变量替换(可以用if实现 这是两外一种方式)

一 ${parameter:-word}

若 parameter 为空或未设置，则用 word 代替 parameter 进行替换，parameter 的值不变

# a=1

# unset b

# a=${b:-3}

# echo $a

# a=1

# b=2

# a=${b:-3}

# echo $a

# echo $b

二 ${parameter:=word}

若 parameter 为空或未设置，则 parameter 设为值 word

# a=1

# unset b

# a=${b:=3}

# echo $a

#echo $b

# a=1

# b=2

# a=${b:=3}

# echo $a

# echo $b

三 ${parameter:+word}

若 parameter 设置了，则用 word 代替 parameter 进行替换，parameter 的值不变

# a=1

# unset b

# a=${b:+3}

# echo $a

# echo $b

#

# a=1

# b=2

# a=${b:+3}

# echo $a

# echo $b

四 ${parameter:?message}

若 parameter 为空或未设置，则 message 作为标准错误打印出来，这可用来检查变量是否正确设置

# unset a

# ${a:?unset a}

-bash: a: unset a

$ a=12345678

$ echo ${a:5}

678

$ echo ${a:3:4}

4567

$ a=123456123789

$ echo ${a#1\*3} 最短匹配截取

456123789

$ echo ${a##1\*3} 最长匹配截取

789

$ a=123

$ echo ${#a} 表示$var的长度

3

$ a=123456123789

$ echo ${a/1/} 第一次匹配的被替换(去掉)

23456123789

$ echo ${a//1/} 全局的匹配被替换

2345623789

$ echo ${a/1/x}

x23456123789

$ echo ${a//1/x}

x23456x23789

**数组**

比如复制100个变量,这样你要起100个变量名称,数组的话使用1个名称就可以了,后变跟100个值,在访问值得时候使用下标的方式,如100个编号,编号即下标

定义一个数组

declare -a myarry=(5 6 7 8)

echo ${myarry[0]}

echo ${myarry[1]}

echo ${myarry[2]}

declare -a myarry=(5 666 7777 88888)

declare -p myarry

echo ${myarry[\*]}显示数组内容

echo ${myarry[@]}

echo ${#myarry[@]} #统计数组元素个数 #变量前行数

echo ${#myarry[2]} 统计某一个元素有多少个字符

**正则表达式**

算术表达式：

1+2 3\*5 1+2\*3 （1+2）\*3

正则表达式在匹配的时候一定要有一定规律,否则无法匹配 如下

特定的模式：

A B C … … -> a b c … …

ABC ADC AEC … …

ab abb abbb abbbb abbbb… …

正则表达式的匹配过程



如:

grep halt /etc/passwd

那我们通过grep来学习正则吧

grep命令是linux下的行过滤工具，其参数繁多，

grep -- print lines matching a pattern (将符合样式的该行列出)

◎语法: grep [options]   
PATTERN [FILE...]   
grep用以在file内文中比对相对应的部分，或是当没有指定档案时，   
由标准输入中去比对。 在预设的情况下，grep会将符合样式的那一行列出。  
其中egrep就等同于grep -E

1. -A NUM，--after-context=NUM   
                除了列出符合行之外，并且列出后NUM行。  
               
          ex:   $ **grep -A 1 panda file**   
                (从file中搜寻有panda样式的行，并显示该行的后1行)  
                                   
     2. -a或--text    
                grep原本是搜寻文字文件，若拿二进制的档案作为搜寻的目标，  
                则会显示如下的讯息: Binary file 二进制文件名 matches 然后结束。  
                    
                若加上-a参数则可将二进制档案视为文本文件搜寻，  
                相当于--binary-files=text这个参数。  
              
          ex:    (从二进制档案mv中去搜寻panda样式)  
                (错误!!!)  
               $ **grep panda mv**   
               Binary file mv matches    
                (这表示此档案有match之处，详见--binary-files=TYPE )  
               $  
                (正确!!!)  
               $ **grep -a panda mv**   
         
     3. -B NUM，--before-context=NUM  
                与 -A NUM 相对，但这此参数是显示除符合行之外  
                并显示在它之前的NUM行。          
               
          ex:    (从file中搜寻有panda样式的行，并显示该行的前1行)  
               $ **grep -B 1 panda file**   
  
