awk 用法

Shaoguang Cheng

May 29, 2015

1 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: [languages=shell] 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

复制代码 1. 命令行方式 awk [-F field-separator] 'commands' input-file(s) 其中，com-

mands 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。

2.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

3. 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。复制代码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′*rootrootrootdmtsairootawk* ″ 0 则表示所有域,1 ,n 表示第 n 个域。默认域分隔符是" 空白键" 或"[tab] 键", 所以 1 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′*rootdaemonbinsys awk* + *action actionprint*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""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"。

复制代码 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 复制代码 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 ,1 表示第一个域,*n* *n* , *action* · · · · · · *END*

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 actionprint7

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。

复制代码 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 控制记录分隔符复制代码此外,0 1 表示当前行的第一个域,2 ,.....

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/passwdfilename : /etc/passwd, linenumber : 1, columns : 7, linecontent : root : x : 0 : 0 : root : /root : /bin/bashfilename : /etc/passwd, linenumber : 2, columns : 7, linecontent : daemon : x : 1 : 1 : daemon : /usr/sbin : /bin/shfilename : /etc/passwd, linenumber : 3, columns : 7, linecontent : bin : x : 2 : 2 : bin : /bin : /bin/shfilename : /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:

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; $ENDprint"usercountis", count'/etc/passwdroot : x : 0 : 0 : root : /root : /bin/bash.....usercountis40count action print, print action ;

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; $ENDprint"[end]usercx : 0 : 0 : root : /root : /bin/bash...[end]usercountis40

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; $ENDprint"[end]sizeis", size'[end]sizeis8657198

如果以 M 为单位显示:

ls -l |awk 'BEGIN size=0; size=size+5; $ENDprint"[end]sizeis", size/1024/1024, "M"'[end]sizeis8.2588

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:

复制代码 if (expression) statement; statement; ... ...

if (expression) statement; else statement2;

if (expression) statement1; else if (expression1) statement2; else statement3; 复制代码

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; ENDprint "[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

复制代码 awk -F ':' 'BEGIN count=0; name[count] = 1; $count++;; END for(i = 0; i < NR; i++) print$