实验87 Linux命令：awk命令

87.1  实验目的

1． 学会linux命令awk的使用方法；

87.2  实验要求

1． 掌握linux命令awk基本用法。

87.3  实验原理

87.3.1 awk命令

awk是一种编程语言，用于在linux/unix下对文本和数据进行处理。数据可以来自标准输入(stdin)、一个或多个文件，或其它命令的输出。它支持用户自定义函数和动态正则表达式等先进功能，是linux/unix下的一个强大编程工具。它在命令行中使用，但更多是作为脚本来使用。awk有很多内建的功能，比如数组、函数等，这是它和C语言的相同之处，灵活性是awk最大的优势。

之所以叫awk是因为其取了三位创始人 Alfred Aho，Peter Weinberger, 和 Brian Kernighan 的Family Name的首字符。

87.3.2 awk命令格式和选项

* 语法形式

awk [options] 'script' var=value file(s)

或

awk [options] -f scriptfile var=value file(s)

* 常用命令选项

-F fs or --field-separator fs

指定输入文件折分隔符，fs是一个字符串或者是一个正则表达式，如-F:。

-v var=value or --asign var=value

赋值一个用户定义变量。

-f scripfile or --file scriptfile

从脚本文件中读取awk命令。

-mf nnn and -mr nnn

对nnn值设置内在限制，-mf选项限制分配给nnn的最大块数目；-mr选项限制记录的最大数目。这两个功能是Bell实验室版awk的扩展功能，在标准awk中不适用。

-W compact or --compat, -W traditional or --traditional

在兼容模式下运行awk。所以gawk的行为和标准的awk完全一样，所有的awk扩展都被忽略。

-W copyleft or --copyleft, -W copyright or --copyright

打印简短的版权信息。

-W help or --help, -W usage or --usage

打印全部awk选项和每个选项的简短说明。

-W lint or --lint

打印不能向传统unix平台移植的结构的警告。

-W lint-old or --lint-old

打印关于不能向传统unix平台移植的结构的警告。

-W posix

打开兼容模式。但有以下限制，不识别：/x、函数关键字、func、换码序列以及当fs是一个空格时，将新行作为一个域分隔符；操作符\*\*和\*\*=不能代替^和^=；fflush无效。

-W re-interval or --re-inerval

允许间隔正则表达式的使用，参考(grep中的Posix字符类)，如括号表达式[[:alpha:]]。

-W source program-text or --source program-text

使用program-text作为源代码，可与-f命令混用。

-W version or --version

打印bug报告信息的版本。

87.3.3 awk**模式和操作**

awk脚本是由模式和操作组成的。

* **模式**

**模式可以是以下任意一个：**

**/正则表达式/：使用通配符的扩展集。**

**关系表达式：使用运算符进行操作，可以是字符串或数字的比较测试。**

**模式匹配表达式：用运算符~（匹配）和~!（不匹配）。**

**BEGIN语句块、pattern语句块、END语句块：参见awk的工作原理。**

* **操作**

**操作由一个或多个命令、函数、表达式组成，之间由换行符或分号隔开，并位于大括号内，主要部分是：**

**变量或数组赋值**

**输出命令**

**内置函数**

**控制流语句**

* **awk脚本基本结构**

**awk 'BEGIN{ print "start" } pattern{ commands } END{ print "end" }' file**

一个awk脚本通常由：BEGIN语句块、能够使用模式匹配的通用语句块、END语句块3部分组成，这三个部分是可选的。任意一个部分都可以不出现在脚本中，脚本通常是被单引号或双引号中，例如：

awk 'BEGIN{ i=0 } { i++ } END{ print i }' filename

awk "BEGIN{ i=0 } { i++ } END{ print i }" filename

87.3.4 awk的工作原理

awk 'BEGIN{ commands } pattern{ commands } END{ commands }'

第一步：执行BEGIN{ commands }语句块中的语句。

第二步：从文件或标准输入(stdin)读取一行，然后执行pattern{ commands }语句块，它逐行扫描文件，从第一行到最后一行重复这个过程，直到文件全部被读取完毕。

第三步：当读至输入流末尾时，执行END{ commands }语句块。

BEGIN语句块在awk开始从输入流中读取行之前被执行，这是一个可选的语句块，比如变量初始化、打印输出表格的表头等语句通常可以写在BEGIN语句块中。

END语句块在awk从输入流中读取完所有的行之后即被执行，比如打印所有行的分析结果这类信息汇总都是在END语句块中完成，它也是一个可选语句块。

pattern语句块中的通用命令是最重要的部分，它也是可选的。如果没有提供pattern语句块，则默认执行{ print }，即打印每一个读取到的行，awk读取的每一行都会执行该语句块。

