Linux 教程
Linux 教程
Linux 简介
Linux 安装
Linux 云服务器
Linux 系统启动过程
Linux 系统目录结构
Linux 忘记密码解决方法
Linux 远程登录
Linux 文件基本属性
Linux 文件与目录管理
Linux 用户和用户组管理
Linux 磁盘管理
Linux vi/vim
linux yum 命令
Linux apt 命令
Shell 教程
Shell 教程
Shell 变量
Shell 传递参数
Shell 数组
Shell 运算符
Shell echo命令
Shell printf命令
Shell test 命令
Shell 流程控制
Shell 函数
Shell 输入/输出重定向
Shell 文件包含
Linux 参考手册
Linux 命令大全
Nginx 安装配置
MySQL 安装配置

Shell 文件包含　　Nginx 安装配置

Linux awk 命令

Linux 命令大全

AWK 是一种处理文本文件的语言，是一个强大的文本分析工具。

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。

语法

```
awk [选项参数] 'script' var=value file(s)
或
awk [选项参数] -f scriptfile var=value file(s)
```

选项参数说明：

- -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。

基本用法

log.txt文本内容如下：

```
2 this is a test
3 Do you like awk
This's a test
10 There are orange,apple,mongo
```

用法一：

```
awk '{[pattern] action}' {filenames}   # 行匹配语句 awk '' 只能用单引号
```

实例：

```
# 每行按空格或TAB分割，输出文本中的1、4项
 $ awk '{print $1,$4}' log.txt
 ---------------------------------------------
 2 a
 3 like
 This's
 10 orange,apple,mongo
 # 格式化输出
 $ awk '{printf "%-8s %-10s\n",$1,$4}' log.txt
 ---------------------------------------------
 2        a
 3        like
 This's
 10       orange,apple,mongo
```

用法二：

```
awk -F  #-F相当于内置变量FS, 指定分割字符
```

实例：

```
# 使用","分割
 $  awk -F, '{print $1,$2}'   log.txt
 ---------------------------------------------
 2 this is a test
 3 Do you like awk
 This's a test
 10 There are orange apple
 # 或者使用内建变量
 $ awk 'BEGIN{FS=","} {print $1,$2}'     log.txt
 ---------------------------------------------
 2 this is a test
 3 Do you like awk
 This's a test
 10 There are orange apple
 # 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
 $ awk -F '[ ,]'  '{print $1,$2,$5}'   log.txt
 ---------------------------------------------
 2 this test
 3 Are awk
 This's a
 10 There apple
```

用法三：

```
awk -v  # 设置变量
```

实例：

```
 $ awk -va=1 '{print $1,$1+a}' log.txt
 ---------------------------------------------
 2 3
 3 4
 This's 1
 10 11
 $ 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
```

用法四：

```
awk -f {awk脚本} {文件名}
```

实例：

```
 $ awk -f cal.awk log.txt
```

运算符

运算符	描述
= += -= *= /= %= ^= **=	赋值
?:	C条件表达式
||	逻辑或
&&	逻辑与
~ 和 !~	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

过滤第一列大于2的行

```
$ awk '$1>2' log.txt    #命令
#输出
3 Do you like awk
This's a test
10 There are orange,apple,mongo
```

过滤第一列等于2的行

```
$ awk '$1==2 {print $1,$3}' log.txt    #命令
#输出
2 is
```

过滤第一列大于2并且第二列等于'Are'的行

```
$ awk '$1>2 && $2=="Are" {print $1,$2,$3}' log.txt    #命令
#输出
3 Are you
```

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	一条记录的字段的数目
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	输出字段分隔符，默认值与输入字段分隔符一致。
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)

```
$ awk 'BEGIN{printf "%4s %4s %4s %4s %4s %4s %4s %4s %4s\n","FILENAME","ARGC","FNR","FS","NF","NR","OFS",
FILENAME ARGC  FNR   FS   NF   NR  OFS  ORS   RS
---------------------------------------------
log.txt    2    1         5    1
log.txt    2    2         5    2
log.txt    2    3         3    3
log.txt    2    4         4    4
$ awk -F\' 'BEGIN{printf "%4s %4s %4s %4s %4s %4s %4s %4s %4s\n","FILENAME","ARGC","FNR","FS","NF","NR",
FILENAME ARGC  FNR   FS   NF   NR  OFS  ORS   RS
---------------------------------------------
log.txt    2    1    '    1    1
log.txt    2    2    '    1    2
log.txt    2    3    '    2    3
log.txt    2    4    '    1    4
# 输出顺序号 NR, 匹配文本行号
$ awk '{print NR,FNR,$1,$2,$3}' log.txt
---------------------------------------------
1 1 2 this is
2 2 3 Are you
3 3 This's a test
4 4 10 There are
# 指定输出分割符
$  awk '{print $1,$2,$5}' OFS=" $ "  log.txt
---------------------------------------------
2 $ this $ test
3 $ Are $ awk
This's $ a $
10 $ There $
```