     4. -C [NUM], -NUM, --context[=NUM]    
                列出符合行之外并列出上下各NUM行，默认值是2。  
               
          ex:    (列出file中除包含panda样式的行外并列出其上下2行)  
                (若要改变默认值，直接改变NUM即可)  
               $ **grep -C[NUM] panda file**   
               
     5. -b, --byte-offset  
                列出样式之前的内文总共有多少byte ..  
                
           ex:  $ **grep -b panda file**    
        显示结果类似于:  
          0:panda  
         66:pandahuang  
        123:panda03  
             
     6. --binary-files=TYPE  
                此参数TYPE预设为binary(二进制)，若以普通方式搜寻，只有2种结果:  
                  1.若有符合的地方：显示Binary file 二进制文件名 matches  
                  2.若没有符合的地方：什么都没有显示。  
                     
                若TYPE为without-match，遇到此参数，  
                grep会认为此二进制档案没有包含任何搜寻样式，与-I 参数相同。  
                     
                若TPYE为text, grep会将此二进制文件视为text档案，与-a 参数相同。  
          
     **Warning**: --binary-files=text 若输出为终端机，可能会产生一些不必要的输出。  
                
     7. -c, --count  
        不显示符合样式行，只显示符合的总行数。  
        若再加上-v,--invert-match，参数显示不符合的总行数。  
  
     8. -d ACTION, --directories=ACTION  
                若输入的档案是一个资料夹，使用ACTION去处理这个资料夹。  
        预设ACTION是read(读取)，也就是说此资料夹会被视为一般的档案；  
        若ACTION是skip(略过)，资料夹会被grep略过：  
        若ACTION是recurse(递归)，grep会去读取资料夹下所有的档案，  
        此相当于-r 参数。  
  
     9.  -E, --extended-regexp  
        采用规则表示式去解释样式。  
        
    10.  -e PATTERN, --regexp=PATTERN  
        把样式做为一个partern，通常用在避免partern用-开始。    
  
    11.  -f FILE, --file=FILE  
        事先将要搜寻的样式写入到一个档案，一行一个样式。  
        然后采用档案搜寻。  
        空的档案表示没有要搜寻的样式，因此也就不会有任何符合。  
         
    ex: (newfile为搜寻样式文件)  
       $**grep -f newfile file**      
  
    12.  -G, --basic-regexp  
        将样式视为基本的规则表示式解释。(此为预设)  
  
    13.  -H, --with-filename  
        在每个符合样式行前加上符合的文件名称，若有路径会显示路径。  
         
    ex: (在file与testfile中搜寻panda样式)     
       $**grep -H panda file ./testfile**  
                file:panda  
                 ./testfile:panda  
                 $  
       
    14.  -h, --no-filename    
                与-H参数相类似，但在输出时不显示路径。  
  
    15.  --help   
                产生简短的help讯息。  
  
    16.  -I  
                grep会强制认为此二进制档案没有包含任何搜寻样式，  
                与--binary-files=without-match参数相同。  
                     
            ex:  $ **grep -I panda mv**  
  
    17.  -i, --ignore-case        
                忽略大小写，包含要搜寻的样式及被搜寻的档案。  
                 
            ex:  $ **grep -i panda mv**  
                  
    18.  -L, --files-without-match   
                不显示平常一般的输出结果，反而显示出没有符合的文件名称。  
  
    19.  -l, --files-with-matches                 
                不显示平常一般的输出结果，只显示符合的文件名称。  
  
    20.  --mmap                 
                如果可能，使用mmap系统呼叫去读取输入，而不是预设的read系统呼叫。   
                在某些状况，--mmap 能产生较好的效能。 然而，--mmap   
                如果运作中档案缩短，或I/O 错误发生时，  
                可能造成未定义的行为(包含core dump)，。  
                 
    21.  -n, --line-number  
                在显示行前，标上行号。  
                 
             ex:  $ **grep -n panda file**    
                 显示结果相似于下:  
                 行号:符合行的内容  
  