87.3.5 awk运算符

运算符 描述

= += -= \*= /= %= ^= \*\*= 赋值

?: C条件表达式

|| 逻辑或

&& 逻辑与

~ ~! 匹配正则表达式和不匹配正则表达式

< <= > >= != == 关系运算符

空格 连接

+ - 加，减

\* / % 乘，除与求余

+ - ! 一元加，减和逻辑非

^ \*\*\* 求幂

++ -- 增加或减少，作为前缀或后缀

$ 字段引用

in 数组成员

87.3.6 awk内建变量

变量 描述

\$n 当前记录的第n个字段，字段间由FS分隔

\$0 完整的输入记录

ARGC 命令行参数的数目

ARGIND 命令行中当前文件的位置(从0开始算)

ARGV 包含命令行参数的数组

CONVFMT 数字转换格式(默认值为%.6g)ENVIRON环境变量关联数组

ERRNO 最后一个系统错误的描述

FIELDWIDTHS 字段宽度列表(用空格键分隔)

FILENAME 当前文件名

FNR 各文件分别计数的行号

FS 字段分隔符(默认是任何空格)

IGNORECASE 如果为真，则进行忽略大小写的匹配

NF 输入字段分割符

NR 已经读出的记录数，就是行号，从1开始

OFMT 数字的输出格式(默认值是%.6g)

OFS 输出记录分隔符（输出换行符），输出时用指定的符号代替换行符

ORS 输出记录分隔符(默认值是一个换行符)

RLENGTH 由match函数所匹配的字符串的长度

RS 记录分隔符(默认是一个换行符)

RSTART 由match函数所匹配的字符串的第一个位置

SUBSEP 数组下标分隔符(默认值是/034)

87.4  实验步骤

### 87.4.1 创建实验集群

登录大数据实验平台，选择启动实验87。

87.4.2  编写带参数的 sed命令(篇幅有限,列举9个)

连接OpenVPN后，使用Xshell软件ssh登录master服务器。

* 87.4.2.1 预准备数据文件log.txt

[root@master ~]# vi log.txt

[root@master ~]# cat log.txt

2 this is a test

3 Are you like awk

This's a test

10 There are orange,apple,mongo

[root@master ~]#

* 87.4.2.2 awk '{[pattern] action}' {filenames}，行匹配语句 awk '' 只能用单引号

[root@master ~]# awk '{print $1,$4}' log.txt ### 每行按空格或TAB分割，输出文本中的1、4项

2 a

3 like

This's

10 orange,apple,mongo

[root@master ~]# awk '{printf "%-8s %-10s\n",$1,$4}' log.txt #### 格式化输出

2 a

3 like

This's

10 orange,apple,mongo

* 87.4.2.3 awk -F,-F相当于内置变量FS, 指定分割字符

[root@master ~]# awk -F, '{print $1,$2}' log.txt ### 使用","分割

2 this is a test

3 Are you like awk

This's a test

10 There are orange apple

[root@master ~]# awk 'BEGIN{FS=","} {print $1,$2}' log.txt ### 使用内建变量

2 this is a test

3 Are you like awk

This's a test

10 There are orange apple

[root@master ~]# awk -F '[ ,]' '{print $1,$2,$5}' log.txt ### 使用多个分隔符.先使用空格分割，然后对分割结果再使用","分割

2 this test

3 Are awk

This's a

10 There apple

[root@master ~]#

* 87.4.2.4 awk –v,-v表示设置变量

[root@master ~]# awk -va=1 '{print $1,$1+a}' log.txt

2 3

3 4

This's 1

10 11

[root@master ~]# awk -va=1 -vb=s '{print $1,$1+a,$1b}' log.txt

2 3 2s

3 4 3s

This's 1 This'ss

10 11 10s

[root@master ~]#

* 87.4.2.5 awk运算符(部分例子)

[root@master ~]# awk '$1> 2' log.txt ###过滤第一列大于2的行

3 Are you like awk

This's a test

10 There are orange,apple,mongo

[root@master ~]# awk '$1==2 {print $1,$3}' log.txt ###过滤第一列等于2的行

2 is

[root@master ~]# awk '$1>2 && $2=="Are" {print $1,$2,$3}' log.txt ###过滤第一列大于2并且第二列等于'Are'的行

3 Are you

[root@master ~]#

* 87.4.2.6 **awk内建变量(部分例子)**

[root@master~]#awk 'BEGIN{printf "%4s %4s %4s %4s %4s %4s %4s %4s %4s\n","FILENAME","ARGC","FNR","FS","NF","NR","OFS","ORS","RS";printf "---------------------------------------------\n"} {printf "%4s %4s %4s %4s %4s %4s %4s %4s %4s\n",FILENAME,ARGC,FNR,FS,NF,NR,OFS,ORS,RS}' log.txt

