 的格式, %03g' 代表以三位浮點數,以此方法，如用bash3 的printf

也可作為等價命令

printf 'dir%03d\n' {1..10} | xargs mkdir 或mkdir `printf 'dir%03d ' {1..10}`

awk 當然也可以

awk 'BEGIN { while (num < 10 ) printf "dir%03d\n", ++num ; exit}' | xargs mkdir

這樣會比寫一個腳本快, 不必寫成

for dir in 001 002 003 004 005 006 007 008 009 010

do

mkdir dir${dir}

done

我也常用seq 下載用數字的jpeg , 只要格式有數字順序便可,尤以一些xxx site

for i in `seq -f '%02g' 1 20`

do

if ! wget -P $HOME/tmp -c [img]http://www.xxxsite.com/photo/$i.jpg[/img] ; then

wget -P $HOME/tmp -c $\_

fi

done

-s 選項主要改變輸出的分格符, 預設是\n , 就是newline

如用-s 便可改變, 如

seq -s ' ' 1 10

1 2 3 4 5 6 7 8 9 10 , 以空格作為分格, 但在Gnu 的seq , 好像

不支援\n ,\t ...等字符？ 如用\n\n, 以兩個空格, 便得寫成

[victor@localhost ~]$ seq -s '

>

> ' 1 5

1

2

3

4

5

\t 便得改變IFS, 如用\t\t

OIFS=$IFS

IFS="\t\t"

seq -s `echo -e $IFS` 1 5

IFS=$OIFS

其它的字符也是這樣吧？

seq命令的作用就是打印出一串有序的数字,它主要有以下3个参数构成:

-f, --format=FORMAT

use printf style floating-point FORMAT (default: %g)

-f 指定打印的格式:

例如:

[root@hao32]# seq -f %05g 2 7

00002

00003

00004

00005

00006

00007

-s, --separator=STRING

use STRING to separate numbers (default: \n)

-s 指定分隔符 默认是回车:

例如:

[root@hao32]# seq -s" " 2 7

2 3 4 5 6 7

-w, --equal-width

equalize width by padding with leading zeroes

-w 输出是同宽 前面不足的用"0" 补全，即与位数最多的数对齐

例如:

[root@hao32]# seq -w 2 11

02

03

04

05

06

07

08

09

10

11