使用正则，字符串匹配

```
# 输出第二列包含 "th"，并打印第二列与第四列
$ awk '$2 ~ /th/ {print $2,$4}' log.txt
---------------------------------------------
this a
```

~ 表示模式开始。// 中是模式。

```
# 输出包含 "re" 的行
$ awk '/re/ ' log.txt
---------------------------------------------
3 Do you like awk
10 There are orange,apple,mongo
```

忽略大小写

```
$ awk 'BEGIN{IGNORECASE=1} /this/' log.txt
---------------------------------------------
2 this is a test
This's a test
```

模式取反

```
$ awk '$2 !~ /th/ {print $2,$4}' log.txt
---------------------------------------------
Are like
a
There orange,apple,mongo
$ awk '!/th/ {print $2,$4}' log.txt
---------------------------------------------
Are like
a
There orange,apple,mongo
```

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}

假设有这么一个文件（学生成绩表）：

```
$ cat score.txt
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
```

我们的 awk 脚本如下：

```
$ cat cal.awk
#!/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
}
```

我们来看一下执行结果：

```
$ 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
---------------------------------------------
  TOTAL:       319      393      350
AVERAGE:     63.80    78.60    70.00
```

另外一些实例

AWK 的 hello world 程序为：

```
BEGIN { print "Hello, world!" }
```

计算文件大小

```
$ ls -l *.txt | awk '{sum+=$5} END {print sum}'
--------------------------------------------------
666581
```

从文件中找出长度大于 80 的行：

```
awk 'length>80' log.txt
```

打印九九乘法表

```
seq 9 | sed 'H;g' | awk -v RS='' '{for(i=1;i<=NF;i++)printf("%dx%d=%d%s", i, NR, i*NR, i==NR?"\n":"\t")}'
```

更多内容：

- *AWK 工作原理*
- *AWK 数组*
- *AWK 条件语句与循环*
- *AWK 用户自定义函数*
- *AWK 内置函数*
- *8 个有力的 Awk 内建变量*
- *AWK 官方手册*

Linux 命令大全

Shell 文件包含　　Nginx 安装配置

1 篇笔记　　写笔记

awk、sed、grep更适合的方向：

- grep 更适合单纯的查找或匹配文本
- sed 更适合编辑匹配到的文本
- awk 更适合格式化文本，对文本进行较复杂格式处理

47

关于awk内建变量个人见解，简单易懂

解释一下变量：

变量：分为内置变量和自定义变量;输入分隔符FS和输出分隔符OFS都属于内置变量。

内置变量就是awk预定义好的、内置在awk内部的变量，而自定义变量就是用户定义的变量。

- FS(Field Separator)：输入字段分隔符，默认为空白字符
- OFS(Out of Field Separator)：输出字段分隔符，默认为空白字符
- RS(Record Separator)：输入记录分隔符(输入换行符)，指定输入时的换行符
- ORS(Output Record Separate)：输出记录分隔符（输出换行符），输出时用指定符号代替换行符
- NF(Number for Field)：当前行的字段的个数(即当前行被分割成了几列)
- NR(Number of Record)：行号，当前处理的文本行的行号。
- FNR：各文件分别计数的行号
- ARGC：命令行参数的个数
- ARGV：数组，保存的是命令行所给定的各参数

自定义变量的方法

- 方法一：-v varname=value ，变量名区分字符大小写。
- 方法二：在program中直接定义。

z977690557 6年前 (2017-04-01)

分类导航
HTML / CSS
JavaScript
服务端
数据库
数据分析
移动端
XML 教程
ASP.NET
Web Service
开发工具
网站建设

反馈/建议