    22.  -q, --quiet, --silent   
                不显示任何的一般输出。请参阅-s或--no-messages  
  
    23.  -r, --recursive  
        递归地，读取每个资料夹下的所有档案，此相当于 -d recsuse 参数。  
  
    24.  -s, --no-messages  
        不显示关于不存在或无法读取的错误讯息。  
       
小注: 不像GNU grep，传统的grep不符合POSIX.2协议，  
        因为缺乏-q参数，且他的-s 参数表现像GNU grep的 -q 参数。  
        Shell Script倾向将传统的grep移植，避开-q及-s参数，  
        且将输出限制到/dev/null。  
      
POSIX: 定义UNIX及UNIX-like系统需要提供的功能。               
      
    25.  -V, --version   
   显示出grep的版本号到标准错误。  
   当您在回报有关grep的bugs时，grep版本号是必须要包含在内的。  
  
    26.  -v, --invert-match  
   显示除搜寻样式行之外的全部。  
                     
    27.  -w, --word-regexp  
           将搜寻样式视为一个字去搜寻，完全符合该"字"的行才会被列出。  
  
    28.  -x, --line-regexp  
   将搜寻样式视为一行去搜寻，完全符合该"行"的行才会被列出。

常用参数

-A 匹配行的后多少行

-B 匹配行的前多少行

-c 统计匹配行

-I 忽略大小写匹配

-E 扩展正则

-n 显示匹配行行号

-x 显示完全匹配的行

-v 取反

如下文件 a.txt

我要找到含有ABC ADC AEC 的行

asdfasdf ABC aasdajsd

afdafjal ADC qqweqeqwe

qweqe AEC adfajf

grep A.C a.txt

这里的.称为元字符

元字符

. 匹配除换行符之外的任意单个字符，awk中可以匹配换行符

\* 匹配任意一个（包括零个）在它前面的字符

如下

a

ab

ac

abb

abc

abbc

abbbbbbbbbbbc

grep ab\* b.txt

grep ab\*c b.txt

grep a\* b.txt

.\*任意一个字符出现一次到多次

asadfsadfc

ac

afdsdfasdfasdfasdfc

aacc

aaaaaa1ccccccccc2aaaaaaaaaa3ccccccccc4

grep a.\*c c.txt

[...] 匹配方括号中的任意一个字符，^为否定匹配， -表示字符的范围

acc

a.c

abc

a2c

a6c

a7c

a8c

grep [6789]c a.txt

grep [1-9]c a.txt

grep [^1-9]c a.txt

1+2

1-2

1\*2

1/2

grep ‘[+-\*/]’ a.txt

^和$

root

aroot

roota

grep ^root a.txt

grep root$ a.txt

grep ^root$ a.txt

只包含3个字符?

最少包3个字符?

匹配空行?显示行号?

多个空格的空行?

grep "\t"是不行的，应该输入一个正在的制表符，方法为先按 CTRL+V，再按 Tab 键。

\ 转移字符

abc

a,c

grep a\,c a.txt

扩展元字符

+ 匹配前面的正则表达式的一次出现或多次出现

a

ab

abb

abbb

egrep ab+ a.txt

? 前边字符出现0或1次

y

yes

Y

Yes

egrep [Yy][es]? a.txt

| 替代方案

company

companies

egrep compan'[y|iess]' a.txt

egrep company’(y|iess)’a.txt

{n,m}匹配出现的n到m次数， {n}匹配出现n次。{n,}匹配至少出现n次，

大多数awk都不支持，用于POSIX egrep和POSIX awk

aaaa 4

aaaaa 5

aaaaaa 6

aaaaaaa 7

egrep a\{4,5\} a.txt

字符类

[Ww]hat

\.H[12345]

字符的范围

[a-z]

[0-9]

[Cc]hapter[1-9]

[-+\*/]

[0-1][0-9][-/][0-3][0-9][-/][0-9][0-9]

排除字符类

[^0-9]

7

重复出现的字符

5

10

50

100

500

1000

5000

[15]0\*

[15]00\*

字符的跨度

\* 与 \{n,m\}

电话号码的匹配

[0-9]\{3\}-[0-9]\{7,8\}

分组操作

compan(y|ies)

