

AWK クイックリファレンス

パターン、フィールド、配列、関数、テキスト処理

基本操作

AWKの実行

```
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
```

プログラム構造

```
awk 'pattern { action }' 基本形 - パターンが一致したときにアクションを実行
BEGIN { ... }           入力処理前に一度だけ実行
END { ... }             全入力処理後に一度だけ実行
パターンなし           全行に対してアクションを実行
アクションなし         デフォルトアクションは { print }
```

パターン & アクション

パターンの種類

```
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
```

パターン早見表

/regex/	行を正規表現でマッチ
\$1 ~ /pat/	フィールドが正規表現にマッチ
\$1 !~ /pat/	フィールドが正規表現にマッチしない
expr1, expr2	範囲: 最初のマッチから 2 番目のマッチまで
expr1 && expr2	論理 AND
expr1 expr2	論理 OR
!expr	論理 NOT

変数

組み込み変数

NR	現在のレコード (行) 番号
NF	現在のレコードのフィールド数
FS	入力フィールド区切り文字 (デフォルト: 空白)
OFS	出力フィールド区切り文字 (デフォルト: スペース)
RS	入力レコード区切り文字 (デフォルト: 改行)
ORS	出力レコード区切り文字 (デフォルト: 改行)
FILENAME	現在の入力ファイル名
FNR	現在のファイル内のレコード番号

ユーザー変数

```
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
```

フィールド

フィールドへのアクセス

\$0	現在行全体
\$1, \$2, ...	1 番目、2 番目、...のフィールド
\$NF	最後のフィールド
\$(NF-1)	最後から 2 番目のフィールド

フィールド区切り文字

```
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
```

制御フロー

条件分岐 & ループ

```
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
```

制御文

if (cond) { ... } else { ... }	条件分岐
for (i = 0; i < n; i++) { ... }	C スタイル for ループ
for (key in array) { ... }	配列キーをイテレート
while (cond) { ... }	while ループ
do { ... } while (cond)	do-while ループ
next	次の入力レコードへスキップ
exit	処理を停止し END ブロックを実行

関数

ユーザー定義関数

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

数値関数

int(x)	整数に切り捨て
sqrt(x)	平方根
sin(x), cos(x)	三角関数
log(x), exp(x)	自然対数と指数
rand()	0~1の乱数 (浮動小数点)
srand(seed)	乱数シードの設定

配列

連想配列

```
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
```

配列操作

arr[key] = val	要素を設定
arr[key]	要素を取得 (アクセス時に自動生成)
key in arr	キーが存在するかテスト
delete arr[key]	1 要素を削除
delete arr	配列全体を削除
for (k in arr)	キーをイテレート (順不同)
length(arr)	要素数 (gawk)

文字列関数

文字列関数早見表

length(s)	文字列の長さ
substr(s, start, len)	部分文字列 (1 始まり)
index(s, target)	s 中の target の位置 (なければ 0)
split(s, arr, sep)	文字列を配列に分割
sub(pat, repl, s)	最初のマッチを置換
gsub(pat, repl, s)	全マッチを置換
match(s, pat)	正規表現マッチの位置 (RSTART, RLENGTH を設定)
tolower(s) / toupper(s)	大文字小文字変換
sprintf(fmt, ...)	書式文字列 (C の printf と同様)

文字列の例

```
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

出力

```
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 早見表

print	ORS 付きで出力 (デフォルトは改行)
printf fmt, ...	書式付き出力 (末尾に改行なし)
print > file	ファイルに出力をリダイレクト
print >> file	ファイルに追記
print cmd	コマンドにパイプ
getline < file	ファイルから 1 行読み込む
cmd getline var	コマンド出力を変数に読み込む
close(file)	ファイルまたはパイプを閉じる

よく使うパターン

ワンライナー

```
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
```

レシピ

CSV → TSV	awk -F, 'BEGIN{OFS="\t"} {\$1=\$1; print}'
2 列目の合計	awk '{ s += \$2 } END { print s }'
先頭 N 行	awk 'NR <= 10' (head と同様)
頻度カウント	awk '{ c[\$1]++ } END { for (k in c) print k, c[k] }'
マーカー間を抽出	awk '/BEGIN/,/END/'
N 列目を表示	awk '{ print \$N }' (N を置換)