[root@master~]#awk -F\' 'BEGIN{printf "%4s %4s %4s %4s %4s %4s %4s %4s %4s\n","FILENAME","ARGC","FNR","FS","NF","NR","OFS","ORS","RS";printf "---------------------------------------------\n"} {printf "%4s %4s %4s %4s %4s %4s %4s %4s %4s\n",FILENAME,ARGC,FNR,FS,NF,NR,OFS,ORS,RS}' log.txt

[root@master ~]# awk '{print NR,FNR,$1,$2,$3}' log.txt ### 输出顺序号 NR, 匹配文本行号

1 1 2 this is

2 2 3 Are you

3 3 This's a test

4 4 10 There are

[root@master ~]# awk '{print $1,$2,$5}' OFS=" $ " log.txt ### 指定输出分割符

2 $ this $ test

3 $ Are $ awk

This's $ a $

10 $ There $

* 87.4.2.7 使用正则，字符串匹配

[root@master ~]# awk '$2 ~ /th/ {print $2,$4}' log.txt #### 输出第二列包含 "th"，并打印第二列与第四列(~ 表示模式开始，// 中是模式)

this a

[root@master ~]# awk '/re/ ' log.txt ### 输出包含"re" 的行

3 Are you like awk

10 There are orange,apple,mongo

[root@master ~]#

* 87.4.2.8 忽略大小写

[root@master ~]# awk 'BEGIN{IGNORECASE=1} /this/' log.txt

2 this is a test

This's a test

[root@master ~]#

* 87.4.2.9 模式取反

[root@master ~]# awk '$2 !~ /th/ {print $2,$4}' log.txt

Are like

a

There orange,apple,mongo

[root@master ~]# awk '!/th/ {print $2,$4}' log.txt

Are like

a

There orange,apple,mongo

[root@master ~]#

* 87.4.2.10 awk脚本

关于awk脚本，我们需要注意两个关键词BEGIN和END。

BEGIN{ 这里面放的是执行前的语句 }

END {这里面放的是处理完所有的行后要执行的语句 }

{这里面放的是处理每一行时要执行的语句}

首先使用vi命令编写文件score.txt，

[root@master ~]# vi score.txt

键入如下测试数据：

Marry 2143 78 84 77

Jack 2321 66 78 45

Tom 2122 48 77 71

Mike 2537 87 97 95

Bob 2415 40 57 62

键入完成后，保存退出。

然后再使用vi命令编写文件cal.awk，

[root@master ~]# vi cal.awk

键入如下代码：

#!/bin/awk -f

#运行前

BEGIN {

math = 0

english = 0

computer = 0

printf "NAME NO. MATH ENGLISH COMPUTER TOTAL\n"

printf "---------------------------------------------\n"

}

#运行中

{

math+=$3

english+=$4

computer+=$5

printf "%-6s %-6s %4d %8d %8d %8d\n", $1, $2, $3,$4,$5, $3+$4+$5

}

#运行后

END {

printf "---------------------------------------------\n"

printf " TOTAL:%10d %8d %8d \n", math, english, computer

printf "AVERAGE:%10.2f %8.2f %8.2f\n", math/NR, english/NR, computer/NR

}

键入完成后，保存退出。

执行代码,查看运行结果

[root@master ~]# awk -f cal.awk score.txt

NAME NO. MATH ENGLISH COMPUTER TOTAL

---------------------------------------------

Marry 2143 78 84 77 239

Jack 2321 66 78 45 189

Tom 2122 48 77 71 196

Mike 2537 87 97 95 279

Bob 2415 40 57 62 159

0 0 0 0

---------------------------------------------

TOTAL: 319 393 350

AVERAGE: 53.17 65.50 58.33

[root@master ~]#

87.5  实验结果

* 87.4.2.1运行结果如下：

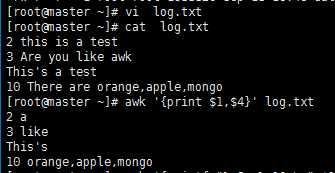


图87-1 运行结果

* 87.4.2.2运行结果如下：

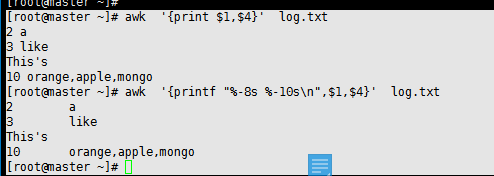


图87-2 运行结果

* 87.4.2.3运行结果如下：

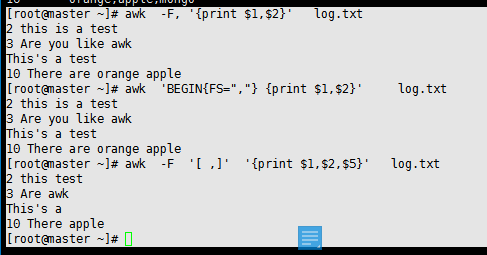


图87-3 运行结果

* 87.4.2.4运行结果如下：

![C:\Users\dongguangming\AppData\Roaming\Tencent\Users\1056764180\QQ\WinTemp\RichOle\O}H{R0J2TX$GVGBMC](6IXG.png](data:image/png;base64,)

