Sed 命令

sed

什么是sed

sed(意为流编辑器 , 源自英语 “ stream editor” 的缩写) 是 Unix 常见的命令行程序。 sed 用来把文档或字符串里面的文字经过一系列编辑命令转换为另一种格式输出。 sed 通常 用来匹配一个或多个正则表达式的文本进行处理。分号 (;) 可以用作分隔命令的指示符。尽管 sed 脚本固有的很多限制 , 一连串的 sed 指令加起来可以编程像 仓 库番、快打砖块、甚至俄罗斯方块等电脑游戏的复杂程序。

如何使用sed

语法模式

• 命令行模式
• 脚本模式

命令行模式

sed 流编辑器 针对行进行操作的

sed命令的原理

```
    读文件---一行一行读
    存入缓存空间
    匹配行---是---动作--继续读取
          ---不是------ 继续读取
    读到最后一行为止
    输出
```

sed命令的用法

- sed [-options] '[cmd]' filename
- sed [-options] '[哪些行][干什么]' filename

```
cmd
    操作定位（哪一行）
        1    第一行
        2,3    从第二行到第三行
        $    最后一行
        正则表达式
        /^root/    以root开头的行
        /bin$/    以bin结尾的行

        1. 十进制数字
        2. 正则表达式
        3. 逗号分隔符
        4. 组合方式
        5. 特殊方式


    函数（干什么）
        p    打印，输出到屏幕上
        d    删除
        s    替换                    sed '/^#/s/\/\*.*\*\///' file
        a    当前行的行后，添加一行    sed '1ahello word' file
        i    当前行的行前，添加一行    sed '1ihello word' file
参数options
    -n 不输出所有的行
    -i 直接修改目标文件
    -e 连接多个cmd

简单控制流

1. ! 命令取反
    例如:sed '/kevin/!d' file
    删除不包含字符串”kevin“的行
2. { } 组合多个命令
组合命令作为一个整体被执行,函数命令之间用” ;  “分隔,组合命令可以嵌套。
例如:sed '/kevin/{s/1/2/; /3/d}' file。

3. n 读取下一输入行,从下一条命令而非第一条命令开始操作
例如:sed '/kevin/{n; d}' file
删除带字符串”kevin“行的下一行
```

课堂练习

1. 下载mysqlbinlog.row文件
2. 打印第三行
3. 打印1到5行
4. 打印最后一行
5. 打印30到最后一行
6. 打印包含“BEGIN”的行
7. 打印包含“COMMIT”的行
8. 打印以“###”开头的行
9. 删除每一行的“### ”
10. 删除所有“/到/”
11. 将“DELETE FROM”替换为“insert into”
12. 将“INSERT INTO”替换为“delete from”
13. 将“SET”替换为“where”
14. 将“WHERE”替换为“set”
15. 将“@1”替换为“id”
16. 将“@2”替换为“name”

```
[root@client0 ~]# sed '3p' mysqlbinlog.row
[root@client0 ~]# sed -n '3p' mysqlbinlog.row
[root@client0 ~]# sed -n '1,5p' mysqlbinlog.row
[root@client0 ~]# sed -n '$p' mysqlbinlog.row
[root@client0 ~]# sed -n '30,$p' mysqlbinlog.row
[root@client0 ~]# sed -n '/BEGIN/p' mysqlbinlog.row
[root@client0 ~]# sed -n '/COMMIT/p' mysqlbinlog.row
[root@client0 ~]# sed '{s/### //;s@\/\*.*\*\/@@;s/DELETE FROM/insert into/;s/INSERT INTO/delete
from/;s/SET/where/;s/WHERE/set/;s/@1/id/;s/@2/name/}' mysqlbinlog.row
```

