AWK 빠른 참조

패턴, 필드, 배열, 함수, 텍스트 처리

기본

AWK 실행

```
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
```

프로그램 구조

`awk 'pattern { action }'`	기본 형태 — 패턴 매칭 시 액션 실행
`BEGIN { ... }`	입력 처리 전 한 번 실행
`END { ... }`	모든 입력 처리 후 한 번 실행
`No pattern`	모든 줄에 대해 액션 실행
`No action`	기본 액션은 `{ print }`

패턴 & 액션

패턴 유형

```
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
```

패턴 참조

`/regex/`	줄을 정규식으로 매칭
`$1 ~ /pat/`	필드가 정규식에 매칭
`$1 !~ /pat/`	필드가 정규식에 미매칭
`expr1, expr2`	범위: 첫 매칭부터 두 번째 매칭까지
`expr1 && expr2`	논리 AND
`expr1 || expr2`	논리 OR
`!expr`	논리 NOT

변수

내장 변수

`NR`	현재 레코드(줄) 번호
`NF`	현재 레코드의 필드 수
`FS`	입력 필드 구분자 (기본값: 공백)
`OFS`	출력 필드 구분자 (기본값: 공백)
`RS`	입력 레코드 구분자 (기본값: 개행)
`ORS`	출력 레코드 구분자 (기본값: 개행)
`FILENAME`	현재 입력 파일명
`FNR`	현재 파일 내 레코드 번호

사용자 변수

```
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
```

필드

필드 접근

`$0`	현재 줄 전체
`$1, $2, ...`	첫 번째, 두 번째, ... 필드
`$NF`	마지막 필드
`$(NF-1)`	마지막에서 두 번째 필드

필드 구분자

```
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
```

제어 흐름

조건문 & 반복문

```
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
```

제어 구문

`if (cond) { ... } else { ... }`	조건문
`for (i = 0; i < n; i++) { ... }`	C 스타일 for 반복문
`for (key in array) { ... }`	배열 키 순회
`while (cond) { ... }`	while 반복문
`do { ... } while (cond)`	do-while 반복문
`next`	다음 입력 레코드로 건너뜀
`exit`	처리 중단, END 블록 실행

함수

사용자 정의 함수

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

수치 함수

`int(x)`	정수로 절삭
`sqrt(x)`	제곱근
`sin(x), cos(x)`	삼각 함수
`log(x), exp(x)`	자연로그 및 지수
`rand()`	0과 1 사이의 난수
`srand(seed)`	난수 생성기 시드 설정

배열

연관 배열

```
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
```

배열 연산

`arr[key] = val`	요소 설정
`arr[key]`	요소 읽기 (접근 시 자동 생성)
`key in arr`	키 존재 여부 확인
`delete arr[key]`	단일 요소 삭제
`delete arr`	배열 전체 삭제
`for (k in arr)`	키 순회 (순서 무작위)
`length(arr)`	요소 수 (gawk)

문자열 함수

문자열 참조

`length(s)`	문자열 길이
`substr(s, start, len)`	부분 문자열 (1-인덱스)
`index(s, target)`	s 내 target 위치 (없으면 0)
`split(s, arr, sep)`	문자열을 배열로 분할
`sub(pat, repl, s)`	첫 번째 매칭 교체
`gsub(pat, repl, s)`	모든 매칭 교체
`match(s, pat)`	정규식 매칭 위치 (RSTART, RLENGTH 설정)
`tolower(s) / toupper(s)`	대소문자 변환
`sprintf(fmt, ...)`	형식 문자열 (C printf 방식)

문자열 예시

```
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

출력

```
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 참조

`print`	ORS와 함께 출력 (기본: 개행)
`printf fmt, ...`	형식화 출력 (개행 없음)
`print > file`	파일로 출력 리다이렉트
`print >> file`	파일에 출력 추가
`print | cmd`	명령어로 출력 파이프
`getline < file`	파일에서 한 줄 읽기
`cmd | getline var`	명령어 출력을 변수로 읽기
`close(file)`	파일 또는 파이프 닫기

일반 패턴

원라이너

```
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
```

레시피

`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)