    简介：是一个优良的文本处理工具，Linux及Unix环境中现有的功能最强大的数据处理引擎之一。这种编程及数据操作语言的最大功能

          取决于一个人所拥有的知识。awk经过改进生成的新的版本nawk,gawk，现在默认linux系统下日常使用的是gawk，用命令可以查

          看正在应用的awk的来源（ls -l /bin/awk ）

    名称由来：三位作者名字的首字母，Aho, Weinberger, Kernighan

    格式：

        awk [options] ‘program’ var=value file…

        awk [options] - f programfile var=value file…

        awk [options] 'BEGIN{ action;… } pattern{ action;… } END{action;… }' file ...

     awk通常组成：

         BEGIN语句块、能够使用模式匹配的通用语句块、END语句块

         program通常是被单引号或双引号中

    常用选项

        -F:指定分割符，默认为空白分隔符

        -v:定义变量，每个变量前添加-v

    基本格式：awk [options] 'program' file…

       program组成：patter{action}两个部分组成

           patter:决定动作的何时触发和触发事件

           (BEGIN,END)

          action对数据进行处理，放在{}内指明

          （ print, printf）

    分割符、域和记录

    • awk执行时， 由分隔符分隔的字段（域）标记$1,$2..$n称为域标识。 $0为所有域，注意：和shell中变量$符含义不同

    • 文件的每一行称为记录

    • 省略action，则默认执行 print $0 的操作。

    工作原理

        第一步：执行BEGIN{action;… }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{action;… }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{action;…}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，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

       action动作

           print格式:print item1,item2

           (1)用逗号隔开，输入结果以空格分割

          （2）item的值可以是字符串（用双引号引起来）、变量、当前记录（行）的字段、数值、awk表达式

          （3）如果不写item默认打印$0

       awk 变量

           内置变量：

           FS:指定字段的分隔符，默认为空格

               awk -v FS=":" '{print $1}' file

               awk  -F  "："  '{print $1}' file

           OFS：输出字段的分隔符，默认的为空白

               awk - v FS=‘:’ - v OFS=‘:’ '{print $1,$3,$7}’ /etc/passwd

           RS：输入记录分隔符， 指定输入时的换行符，原换行符仍有效

            awk -v RS=' ' ‘{print }’ /etc/passwd

           NF：字段数量

               awk  -F： '{print NF}' /etc/passwd

             NR:行号（多个文件时，将不单独统计每个文件的行号）

               awk  '{print NR,$0}' /etc/passwd

           FNR:分别显示每个文件的行号

               awk    '{print FNR,$0}' /etc/passwd /etc/fstab

           FILENAME：当前文件名

               awk '{print FILENAME}’ /etc/fstab

            ARGV：数组，保存的是命令行所给定的各参数

                awk ‘BEGIN {print ARGV[0]}’ /etc/fstab

        自动变量：

            (1) - v var=value

              变量名区分字符大小写

            (2) 在program中直接定义

             awk - v test='hello gawk' '{print test}' /etc/fstab  ：/etc/fstab有几行将会打印出几行‘hellow gawk’

                         awk - v test='hello gawk' 'BEGIN{print test}'

                         awk 'BEGIN{test="hello,gawk";print test}'

        printf

          格式：printf "格式"，item1，item2.....

                       (1) 必须指定FORMAT

                       (2) 不会自动换行，需要显式给出换行控制符， \n

                       (3) FORMAT中需要分别为后面每个item指定格式符

          格式符：与item一一对应

              %c:显示字符的ASCII码

              %s:字符

              %d %i：数字十进制格式

              %f:浮点型

              %e %E:科学计数法表示

              %u:显示无符号整数

              %g %G：科学计数法表示或者浮点型表示

              %%：显示%本身

          修饰符：

             #[.#]：第一个数字控制显示的宽度；第二个#表示小数点后精度， %3.1f

             - : 左对齐（默认右对齐） %-15s

             +：显示数值的正负符号 %+d

          awk实例：

              awk -F: '{printf "username:%-15s \t  UID:%+4d\n",$1,$3}' /etc/passwd

        awk操作符

            算术操作符：

             x+y, x- y, x\*y, x/y, x^y, x%y

                    - x: 转换为负数

                    +x: 转换为数值

                    [root@mycentos6 ~]# awk -v a=10 -v b=10  'BEGIN{print a + b}'

                    20

                    [root@mycentos6 ~]#

                    字符串操作符：没有符号的操作符，字符串连接

                    [root@mycentos6 ~]# awk -v a=10 -v b=10  'BEGIN{print a  b}'

                    1010

                    [root@mycentos6 ~]#

       赋值操作符：

                    =, +=, - =, \*=, /=, %=, ^=

                    ++, --

                 比较操作符：

                    >, >=, <, <=, !=, ==

                 模式匹配符：

                    ~：左边是否和右边匹配包含

                    !~：是否不匹配

                 [root@mycentos6 ~]# cat /etc/passwd |awk '$0 !~ /root/' |wc -l

                    35

                [root@mycentos6 ~]# cat /etc/passwd |awk '$0 !~ /^root/' |wc -l

                    36

                [root@mycentos6 ~]#

                 逻辑操作符： &&， ||， !

                     示例：注意{}位置

                    • awk –F: '$3>=0 && $3<=1000 {print $1}' /etc/passwd

                    • awk - F: '$3 ==0 || $3>=1000 {print $1 }' /etc/passwd

                    • awk - F: ‘!($3==0) {print $1 }' /etc/passwd

                    • awk - F: '!($3>=500) {print $3}}' /etc/passwd

                条件表达示

                    selector?if- true- expression:if- false- expression

                    selector:为条件，如果为真执行？后面的语句，如果为假则执行：后面的语句

                    示例：

                    [root@mycentos6 ~]# awk -F: '{$3>=500?type="commonuser":type="systemuser"; \ :分行显示，可以直接写一行

                    > printf "%-10s \t %s\n ",$1,type}' /etc/passwd

    PATTERN

              作用：根据pattern条件，过滤匹配的行，再做处理

               (1)如果未指定： 空模式，匹配每一行

               (2) /regular expression/：仅处理能够模式匹配到的行，需要用 / /括起来

                    awk '/^UUID/{print $1}' /etc/fstab

                    awk '!/^UUID/{print $1}' /etc/fstab

              (3) relational expression: 关系表达式；结果有“真”有“假”；结果为“

                    真”才会被处理；

                   真：结果为非0值，非空字符串 ‘0,1’:结果为真

                   假：结果为空字符串

              示例：

                   [root@mycentos6 ~]# awk -F: '$NF=="/bin/bash"{print $1 ,$3}' /etc/passwd

                    root 0

                    mysql 27

                    user1 500

                    user2 501

                    user3 502

                   [root@mycentos6 ~]#

                   [root@mycentos6 ~]# seq 10 |awk 'i=!i'

                    1

                    3

                    5

                    7

                    9

                    [root@mycentos6 ~]#

             （4） line ranges： 行范围

                startline,endline： /pat1/,/pat2/ 不支持直接给出数字格式

                awk - F: '/^root/,/^nobody/{print $1}' /etc/passwd

                awk - F: '(NR>=2&&NR<=10){print $1}' /etc/passwd

              [root@mycentos6 ~]# awk -F: '/^root/,/^user2/{print $1}' /etc/passwd

              5) BEGIN/END模式

                BEGIN{}: 仅在开始处理文件中的文本之前执行一次 使用于打印标题等

                END{}：仅在文本处理完成之后执行一次