课堂练习

1. 将192.168.1.1替换成192.168.2.2
2. 将192.168.1.1替换成192.188.5.1
3. 将192.168.1.1替换成192.192.192.1
4. 将hello,babay中babay后面追加",mybabay"
5. 将hello,babay中hello后面追加",mybabay"
6. 将hello,babay替换为"hello1hello2hello3,babay"
7. 将/tmp/shadow的内容追加到/tmp/passwd中以root开头的行的后面
8. 将/tmp/passwd中以root开头的行和后面的2行写入/tmp/shadow

```
[root@client0 ~]# echo 192.168.1.1|sed 's/\(.*\)1.1/\12.2/'
192.168.2.2
[root@client0 ~]# echo 192.168.1.1|sed 's/\(.*\)168.1\(.*\)/\1188.5\2/'
192.188.5.1
[root@client0 ~]# echo 192.168.1.1|sed 's/\(.*\)168.1\(.*\)/\1\1\11/'
192.192.192.1


[root@client0 ~]# echo hello,babay|sed 's/babay/&,mybabay/'
hello,babay,mybabay
[root@client0 ~]# echo hello,babay|sed 's/hello/&,mybabay/'
hello,mybabay,babay
[root@client0 ~]# echo hello,babay|sed 's/hello/&1&2&3/'
hello1hello2hello3,babay

[root@client0 ~]# sed '/^root/r /tmp/shadow' /tmp/passwd
[root@client0 ~]# sed '/^root/,+2w /tmp/shadow' /tmp/passwd
```

课后习题

1. 将selinux设置成开机关闭状态。用sed完成
2. 设置当前用户的umask值永久生效为033 ~/.bashrc。用sed完成
3. 用脚本实现自动化搭建DNS服务器，并自动化配置解析，自动化测试。

awk

简介

awk是一个强大的文本分析工具，相对于grep的查找，sed的编辑，awk在其对数据分析并生成报告时，显得尤为强大。简单来说awk就是把文件逐行的读入，以空格为默认分隔符将每行切片，切开的部分再进行各种分析处理。

awk有3个不同版本: awk、nawk和gawk，未作特别说明，一般指gawk，gawk 是 AWK 的 GNU 版本。

awk其名称得自于它的创始人 Alfred Aho 、Peter Weinberger 和 Brian Kernighan 姓氏的首个字母。实际上 AWK 的确拥有自己的语言： AWK 程序设计语言 ， 三位创建者已将它正式定义为“样式扫描和处理语言”。它允许您创建简短的程序，这些程序读取输入文件、为数据排序、处理数据、对输入执行计算以及生成报表，还有无数其他的功能。

使用方法

awk '{pattern + action}' {filenames}

尽管操作可能会很复杂，但语法总是这样，其中 pattern 表示 AWK 在数据中查找的内容，而 action 是在找到匹配内容时所执行的一系列命令。花括号（{}）不需要在程序中始终出现，但它们用于根据特定的模式对一系列指令进行分组。 pattern就是要表示的正则表达式，用斜杠括起来。

awk语言的最基本功能是在文件或者字符串中基于指定规则浏览和抽取信息，awk抽取信息后，才能进行其他文本操作。完整的awk脚本通常用来格式化文本文件中的信息。

通常，awk是以文件的一行为处理单位的。awk每接收文件的一行，然后执行相应的命令，来处理文本。

调用awk

有三种方式调用awk

1. 命令行方式
   awk [-F field-separator] 'commands' input-file(s)

其中，commands 是真正awk命令，[-F域分隔符]是可选的。 input-file(s) 是待处理的文件。

在awk中，文件的每一行中，由域分隔符分开的每一项称为一个域。通常，在不指名-F域分隔符的情况下，默认的域分隔符是空格。

1. shell脚本方式

将所有的awk命令插入一个文件，并使awk程序可执行，然后awk命令解释器作为脚本的首行，一遍通过键入脚本名称来调用。

相当于shell脚本首行的：#!/bin/sh

可以换成：#!/bin/awk

1. 将所有的awk命令插入一个单独文件，然后调用：
   awk -f awk-script-file input-file(s)