**sed和awk**

sed和awk用于处理文本,在linux中大量操作都涉及到文本,如系统日志,应用程序日志.批量处理时对配置文件做修改.操作文件时有些文件的语法不规范,如messages有空格分隔 :分隔 分号分隔 ,号分隔

如apache日志,分隔符不明显,而且分割时都是特殊字符,提取时我们可能只要ip 访问目标地址 访问时间 还有格式修改,sed主要处理不规范的格式改为规范.

sed是一个“非交互式的”面向字符流的编辑器，awk是一种负责模式匹配的程序设计语言，它的典型示例是将数据转换成格式化的报表。

**sed流媒体编辑器的使用**

练习文本:

John Daggett, 341 King Road, Plymouth MA

Alice Ford, 22 East Broadway, Richmond VA

Orville Thomas, 11345 Oak Bridge Road, Tulsa OK

Terry Kalkas, 402 Lans Road, Beaver Falls PA

Eric Adams, 20 Post Road, Sudbury MA

Hubert Sims, 328A Brook Road, Roanoke VA

Amy Wilde, 334 Bayshore Pkwy, Mountain View CA

Sal Carpenter, 73 6th Street, Boston MA

(名字 街道 城市 大州)

dos2unix 文件转换

unix2dos

在windows和linux之间文本格式互转的工具

linux \n

windows \n\r 换行

语法

sed ‘args’ file.txt file2.txt 但是官方文档称args为命令

sed ‘cmd’file.txt file.txt

替换:

sed ‘s/old/new/’ file1.txt 结果显示在屏幕上云文件不变

sed 's/MA/Massachusetts/' file1.txt

格式调整

sed 's/ MA/, Massachusetts/' file1.txt

多条处理

sed 's/ MA/, Massachusetts/ ; s/ PA/, Pennsylvania/' file1.txt

或者

sed -e 's/ MA/, Massachusetts/' -e 's/ PA/, Pennsylvania/' file1.txt

脚本支持

脚本：sedsrc

s/ MA/, Massachusetts/

s/ PA/, Pennsylvania/

s/ CA/, California/

s/ VA/, Virginia/

s/ OK/, Oklahoma/

sed -f sedsrc file1.txt

保存输出

sed -f sedsrc file1.txt > newfile.txt

阻止输入行自动显示

sed -n 's/MA/Massachusetts/p' file1.txt

awk的简单使用

语法

awk ‘{主输入循环}’file

输出每个人的名字

awk ‘{print $1}’file

第二列 默认分隔符(一个空格 多个空格 一个制表符 多个制表符) 所以,显示

awk ‘{print $2}’file

指定分隔符

awk –F, ‘{ print$1 }’file1.txt

//匹配功能如

匹配包含MA的行

awk ‘/MA/’file1.txt

还想要用户名字

awk ‘/MA/{print $1}’file1.txt

取root用户的pid

awk –F: ‘/root/{print $1 $3}’/etc/passwd

匹配字符串 $1~表示匹配第一个字段

awk –F: ‘$1~/root/{print $3 $1}’/etc/passwd

找到所有uid并求出和

正则表达式及字符处理

1．使用grep显示出/usr/share/dict/words文件中，例如显示出所有含有fish的行：

2．使用grep显示出/usr/share/dict/words文件中，输出任何包含fish的所有行，还要输出紧接着这行的上下各两行的内容：

3. 使用grep显示出/usr/share/dict/words文件中，来显示出在words文件中有多少行含有fish。

4．使用grep显示出/usr/share/dict/words文件中，显示出那些行含有fish，并将行号一块输出，看一看starfish在哪行

5．想列出/usr/share/dict/words中包含先有字母t然后有一个元音字母，之后是sh的单词，命令为

6．在/usr/share/dict/words文件中，创建可以符合abominable，abominate，anomie和atomize的正则表达式，但是不要选到别的单词

7．在/usr/share/dict/words文件中包含多少先有字母t然后有一个元音字母，之后是sh的单词，只输出数量

8．列出/usr/share/dict/words中刚好包含16个字母的单词

9．我们将要使用/usr/share/doc文件夹来完成我们的下几个任务

列出/usr/share/doc/bash-\* 文件夹中，所有包含单词expansion的文件