**AWK学习笔记**

linux文本处理三剑客：

grep：文本抓取器，根据模式查找匹配文本的行，把行内容显示出来。

grep ‘pattern’ file1 file2......

sed：流编辑器

awk：报告生成器，从文本中抽取符合条件的信息，并以特定格式显示出来的工具。

# 1 awk基本使用格式

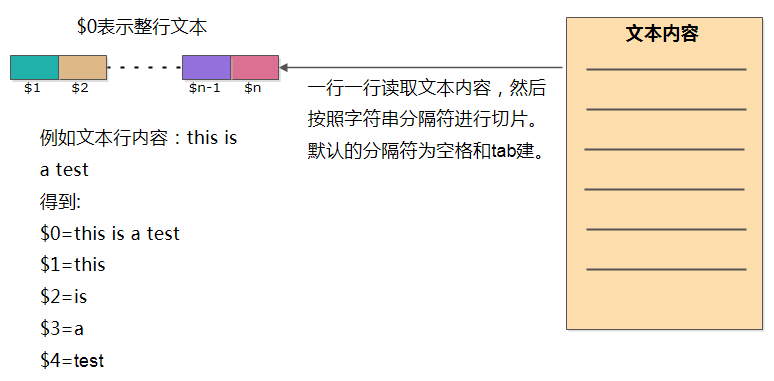
(1) awk [options] ‘script’ file1 file2......

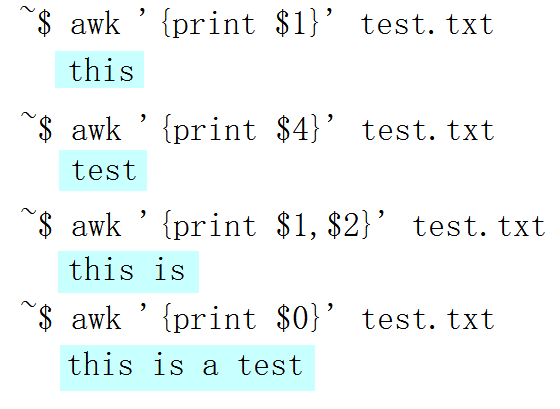
(2) awk [options] ‘pattern {ation}’ file1 file2......

格式2是格式1的细化，常用显示option为print printf

|  |
| --- |
| 例如简单用法：  awk '{print $0}' test.txt // 输出整行内容  awk '{print $1}' test.txt // 输出第一个值  awk '{print $1,$2}' test.txt |

# 2 awk原理





# 3 awk变量

## 3.1 awk内置记录变量

1. FS：file separator，输入字段分隔符，默认是空白字符
2. RS: record separator，输入行分隔符，一次读取一行文本信息的符号，默认是换行符。
3. OFS: output file separator，输出字段分割符，默认是空白字符。
4. ORS: output file separator，输出行分隔符，默认是换行符。

例如输入分隔符为冒号FS=”:”， 例如读取用户名和用户id语句：

awk 'BEGIN{FS=":"}{print $1,$3}' /etc/passwd

例如输出分隔符为加号OFS=”+”,

awk 'BEGIN{OFS="+"}{print $1,$2}' test.txt

## 3.2 awk内置数据变量

1. NR: the number of input records，awk处理的记录数，如果有多个文件，会把多个文件记录数累加。比如有两个文件，每个文件有10行内容，当独到第二个文件的第一行时NR=11。
2. FNR: 与NR有区别，FNR是各自文件的记录数。比如有两个文件，每个文件有10行内容，当独到第二个文件的第一行时FNR=1。
3. NF: number of field，当前记录字段的个数。

例如计算改行有多少个字段：awk '{print NF}' test.txt

例如输出最后一个字段内容awk '{print $NF}' test.txt

1. ARGV: 数组，保存命令行本身这个字符串。

例如语句awk ‘{print $0}’ a.txt b.txt中，ARGV[0]=awk，ARGV[1]=a.txt

1. ARGC: awk参数命令的个数。
2. FILENAME: awk命令所处理文件名称。
3. ENVIRON: 当前shell环境变量及其值的关联数组。

## 3.3 用户自定义变量

命名规则：用字母、数字、下划线组成，区分大小写，名称不能以数字开头。

1. 在脚本中使用赋值变量

例如：awk 'BEGIN{a="hello world"; print a}'

1. 在命令行中使用变量

例如：awk -v a="hello world" 'BEGIN{print a}'

# 4 print和printf的用法

## 4.1 print的用法

printf使用格式：print item1, item2, ......

**要点：**

1. 各项目之间如果用逗号隔开，则输出以空白分隔，

例如语句awk '{print $1,$2}' test.txt执行结果是this is

也可以自定义输出分割符，例如语句awk 'BEGIN{OFS="--"}{print $1,$2}' test.txt执行结果是this--is

1. 输出的item之间可以插入任意字符串。例如awk '{print "1:",$1,"2:",$2}' test.txt的结果是1: this 2: is
2. 输出的item可以为字符串或数字，当前记录的字段(例如$1)、变量、awk的表达式，数字会先转换为字符串，然后再输出。
3. print的后面的item可以省略，如果想输出空白行则执行语句awk 'BEGIN{print ""}'即可。

## 4.2 printf的用法

格式：printf format item1,item2,......