其中，-f选项加载awk-script-file中的awk脚本，input-file(s)跟上面的是一样的。

本章重点介绍命令行方式。

入门实例

假设last -n 5的输出如下

```
[root@www ~]# last -n 5 <==仅取出前五行
root     pts/1   192.168.1.100  Tue Feb 10 11:21   still logged in
root     pts/1   192.168.1.100  Tue Feb 10 00:46 - 02:28  (01:41)
root     pts/1   192.168.1.100  Mon Feb  9 11:41 - 18:30  (06:48)
dmtsai   pts/1   192.168.1.100  Mon Feb  9 11:41 - 11:41  (00:00)
root     tty1                   Fri Sep  5 14:09 - 14:10  (00:01)
```

如果只是显示最近登录的5个帐号

```
#last -n 5 | awk  '{print $1}'
root
root
root
dmtsai
root
```

awk工作流程是这样的：读入有'\n'换行符分割的一条记录，然后将记录按指定的域分隔符划分域，填充域，0则表示所有域,1表示第一个域,n表示第n个域。默认域分隔符是”空白键”或”[tab]键”,所以1表示登录用户，$3表示登录用户ip,以此类推。

如果只是显示/etc/passwd的账户

```
#cat /etc/passwd |awk  -F ':'  '{print $1}'
root
daemon
bin
sys
```

这种是awk+action的示例，每行都会执行action{print $1}。

-F指定域分隔符为':'。

如果只是显示/etc/passwd的账户和账户对应的shell,而账户与shell之间以tab键分割

```
#cat /etc/passwd |awk  -F ':'  '{print $1"\t"$7}'
root    /bin/bash
daemon  /bin/sh
bin     /bin/sh
sys     /bin/sh
```

如果只是显示/etc/passwd的账户和账户对应的shell,而账户与shell之间以逗号分割,而且在所有行添加列名name,shell,在最后一行添加"blue,/bin/nosh"。

```
cat /etc/passwd |awk  -F ':'  'BEGIN {print "name,shell"}  {print $1","$7} END {print
"blue,/bin/nosh"}'
name,shell
root,/bin/bash
daemon,/bin/sh
bin,/bin/sh
sys,/bin/sh
....
blue,/bin/nosh
```

awk工作流程是这样的：先执行BEGING，然后读取文件，读入有/n换行符分割的一条记录，然后将记录按指定的域分隔符划分域，填充域，**0**则表示所有域,1表示第一个域,$n表示第n个域,随后开始执行模式所对应的动作action。接着开始读入第二条记录……直到所有的记录都读完，最后执行END操作。

搜索/etc/passwd有root关键字的所有行

```
#awk -F: '/root/' /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

这种是pattern的使用示例，匹配了pattern(这里是root)的行才会执行action(没有指定action，默认输出每行的内容)。

搜索支持正则，例如找root开头的: awk -F: '/^root/' /etc/passwd

搜索/etc/passwd有root关键字的所有行，并显示对应的shell

```
# awk -F: '/root/{print $7}' /etc/passwd
/bin/bash
```

这里指定了action{print $7}

awk内置变量

awk有许多内置变量用来设置环境信息，这些变量可以被改变，下面给出了最常用的一些变量。

```
ARGC            命令行参数个数
ARGV            命令行参数排列
ENVIRON         支持队列中系统环境变量的使用
FILENAME        awk浏览的文件名
FNR             浏览文件的记录数
FS              设置输入域分隔符，等价于命令行 -F选项
NF              浏览记录的域的个数
NR              已读的记录数
OFS             输出域分隔符
ORS             输出记录分隔符
RS              控制记录分隔符
```

此外,**0**变量是指整条记录。1表示当前行的第一个域,$2表示当前行的第二个域,.....以此类推。

统计/etc/passwd:文件名，每行的行号，每行的列数，对应的完整行内容:

```
#awk  -F ':'  '{print "filename:" FILENAME ",linenumber:" NR ",columns:" NF ",linecontent:"$0}'
/etc/passwd
filename:/etc/passwd,linenumber:1,columns:7,linecontent:root:x:0:0:root:/root:/bin/bash
filename:/etc/passwd,linenumber:2,columns:7,linecontent:daemon:x:1:1:daemon:/usr/sbin:/bin/sh
filename:/etc/passwd,linenumber:3,columns:7,linecontent:bin:x:2:2:bin:/bin:/bin/sh
filename:/etc/passwd,linenumber:4,columns:7,linecontent:sys:x:3:3:sys:/dev:/bin/sh
```

使用printf替代print,可以让代码更加简洁，易读

