Riferimento Rapido AWK

Pattern, campi, array, funzioni, elaborazione testo

Basi

Eseguire AWK

```
awk '{ print }' file.txt          # stampa ogni riga
awk '{ print $1 }' file.txt       # stampa primo campo
awk -F: '{ print $1 }' /etc/passwd  # delimitatore personalizzato
awk -f script.awk file.txt        # esegui da file
cmd | awk '{ print $2 }'          # input da pipe
```

Struttura del Programma

`awk 'pattern { action }'`	Forma base — l'azione viene eseguita quando il pattern corrisponde
`BEGIN { ... }`	Eseguito una volta prima di elaborare l'input
`END { ... }`	Eseguito una volta dopo tutto l'input
`Nessun pattern`	L'azione viene eseguita per ogni riga
`Nessuna azione`	L'azione predefinita è `{ print }`

Pattern e Azioni

Tipi di Pattern

```
awk '/error/' file.txt            # corrispondenza regex
awk '$3 > 100' file.txt           # confronto
awk 'NR >= 5 && NR <= 10' file.txt  # intervallo di righe
awk '/start/,/end/' file.txt      # pattern intervallo
```

Riferimento Pattern

`/regex/`	Confronta la riga con regex
`$1 ~ /pat/`	Il campo corrisponde alla regex
`$1 !~ /pat/`	Il campo non corrisponde alla regex
`expr1, expr2`	Intervallo: dalla prima alla seconda corrispondenza
`expr1 && expr2`	AND logico
`expr1 || expr2`	OR logico
`!expr`	NOT logico

Variabili

Variabili Predefinite

`NR`	Numero del record (riga) corrente
`NF`	Numero di campi nel record corrente
`FS`	Separatore di campo input (predefinito: spazio bianco)
`OFS`	Separatore di campo output (predefinito: spazio)
`RS`	Separatore di record input (predefinito: newline)
`ORS`	Separatore di record output (predefinito: newline)
`FILENAME`	Nome del file di input corrente
`FNR`	Numero del record nel file corrente

Variabili Utente

```
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
```

Campi

Accesso ai Campi

`$0`	Intera riga corrente
`$1, $2, ...`	Primo, secondo, ... campo
`$NF`	Ultimo campo
`$(NF-1)`	Penultimo campo

Separatori di Campo

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

Controllo del Flusso

Condizioni e Cicli

```
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  # salta righe corrispondenti
```

Istruzioni di Controllo

`if (cond) { ... } else { ... }`	Condizionale
`for (i = 0; i < n; i++) { ... }`	Ciclo for in stile C
`for (key in array) { ... }`	Itera sulle chiavi dell'array
`while (cond) { ... }`	Ciclo while
`do { ... } while (cond)`	Ciclo do-while
`next`	Salta al record di input successivo
`exit`	Ferma elaborazione, esegui blocco END

Funzioni

Funzioni Definite dall'Utente

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

Funzioni Numeriche

`int(x)`	Tronca a intero
`sqrt(x)`	Radice quadrata
`sin(x), cos(x)`	Funzioni trigonometriche
`log(x), exp(x)`	Logaritmo naturale ed esponenziale
`rand()`	Numero float casuale tra 0 e 1
`srand(seed)`	Imposta il seme del generatore casuale

Array

Array Associativi

```
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
```

Operazioni sugli Array

`arr[key] = val`	Imposta elemento
`arr[key]`	Ottieni elemento (creato automaticamente all'accesso)
`key in arr`	Verifica se la chiave esiste
`delete arr[key]`	Elimina singolo elemento
`delete arr`	Elimina l'intero array
`for (k in arr)`	Itera sulle chiavi (non ordinato)
`length(arr)`	Numero di elementi (gawk)

Funzioni Stringa

Riferimento Stringhe

`length(s)`	Lunghezza stringa
`substr(s, start, len)`	Sottostringa (indicizzata da 1)
`index(s, target)`	Posizione di target in s (0 se non trovato)
`split(s, arr, sep)`	Divide la stringa in array
`sub(pat, repl, s)`	Sostituisce la prima corrispondenza
`gsub(pat, repl, s)`	Sostituisce tutte le corrispondenze
`match(s, pat)`	Posizione della corrispondenza regex (imposta RSTART, RLENGTH)
`tolower(s) / toupper(s)`	Conversione maiuscolo/minuscolo
`sprintf(fmt, ...)`	Formatta stringa (come printf in C)

Esempi Stringa

```
awk '{ gsub(/old/, "new"); print }' f    # sostituzione simile a sed
awk '{ print toupper($0) }' f             # tutto maiuscolo
awk '{ print substr($0, 1, 40) }' f       # tronca a 40
```

I/O

Output

```
awk '{ print $1, $2 }' f              # separato da spazio
awk '{ printf "%s,%d\n", $1, $2 }' f  # output formattato
awk '{ print $1 > "out.txt" }' f      # reindirizza su file
awk '{ print $1 >> "out.txt" }' f     # accoda su file
```

Riferimento I/O

`print`	Stampa con ORS (newline predefinito)
`printf fmt, ...`	Stampa formattata (senza newline finale)
`print > file`	Reindirizza output su file
`print >> file`	Accoda output su file
`print | cmd`	Invia output a un comando via pipe
`getline < file`	Legge una riga da file
`cmd | getline var`	Legge output di un comando nella variabile
`close(file)`	Chiude file o pipe

Pattern Comuni

One-Liner

```
awk '{ sum += $1 } END { print sum }' f   # somma colonna
awk 'END { print NR }' f                   # conta righe
awk '!seen[$0]++' f                        # rimuovi duplicati
awk 'NF' f                                 # rimuovi righe vuote
awk '{ print NF }' f                       # campi per riga
```

Ricette

`CSV a TSV`	`awk -F, 'BEGIN{OFS="\t"} {$1=$1; print}'`
`Somma colonna 2`	`awk '{ s += $2 } END { print s }'`
`Prime N righe`	`awk 'NR <= 10'` (come head)
`Conteggio frequenza`	`awk '{ c[$1]++ } END { for (k in c) print k, c[k] }'`
`Tra marcatori`	`awk '/BEGIN/,/END/'`
`Stampa campo N`	`awk '{ print $N }'` (sostituire N)