**要点：**

1. printf必须指定输出格式，而print不需要。

例如：awk '{printf "%s", $1}' test.txt

awk '{print $1}' test.txt

1. printf默认输出没有换行，如果想换行，加\n换行符即可，而print有换行。

例如：awk '{printf "%s\n", $1}' test.txt等价于awk '{print $1}' test.txt

1. printf用于指定每个item的输出格式，并且可以添加修饰符。

format格式指示符都以%开头，有下面几种格式符：

%c: 显示ASCII码字符

%s: 显示字符串

%d或%i: 显示十进制数

%u: 显示无符号整数

%f: 显示浮点数

%e或%E: 以科学计数法显示数值

%e或%E: 以科学计数法或浮点数格式显示数值

%%: 显示%本身

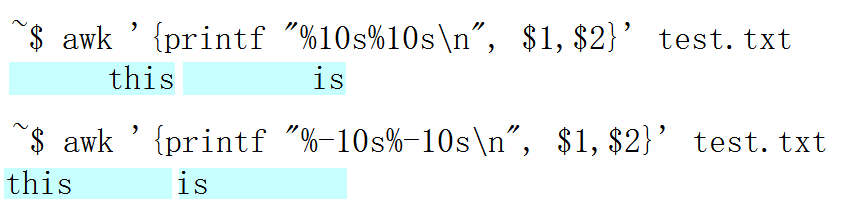
修饰符号有：

N: 显示宽度

-: 左对齐

+: 显示数值符号

例如：



# 5 运算符

1. 算数运算符：+ - \* / ^ % \*\*(次方)
2. 赋值运算符：+= -= \*= /= ^= %= \*\*= ++ --
3. 比较运算符：> < >= <= == != ~(匹配) !~

例如：awk 'BEGIN{a="10test";b="test";print a~b;}'的结果是1

1. 逻辑运算符：&& ||
2. 条件运算符(三目运算符)：真假表达式?结果1:结果2

例如取两个数的最大值：awk 'BEGIN{a=1;b=2;print a<b?b:a;}'

1. 字符串拼接操作符：”” 双引号里边可以有字符串

例如：awk 'BEGIN{a="hello";b="world";c=(a""b);print c}'的结果为helloworld

1. 字符串转数字：只需要将变量通过”+”连接运算。自动强制将字符串转为整型。非数字变成0，发现第一个非数字字符，后面自动忽略。

例如：awk 'BEGIN{a="10";b="10test";print (a+b);}'的结果为20

1. 数字转字符串：只需要将变量与””符号连接起来运算即可。

例如：awk 'BEGIN{a=10;b=10;print (a""b);}'的结果为1010

# 6 awk的控制流程

## 6.1 if-else条件

语法：if (condition){statements} else{[ statements]}

输出用户是管理员还是普通用户：

awk -F: '{if ($1=="root") {printf "%-10sAdmin\n", $1;}else {printf "%-10snormal\n",$1;}}' /etc/passwd

输出uid为大于0小于10的用户

awk -F: '{if ($3>0 && $3<10){printf "%-10s%s\n",$1,$3;}}' /etc/passwd

## 6.2 while和do-while循环

注意是循环一行中的每一个字段。

while语法：while(condition){statements}

do-while语法：do {statements} while(condition)

输出每行的前三个字段，用while和do-while实现

awk -F: '{i=1;while (i<=3){printf "%-10s",$i;i++;}print "";}' /etc/passwd

awk -F: '{i=1;do{printf "%-10s",$i;i++;}while(i<=3);print "";}' /etc/passwd

## 6.3 for循环

语法：for(var; condition; iteration){statements}

awk -F: '{for (i=1;i<=3;i++){printf "%-10s",$i;}print "";}' /etc/passwd

## 6.4 switch-case

语法：switch(expression){case VALUE or /REGEXP/:statements;...... default: statements;}

注意：不是所有awk都支持。

## 6.5 break continue next exit流程控制

break常用在循环或switch中

continue常用在循环体中

next：提前结束，进入下一行处理

输出uid为奇数的用户: awk -F: '{if ($3%2==0){next;}print $1,$3;}' /etc/passwd

# 7 awk的模式

awk的基本格式：awk [options] ‘pattern {ation}’ file1 file2......

其中pattern为模式，常见的模式有：

1. regexp: 正则表达式，格式为/regular expression/

例如找出用户名为r开头的用户awk -F: '/^r/{print $1,$3}' /etc/passwd

1. expression: 表达式，表达式的值为非0或非空字符时则满足条件，一般用比较运算符号的表达式~ > == ......

例如找出用户id大于500的用户awk -F: '$3>500{print $1,$3}' /etc/passwd

例如找出shell命令在bash目录的用户awk -F: '$7~"bash"{print $1,$7}' /etc/passwd

1. ranges: 指定匹配范围，格式为part1,part2，匹配符合pat1的行到符合pat2的行范围内的行,pat1和pat2是行号。

例如输出1到5行内容：awk -F: 'NR==1,NR==5{print $1,$3}' /etc/passwd

1. BEGIN和END: 在执行action之前或结束之后只执行一次。

例如在输出报表之前打印各列名称，输出完报表后提示结束。awk -F: 'BEGIN{print "NAME UID"}{printf "%-10s%-10s\n", $1,$3}END{print "\nreport end."}' /etc/passwd

# 8 数组

awk 中数组叫做关联数组(associative arrays)，下标可以是数字也可以是字符串。awk 中的数组不必提前声明，也不必声明大小，初始化数组元素用0 或空串，这根据上下文而定。

遍历数组语法：for(i in array){statements}

awk 'BEGIN{arr[1]="this";arr[2]="is";arr[3]="a";arr[4]="test";for(i in arr){print arr[i];}}'

例如统计各用户使用shell路径个数：awk -F: '{shell[$NF]++;}END{for(i in shell){print i,shell[i]}}' /etc/passwd

数组常用函数：

(1) length(array) 获取数组长度

(2) asort(array)对数组按照首字母进行排序，返回数组长度

(3) if(key in array)判断数组的key是否存在，存在返回true

(4) delete array[key]删除键值