

# AWK Quick Reference

Patterns, fields, arrays, functions, text processing

## Basics

### Running 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
```

### Program Structure

**awk 'pattern { action }'** Basic form — action runs when pattern matches

**BEGIN { ... }** Runs once before processing input

**END { ... }** Runs once after all input is processed

**No pattern** Action runs for every line

**No action** Default action is **{ print }**

## Patterns & Actions

### Pattern Types

```
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
```

### Pattern Reference

**/regex/** Match line against regex

**\$1 ~ /pat/** Field matches regex

**\$1 !~ /pat/** Field does not match regex

**expr1, expr2** Range: from first match to second

**expr1 && expr2** Logical AND

**expr1 || expr2** Logical OR

**!expr** Logical NOT

## Variables

### Built-in Variables

**NR** Current record (line) number

**NF** Number of fields in current record

**FS** Input field separator (default: whitespace)

**OFS** Output field separator (default: space)

**RS** Input record separator (default: newline)

**ORS** Output record separator (default: newline)

**FILENAME** Current input filename

**FNR** Record number in current file

### User Variables

```
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
```

## Fields

### Field Access

**\$0** Entire current line

**\$1, \$2, ...** First, second, ... field

**\$NF** Last field

**\$(NF-1)** Second-to-last field

### Field Separators

```
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
```

## Control Flow

### Conditionals & Loops

```
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
```

### Control Statements

**if (cond) { ... } else { ... }** Conditional

**for (i = 0; i < n; i++) { ... }** C-style for loop

**for (key in array) { ... }** Iterate array keys

**while (cond) { ... }** While loop

**do { ... } while (cond)** Do-while loop

**next** Skip to next input record

**exit** Stop processing, run END block

## Functions

### User-Defined Functions

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

### Numeric Functions

**int(x)** Truncate to integer

**sqrt(x)** Square root

**sin(x), cos(x)** Trigonometric functions

**log(x), exp(x)** Natural log and exponent

**rand()** Random float between 0 and 1

**srand(seed)** Seed the random number generator

## Arrays

### Associative Arrays

```
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
```

### Array Operations

**arr[key] = val** Set element

**arr[key]** Get element (auto-creates on access)

**key in arr** Test if key exists

**delete arr[key]** Delete single element

**delete arr** Delete entire array

**for (k in arr)** Iterate over keys (unordered)

**length(arr)** Number of elements (gawk)

## String Functions

### String Reference

**length(s)** String length

**substr(s, start, len)** Substring (1-indexed)

**index(s, target)** Position of target in s (0 if not found)

**split(s, arr, sep)** Split string into array

**sub(pat, repl, s)** Replace first match

**gsub(pat, repl, s)** Replace all matches

**match(s, pat)** Position of regex match (sets RSTART, RLENGTH)

**tolower(s) / toupper(s)** Case conversion

**sprintf(fmt, ...)** Format string (like C printf)

## String Examples

```
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

### Output

```
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 Reference

**print** Print with ORS (newline by default)

**printf fmt, ...** Formatted print (no trailing newline)

**print > file** Redirect output to file

**print >> file** Append output to file

**print | cmd** Pipe output to a command

**getline < file** Read a line from file

**cmd | getline var** Read command output into variable

**close(file)** Close file or pipe

## Common Patterns

### 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
```

## Recipes

**CSV to TSV** `awk -F, 'BEGIN{OFS="\t"} {$1=$1; print}'`

**Sum column 2** `awk '{ s += $2 } END { print s }'`

**Top N lines** `awk 'NR <= 10' (like head)`

**Frequency count** `awk '{ c[$1]++ } END { for (k in c) print k, c[k] }'`

**Between markers** `awk '/BEGIN/,/END/'`

**Print nth field** `awk '{ print $N }'` (replace N)