```
 awk  -F ':'
'{printf("filename:%10s,linenumber:%s,columns:%s,linecontent:%s\n",FILENAME,NR,NF,$0)}'
/etc/passwd
```

print和printf

awk中同时提供了print和printf两种打印输出的函数。

其中print函数的参数可以是变量、数值或者字符串。字符串必须用双引号引用，参数用逗号分隔。如果没有逗号，参数就串联在一起而无法区分。这里，逗号的作用与输出文件的分隔符的作用是一样的，只是后者是空格而已。

printf函数，其用法和c语言中printf基本相似,可以格式化字符串,输出复杂时，printf更加好用，代码更易懂。

awk编程

变量和赋值

除了awk的内置变量，awk还可以自定义变量。

下面统计/etc/passwd的账户人数

```
awk '{count++;print $0;} END{print "user count is ", count}' /etc/passwd
root:x:0:0:root:/root:/bin/bash
......
user count is  40
```

count是自定义变量。之前的action{}里都是只有一个print,其实print只是一个语句，而action{}可以有多个语句，以;号隔开。

这里没有初始化count，虽然默认是0，但是妥当的做法还是初始化为0:

```
awk 'BEGIN {count=0;print "[start]user count is ", count} {count=count+1;print $0;} END{print "
[end]user count is ", count}' /etc/passwd
[start]user count is  0
root:x:0:0:root:/root:/bin/bash
...
[end]user count is  40
```

统计某个文件夹下的文件占用的字节数

```
ls -l |awk 'BEGIN {size=0;} {size=size+$5;} END{print "[end]size is ", size}'
[end]size is  8657198
```

如果以M为单位显示:

```
ls -l |awk 'BEGIN {size=0;} {size=size+$5;} END{print "[end]size is ", size/1024/1024,"M"}'
[end]size is  8.25889 M
```

注意，统计不包括文件夹的子目录。

条件语句

awk中的条件语句是从C语言中借鉴来的，见如下声明方式：

```
if (expression) {
    statement;
    statement;
    ... ...
}

if (expression) {
    statement;
} else {
    statement2;
}

if (expression) {
    statement1;
} else if (expression1) {
    statement2;
} else {
    statement3;
}
```

统计某个文件夹下的文件占用的字节数,过滤4096大小的文件(一般都是文件夹):

```
ls -l |awk 'BEGIN {size=0;print "[start]size is ", size} {if($5!=4096){size=size+$5;}} END{print
"[end]size is ", size/1024/1024,"M"}'
[end]size is  8.22339 M
```

循环语句

awk中的循环语句同样借鉴于C语言，支持while、do/while、for、break、continue，这些关键字的语义和C语言中的语义完全相同。

数组

因为awk中数组的下标可以是数字和字母，数组的下标通常被称为关键字(key)。值和关键字都存储在内部的一张针对key/value应用hash的表格里。由于hash不是顺序存储，因此在显示数组内容时会发现，它们并不是按照你预料的顺序显示出来的。数组和变量一样，都是在使用时自动创建的，awk也同样会自动判断其存储的是数字还是字符串。一般而言，awk中的数组用来从记录中收集信息，可以用于计算总和、统计单词以及跟踪模板被匹配的次数等等。

显示/etc/passwd的账户

```
awk -F ':' 'BEGIN {count=0;} {name[count] = $1;count++;}; END{for (i = 0; i < NR; i++) print i,
name[i]}' /etc/passwd
0 root
1 daemon
2 bin
3 sys
4 sync
5 games
......
```

这里使用for循环遍历数组

awk编程的内容极多，这里只罗列简单常用的用法，更多请参考
http://www.gnu.org/software/gawk/manual/gawk.html