BEGIN 在所有输入被读入前执行。awk只包含BEGIN块的话，可以不指定输入文件，如：

awk 'BEGIN{}'

END 在所有输入被读入后执行

-F 指定分隔符，如：awk -F”word” ‘…’ filename，以单词word为分隔符

FS也可指定分隔符，如：awk ‘BEGIN {FS=”:”} …’

测试匹配指定字段，如：

echo "11:21" | awk 'BEGIN {FS=":"} $2 ~ /21$/ {print $1}'

反匹配：

echo "11:31" | awk 'BEGIN {FS=":"} $2 !~ /21$/ {print $1}'

FS=”\t” 每个tab都是分隔符

FS=”\t+” 一个或多个tab构成分隔符

awk区分变量的大小写，x与X是不同的变量

**每个变量有两个值，一个字符串类型，一个数字类型**

OFS设置输出分隔符，如：

echo "11:31" | awk 'BEGIN {FS=":"; OFS="@"} $2 !~ /21$/ {print $1, $2}'

输出11@31

NF 当前行的字段数

NR 当前行号，**从1开始，不是0**

RS/ORS 输入/出的行分隔符，默认是\n

printf 没有自动换行，需要加\n

print有自动换行

printf 支持格式化打印，类似C语言：

echo "11:31" | awk 'BEGIN {FS=":"; OFS="@"} $2 !~ /21$/ {printf("%s, %s\n", $1, $2)}'

输出11, 31

**向awk传命令行参数：**

awk的变量可以定义在’…’中，也可定义在’…’后。所谓命令行参数就是定义在’…’块后的参数，在BEGIN块中不可用。

POSIX awk加入参数-v来指定能在BEGIN块中使用的变量：

awk -v x="X" -v y="Y" 'BEGIN{print x; print y}' #需要传入多个变量的话，每个变量对应一个-v

awk -f scriptfile “high=100” “low=10” datafile #awk要执行的代码下载scriptfile中，传入命令行参数high和low

echo "11:31" | awk '{print dir; print low; print high}' "dir=`pwd`" "high=100" "low=6" #传入参数dir,high和low。其中，dir是当前目录。

也可以传入shell变量：

echo "11:31" | awk '{print dir}' "dir=$cwd" #其中dir为shell变量

**awk的语句块：**

awk的条件判断和循环语句都要加{}，如：

echo "" | awk '{n=1; if (n != 0) {print "No-zero"} else {print "Zero"}}'

awk 'BEGIN{n=5; for (i=n; i >= 1; --i) {print i}}'

**awk的数组：**

awk的数组其实是hash map，下标是key，元素是value，数组遍历是无序的。

awk 'BEGIN{array["k1"]="v1"; array["k2"]="v2"; for (key in array) {print key, array[key]}}'

判断下标是否在数组中：

if (key in array)

如：

awk 'BEGIN{array["k1"]="v1"; array["k2"]="v2"; if (!("k3" in array)) {print "Invalid key"}}'

或：

awk 'BEGIN{array["k1"]="v1"; array["k2"]="v2"; if (("k3" in array)); else {print "Invalid key"}}' #**awk支持空语句，所以if后没有语句，只有一个”;”，然后跟else块**

**awk自定义函数：**

awk的自定义函数，**函数参数列表中定义的是局部变量，函数体内定义的是全局变量**。

echo "" | awk 'function add(x, y, sum) {sum=x+y; return sum} {n=add(2,5); print n}' #其中x,y,sum都是局部变量

echo "" | awk 'function add(x, y) {sum=x+y} {add(2,5); print sum}' #x,y是局部变量, sum是全局变量

awk调用函数传参，**普通变量是传值，数组是传引用**。

awk调用函数传参，允许传入的参数数量小于参数列表，未传的参数被当作0或空字符串，如：

function add(x, y) {sum=x+y}

调用add(2)，相当于add(2, 0)

**awk中的字符串拼接：**

a=”a”; b=”b”;

c=a””b; #c=”ab”

c=a”test”b; #c=”atestb”

**awk的输出可以重定向到管道**：

创建文件datafile，内容为：

4 2

3 1

awk '{print $1, $2 | "sort -k 2"}' datafile

输出为：

3 1

4 2

**注意：sort只会执行一次，而非每次print都会执行，也就是说sort是对所有的输出结果做排序。**

system函数可执行shell命令：

awk 'BEGIN{if (system("mkdir dir") != 0) print "failed"}'

getline函数可以从文件逐行读入：

awk 'BEGIN{while (getline line < "datafile") print line}'

从管道读入：

awk 'BEGIN{while ("ls -l" | getline line) print line}'

或：

awk 'BEGIN{while ("ls -l" | getline) print $3}' #打印ls -l第3列

**awk删除指定列：**

删除第5列: awk ' {$5=null; print $0}'

**awk调用shell命令：**

**使用所以system**

awk程序中可以使用system() 函数去调用shell命令，如:

awk 'BEGIN{system("echo abc")}'

awk 'BEGIN{v1="echo";v2="abc";system(v1" "v2)}'

如果system括号里面的参数没有加上双引号的话，awk认为它是一个变量，它会从awk的变量里面把它们先置换为常量，然后再回传给shell;

如果system括号里面的参数有加上双引号的话，那么awk就直接把引号里面的内容回传给shell，作为shell的“命令行”。

**无论使用system（）还是print cmd | “/bin/bash”  
awk都是新开一个shell，在相应的cmdline参数送回给shell，所以要注意当前shell变量与新开shell变量问题：**

abc=12345567890

awk 'BEGIN{system("echo $abc")}'

输出为空

export abc=12345567890

awk 'BEGIN{system("echo $abc")}'

输出12345567890

abc=12345567890

awk 'BEGIN{print "echo","$abc"| "/bin/bash"}'

输出为空

export abc=12345567890

awk 'BEGIN{print "echo","$abc"| "/bin/bash"}'

输出12345567890

实战例子：

awk -F'\t' '{printf("%s:", $1);create\_time=$2" "$3;system("date -d \"" create\_time "\" +%s")}' origin\_create\_time.txt

origin\_create\_time.txt(位于bjzyx-c3099.zqy:/data/project/imv\_music\_classification\_build)格式：

124 2016-07-09 18:24:40

126 2016-07-09 18:25:48

130 2016-07-13 16:25:29

132 2016-07-13 16:27:02

136 2016-07-19 19:40:07

142 2016-07-27 21:02:17

144 2016-07-28 20:41:26

148 2016-08-01 19:57:02

150 2016-08-02 21:24:22

152 2016-08-03 20:56:41

转后输出为：

124:1468059880

126:1468059948

130:1468398329

132:1468398422

136:1468928407

142:1469624537

144:1469709686

148:1470052622

150:1470144262

152:1470229001