grep:,纯粹的文本查找工具

sed:Stream EDitor, 流编辑器，对文本逐行读入和处理编辑

awk: 将制定文本逐行读入文本但是不对文本处理而是先切片，自由的处理某个段

AWK a.k.a. Aho, Kernighan and Weinberger

new awk: nawk 一次扩展

linux上的awk叫gawk

# awk [options] 'script' file1, file2, ...

# awk [options] 'PATTERN { action }' file1, file2, ...

-F 指定字段分隔符 -F:以冒号为分隔符

################################

例如：

welcome to linux

how are you?

显示文件中的第一个字段：

awk ‘{print $1} a.txt 显示第一字段

df -h | awk ‘{print $1}’



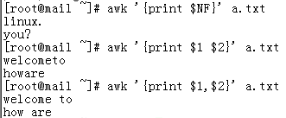
如果字段数不一样，例如都显示最后一段的话：引用特定段：内置变量

NF当前字段个数，

对于不同行的意义不一样。



逗号指定输出分隔符：



-v定义一个变量



FS字段分割符

输出的段分隔符：



这三个最常用：

OFS

FS

NF

#################################

awk的输出：

一、print

print的使用格式：

print item1, item2, ...

要点：

1、各项目之间使用逗号隔开，而输出时则以空白字符分隔；

2、输出的item可以为字符串或数值、当前记录的字段(如$1)、变量或awk的表达式；数值会先转换为字符串，而后再输出；

3、print命令后面的item可以省略，此时其功能相当于print $0, 因此，如果想输出空白行，则需要使用print ""；

例子：

# awk 'BEGIN { print "line one\nline two\nline three" }'

awk -F: '{ print $1, $2 }' /etc/passwd

内置变量：

ORS(output record separator)

OFS(output field separator)

FS: field separator，默认是空白字符；

RS: Record separator，默认是换行符；

NR: The number of input records，awk命令所处理的记录数；如果有多个文件，这个数目会把处理的多个文件中行统一计数；

NF：Number of Field，当前记录的field个数；

FNR: 与NR不同的是，FNR用于记录正处理的行是当前这一文件中被总共处理的行数；

ARGV: 数组，保存命令行本身这个字符串，如awk '{print $0}' a.txt b.txt这个命令中，ARGV[0]保存awk，ARGV[1]保存a.txt；

ARGC: awk命令的参数的个数；

二、printf

printf命令的使用格式：

printf format, item1, item2, ...

要点：

1、其与print命令的最大不同是，printf必须要指定format；

2、format用于指定后面的每个item的输出格式；

3、printf语句不会自动打印换行符；\n

format格式的指示符都以%开头，后跟一个字符；如下：

%c: 显示字符的ASCII码；

%d, %i：十进制整数；

%e, %E：科学计数法显示数值；

%f: 显示浮点数；

%g, %G: 以科学计数法的格式或浮点数的格式显示数值；

%s: 显示字符串；

%u: 无符号整数；

%%: 显示%自身；

修饰符：

N: 显示宽度；

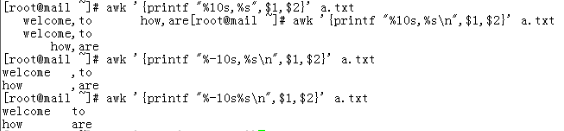
-: 左对齐；

+：显示数值符号；

##################

例子：

# awk -F: '{printf "%-15s %i\n",$1,$3}' /etc/passwd



##################

三、输出重定向

print items > output-file

print items >> output-file

print items | command

特殊文件描述符： 双引号引起来

/dev/stdin：标准输入

/dev/sdtout: 标准输出

/dev/stderr: 错误输出

/dev/fd/N: 某特定文件描述符，如/dev/stdin就相当于/dev/fd/0；

例子：

# awk -F: '{printf "%-15s %i\n",$1,$3 > "/dev/stderr" }' /etc/passwd

awk的操作符：

一、算术操作符：

-x: 负值

+x: 转换为数值；

x^y:

x\*\*y: 次方

x\*y: 乘法

x/y：除法

x+y:

x-y:

x%y:

二、字符串操作符：

只有一个，而且不用写出来，用于实现字符串连接；



三、赋值操作符：

=

+=

-=

\*=

/=

%=

^=

\*\*=

++

--

需要注意的是，如果某模式为=号，此时使用/=/可能会有语法错误，应以/[=]/替代；

四、布尔值

awk中，任何非0值或非空字符串都为真，反之就为假；

五、比较操作符：

x < y True if x is less than y.

x <= y True if x is less than or equal to y.

x > y True if x is greater than y.

x >= y True if x is greater than or equal to y.

x == y True if x is equal to y.

x != y True if x is not equal to y.

x ~ y True if the string x matches the regexp denoted by y.

x !~ y True if the string x does not match the regexp denoted by y.

subscript in array True if the array array has an element with the subscript subscript.

##################

例如：如果能被root匹配那么就显示第34和最后一个字段



如果不能被匹配就显示



##################

六、表达式间的逻辑关系符：

&&

||

七、条件表达式：

selector?if-true-exp:if-false-exp

八、函数调用：

function\_name (para1,para2)

awk的模式：

awk 'program' input-file1 input-file2 ...

