# AWK 快速参考

模式、字段、数组、函数、文本处理

## 基础

### 运行 AWK

```
awk '{ print }' file.txt          # print every line
awk '{ print $1 }' file.txt       # print first field
awk -F: '{ print $1 }' /etc/passwd  # custom delimiter
awk -f script.awk file.txt        # run from file
cmd | awk '{ print $2 }'          # pipe input
```

### 程序结构

| | |
|---|---|
| **awk 'pattern { action }'** | 基本格式——模式匹配时执行 action |
| **BEGIN { ... }** | 在处理输入之前运行一次 |
| **END { ... }** | 处理完所有输入后运行一次 |
| **无模式** | 对每一行都执行 action |
| **无 action** | 默认 action 为 **{ print }** |

## 模式与动作

### 模式类型

```
awk '/error/' file.txt            # regex match
awk '$3 > 100' file.txt           # comparison
awk 'NR >= 5 && NR <= 10' file.txt  # line range
awk '/start/,/end/' file.txt      # range pattern
```

### 模式参考

| | |
|---|---|
| **/regex/** | 将行与正则表达式匹配 |
| **$1 ~ /pat/** | 字段匹配正则 |
| **$1 !~ /pat/** | 字段不匹配正则 |
| **expr1, expr2** | 范围：从第一个匹配到第二个 |
| **expr1 && expr2** | 逻辑与 |
| **expr1 \|\| expr2** | 逻辑或 |
| **!expr** | 逻辑非 |

## 变量

### 内置变量

| | |
|---|---|
| **NR** | 当前记录（行）编号 |
| **NF** | 当前记录的字段数 |
| **FS** | 输入字段分隔符（默认：空白） |
| **OFS** | 输出字段分隔符（默认：空格） |
| **RS** | 输入记录分隔符（默认：换行） |
| **ORS** | 输出记录分隔符（默认：换行） |
| **FILENAME** | 当前输入文件名 |
| **FNR** | 当前文件中的记录编号 |

### 用户变量

```
awk '{ total += $1 } END { print total }' file.txt
awk -v threshold=50 '$1 > threshold' file.txt
awk 'BEGIN { count = 0 } /pat/ { count++ }
     END { print count }' file.txt
```

## 字段

### 字段访问

| | |
|---|---|
| **$0** | 整行内容 |
| **$1, $2, ...** | 第一、第二……字段 |
| **$NF** | 最后一个字段 |
| **$(NF-1)** | 倒数第二个字段 |

### 字段分隔符

```
awk -F, '{ print $2 }' data.csv     # comma
awk -F'\t' '{ print $1 }' data.tsv  # tab
awk 'BEGIN { FS = "[,:]" } { print $1 }' f  # multi-char
awk 'BEGIN { OFS = "," } { print $1, $3 }' f  # output sep
```

## 控制流

### 条件与循环

```
awk '{ if ($1 > 50) print "high"; else print "low" }' f
awk '{ for (i = 1; i <= NF; i++) print $i }' f
awk '{ i = 1; while (i <= NF) { print $i; i++ } }' f
awk '/skip/ { next } { print }' f  # skip matching lines
```

### 控制语句

| | |
|---|---|
| **if (cond) { ... } else { ... }** | 条件判断 |
| **for (i = 0; i < n; i++) { ... }** | C 风格 for 循环 |
| **for (key in array) { ... }** | 遍历数组键 |
| **while (cond) { ... }** | while 循环 |
| **do { ... } while (cond)** | do-while 循环 |
| **next** | 跳到下一条输入记录 |
| **exit** | 停止处理，执行 END 块 |

## 函数

### 用户定义函数

```
awk 'function max(a, b) {
    return (a > b) ? a : b
}
{ print max($1, $2) }' file.txt
```

### 数值函数

| | |
|---|---|
| **int(x)** | 截断为整数 |
| **sqrt(x)** | 平方根 |
| **sin(x), cos(x)** | 三角函数 |
| **log(x), exp(x)** | 自然对数与指数 |
| **rand()** | 0 到 1 之间的随机浮点数 |
| **srand(seed)** | 设置随机数种子 |

## 数组

### 关联数组

```
awk '{ count[$1]++ }
     END { for (k in count) print k, count[k] }' f
awk '{ arr[NR] = $0 }
     END { for (i = NR; i >= 1; i--) print arr[i] }' f
```

### 数组操作

| | |
|---|---|
| **arr[key] = val** | 设置元素 |
| **arr[key]** | 获取元素（访问时自动创建） |
| **key in arr** | 判断键是否存在 |
| **delete arr[key]** | 删除单个元素 |
| **delete arr** | 删除整个数组 |
| **for (k in arr)** | 遍历键（无序） |
| **length(arr)** | 元素数量（gawk） |

## 字符串函数

### 字符串参考

| | |
|---|---|
| **length(s)** | 字符串长度 |
| **substr(s, start, len)** | 子串（从 1 开始计数） |
| **index(s, target)** | target 在 s 中的位置（未找到返回 0） |
| **split(s, arr, sep)** | 将字符串拆分到数组 |
| **sub(pat, repl, s)** | 替换第一个匹配 |
| **gsub(pat, repl, s)** | 替换所有匹配 |
| **match(s, pat)** | 正则匹配位置（设置 RSTART、RLENGTH） |
| **tolower(s) / toupper(s)** | 大小写转换 |
| **sprintf(fmt, ...)** | 格式化字符串（类似 C printf） |

### 字符串示例

```
awk '{ gsub(/old/, "new"); print }' f    # sed-like replace
awk '{ print toupper($0) }' f             # uppercase all
awk '{ print substr($0, 1, 40) }' f       # truncate to 40
```

## I/O

### 输出

```
awk '{ print $1, $2 }' f              # space-separated
awk '{ printf "%s,%d\n", $1, $2 }' f  # formatted output
awk '{ print $1 > "out.txt" }' f      # redirect to file
awk '{ print $1 >> "out.txt" }' f     # append to file
```

### I/O 参考

| | |
|---|---|
| **print** | 带 ORS 输出（默认换行） |
| **printf fmt, ...** | 格式化输出（不自动换行） |
| **print > file** | 重定向到文件 |
| **print >> file** | 追加到文件 |
| **print \| cmd** | 管道输出到命令 |
| **getline < file** | 从文件读取一行 |
| **cmd \| getline var** | 将命令输出读入变量 |
| **close(file)** | 关闭文件或管道 |

## 常用模式

### 单行命令

```
awk '{ sum += $1 } END { print sum }' f   # sum column
awk 'END { print NR }' f                   # count lines
awk '!seen[$0]++' f                        # remove dupes
awk 'NF' f                                 # remove blanks
awk '{ print NF }' f                       # fields per line
```

### 实用配方

| | |
|---|---|
| **CSV 转 TSV** | **awk -F, 'BEGIN{OFS="\t"} {$1=$1; print}'** |
| **第 2 列求和** | **awk '{ s += $2 } END { print s }'** |
| **前 N 行** | **awk 'NR <= 10'**（类似 head） |
| **频次统计** | **awk '{ c[$1]++ } END { for (k in c) print k, c[k] }'** |
| **区间提取** | **awk '/BEGIN/,/END/'** |
| **打印第 N 列** | **awk '{ print $N }'**（N 替换为列号） |