Referência Rápida de AWK

Padrões, campos, arrays, funções, processamento de texto

Básico

Executando 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
```

Estrutura do Programa

`awk 'pattern { action }'`	Forma básica — ação executa quando padrão coincide
`BEGIN { ... }`	Executa uma vez antes de processar a entrada
`END { ... }`	Executa uma vez após toda a entrada ser processada
`Sem padrão`	Ação executa para cada linha
`Sem ação`	Ação padrão é `{ print }`

Padrões & Ações

Tipos de Padrão

```
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
```

Referência de Padrões

`/regex/`	Comparar linha com regex
`$1 ~ /pat/`	Campo coincide com regex
`$1 !~ /pat/`	Campo não coincide com regex
`expr1, expr2`	Intervalo: do primeiro ao segundo match
`expr1 && expr2`	E lógico
`expr1 || expr2`	OU lógico
`!expr`	NÃO lógico

Variáveis

Variáveis Built-in

`NR`	Número do registro (linha) atual
`NF`	Número de campos no registro atual
`FS`	Separador de campo de entrada (padrão: espaço)
`OFS`	Separador de campo de saída (padrão: espaço)
`RS`	Separador de registro de entrada (padrão: newline)
`ORS`	Separador de registro de saída (padrão: newline)
`FILENAME`	Nome do arquivo de entrada atual
`FNR`	Número do registro no arquivo atual

Variáveis de Usuário

```
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
```

Campos

Acesso a Campos

`$0`	Linha inteira atual
`$1, $2, ...`	Primeiro, segundo, ... campo
`$NF`	Último campo
`$(NF-1)`	Penúltimo campo

Separadores de Campo

```
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
```

Controle de Fluxo

Condicionais & Laços

```
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
```

Comandos de Controle

`if (cond) { ... } else { ... }`	Condicional
`for (i = 0; i < n; i++) { ... }`	Laço for estilo C
`for (key in array) { ... }`	Iterar chaves de array
`while (cond) { ... }`	Laço while
`do { ... } while (cond)`	Laço do-while
`next`	Pular para próximo registro de entrada
`exit`	Parar processamento, executar bloco END

Funções

Funções Definidas pelo Usuário

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

Funções Numéricas

`int(x)`	Truncar para inteiro
`sqrt(x)`	Raiz quadrada
`sin(x), cos(x)`	Funções trigonométricas
`log(x), exp(x)`	Logaritmo natural e exponencial
`rand()`	Float aleatório entre 0 e 1
`srand(seed)`	Inicializar gerador de números aleatórios

Arrays

Arrays Associativos

```
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
```

Operações com Arrays

`arr[key] = val`	Definir elemento
`arr[key]`	Obter elemento (auto-cria no acesso)
`key in arr`	Testar se chave existe
`delete arr[key]`	Deletar elemento único
`delete arr`	Deletar array inteiro
`for (k in arr)`	Iterar sobre chaves (sem ordem)
`length(arr)`	Número de elementos (gawk)

Funções de String

Referência de Strings

`length(s)`	Comprimento da string
`substr(s, start, len)`	Substring (índice base 1)
`index(s, target)`	Posição de target em s (0 se não encontrar)
`split(s, arr, sep)`	Dividir string em array
`sub(pat, repl, s)`	Substituir primeira ocorrência
`gsub(pat, repl, s)`	Substituir todas as ocorrências
`match(s, pat)`	Posição do match regex (define RSTART, RLENGTH)
`tolower(s) / toupper(s)`	Conversão de caso
`sprintf(fmt, ...)`	Formatar string (como printf do C)

Exemplos de String

```
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

Saída

```
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
```

Referência de I/O

`print`	Imprimir com ORS (newline por padrão)
`printf fmt, ...`	Impressão formatada (sem newline)
`print > file`	Redirecionar saída para arquivo
`print >> file`	Acrescentar saída ao arquivo
`print | cmd`	Enviar saída por pipe para comando
`getline < file`	Ler uma linha do arquivo
`cmd | getline var`	Ler saída de comando para variável
`close(file)`	Fechar arquivo ou pipe

Padrões Comuns

One-Liners

```
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
```

Receitas

`CSV para TSV`	`awk -F, 'BEGIN{OFS="\t"} {$1=$1; print}'`
`Somar coluna 2`	`awk '{ s += $2 } END { print s }'`
`Primeiras N linhas`	`awk 'NR <= 10'` (como head)
`Contagem de frequência`	`awk '{ c[$1]++ } END { for (k in c) print k, c[k] }'`
`Entre marcadores`	`awk '/BEGIN/,/END/'`
`Imprimir n-ésimo campo`	`awk '{ print $N }'` (substituir N)