图87-4 运行结果

* 87.4.2.5运行结果如下：

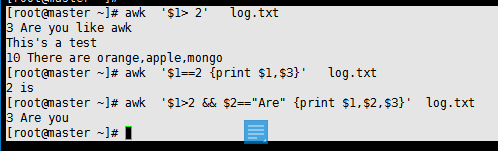


图87-5 运行结果

* 87.4.2.6运行结果如下：

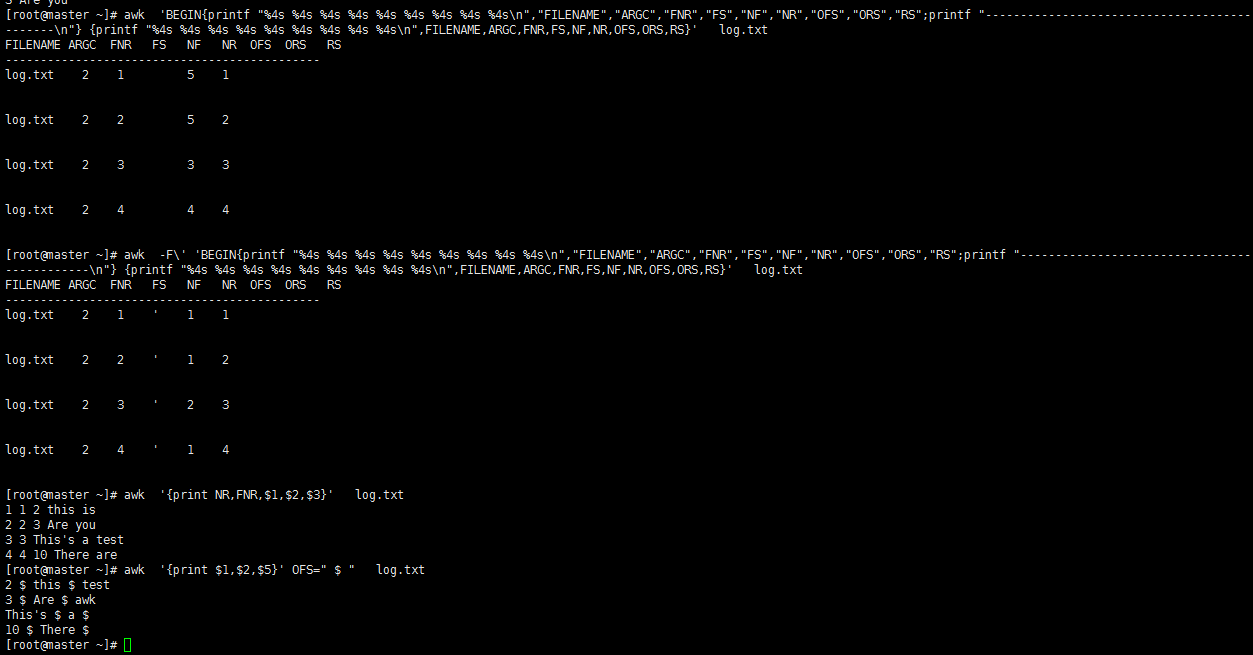


图87-6 运行结果

* 87.4.2.7运行结果如下：

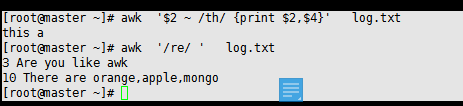


图87-7 运行结果

* 87.4.2.8运行结果如下：

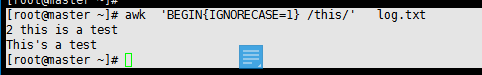


图87-8 运行结果

* 87.4.2.9运行结果如下：

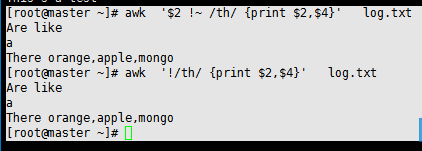


图87-9 运行结果

* 87.4.2.10运行结果如下：

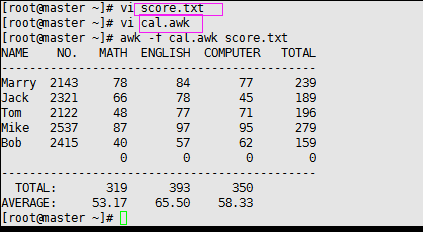


图87-10 运行结果