其中的program为:

pattern { action }

pattern { action }

...

常见的模式类型：

1、Regexp: 正则表达式，格式为/regular expression/

2、expresssion： 表达式，其值非0或为非空字符时满足条件，如：$1 ~ /foo/ 或 $1 == "magedu"，用运算符~(匹配)和~!(不匹配)。

3、Ranges： 指定的匹配范围，格式为pat1,pat2

4、BEGIN/END：特殊模式，仅在awk命令执行前运行一次或结束前运行一次

5、Empty(空模式)：匹配任意输入行；

######################

如果某行包含bash行就显示



/bash/ 这就是模式 正则表达式



表达式：



######################

常见的Action有：

1、Expressions:

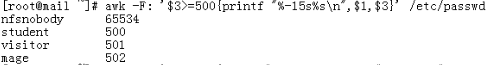
2、Control statements

3、Compound statements

4、Input statements

5、Output statements

#################



#awk -F: ‘$3>=500{print “user uid”;printf “%-15s%s\n”,$1,$3}’ /etc/passwd

只显示一次首部：begin：

awk -F: ‘BEGIN{print “user uid”;printf “%-15s%s\n”,$1,$3}’ /etc/passwd

end：

awk -F: ‘BEGIN{print “user uid”}{printf “%-15s%s\n”,$1,$3}END{print “over”}’ /etc/passwd



#################

/正则表达式/：使用通配符的扩展集。

关系表达式：可以用下面运算符表中的关系运算符进行操作，可以是字符串或数字的比较，如$2>%1选择第二个字段比第一个字段长的行。

模式匹配表达式：

模式，模式：指定一个行的范围。该语法不能包括BEGIN和END模式。

BEGIN：让用户指定在第一条输入记录被处理之前所发生的动作，通常可在这里设置全局变量。

END：让用户在最后一条输入记录被读取之后发生的动作。

控制语句：

1、if-else

语法：if (condition) {then-body} else {[ else-body ]}

例子：

awk -F: '{if ($1=="root") print $1, "Admin"; else print $1, "Common User"}' /etc/passwd

**awk -F: '{if ($1=="root") printf "%-15s: %s\n", $1,"Admin"; else printf "%-15s: %s\n", $1, "Common User"}' /etc/passwd**

awk -F: -v sum=0 '{if ($3>=500) sum++}END{print sum}' /etc/passwd

2、while

语法： while (condition){statement1; statment2; ...}

awk -F: '{i=1;while (i<=3) {print $i;i++}}' /etc/passwd

3、do-while

语法： do {statement1, statement2, ...} while (condition)

awk -F: '{i=1;do {print $i;i++}while(i<=3)}' /etc/passwd

4、for

语法： for ( variable assignment; condition; iteration process) { statement1, statement2, ...}

awk -F: '{for(i=1;i<=3;i++) print $i}' /etc/passwd

for循环还可以用来遍历数组元素：

语法： for (i in array) {statement1, statement2, ...}

awk -F: '$NF!~/^$/{BASH[$NF]++}END{for(A in BASH){printf "%15s:%i\n",A,BASH[A]}}' /etc/passwd

5、case

语法：switch (expression) { case VALUE or /REGEXP/: statement1, statement2,... default: statement1, ...}

6、break 和 continue

常用于循环或case语句中

7、next

提前结束对本行文本的处理，并接着处理下一行；

for i in {0..10}

print A[$i]

for (A in ARRAY) {print ARRAY[A]}

变量：0

S["m"]++,

S["m"]++

awk中使用数组：

array[index-expression]

index-expression可以使用任意字符串；需要注意的是，如果某数据组元素事先不存在，那么在引用其时，awk会自动创建此元素并初始化为空串；因此，要判断某数据组中是否存在某元素，需要使用index in array的方式。

要遍历数组中的每一个元素，需要使用如下的特殊结构：

for (var in array) { statement1, ... }

其中，var用于引用数组下标；

例子：

netstat -n | awk '/^tcp/ {++S[$NF]} END {for(a in S) print a, S[a]}'

每出现一被/^tcp/模式匹配到的行，数组S[$NF]就加1，NF为当前匹配到的行的最后一个字段，此处用其值做为数组S的元素索引；

awk '{counts[$1]++}; END {for(url in counts) print counts[url], url}' /var/log/httpd/access\_log

用法与上一个例子相同，用于统计某日志文件中IP地的访问量

awk的内置函数：

split(string, array [, fieldsep [, seps ] ])

功能：将string表示的字符串以fieldsep为分隔符进行分隔，并将分隔后的结果保存至array为名的数组中；

netstat -ant | awk '/:80/{split($5,clients,":");IP[clients[1]]++}END{for(i in IP){print IP[i],i}}' | sort -rn | head -50

length([string])

功能：返回string字符串中字符的个数；

substr(string, start [, length])

功能：取string字符串中的子串，从start开始，取length个；start从1开始计数；

system(command)

功能：执行系统command并将结果返回至awk命令

systime()

功能：取系统当前时间

PATTERN：

regexp

expression

pat1,pat2

BEGIN/END

empty

print item1,item2

printf format,iterm,iterm2,...

awk流程控制语句：if (condition) {} esle {};