



Awk cheatsheet

This is a one page quick reference cheat sheet to the **GNU awk**, which covers commonly used awk expressions and commands.

Getting Started

| Have a try | |
|--|---------------------------|
| \$ awk -F: '{print \$1, \$NF}' /etc/passwd | |
| -F: | Colon as a separator |
| {...} | Awk program |
| print | Prints the current record |
| \$1 | First field |
| \$NF | Last field |
| /etc/passwd | Input data file |

| Awk program | | |
|---------------------------------------|---------------------|--|
| BEGIN | {<initializations>} | |
| <pattern 1> | {<program actions>} | |
| <pattern 2> | {<program actions>} | |
| ... | | |
| END | {< final actions >} | |
| | Example | |
| awk ' | | |
| BEGIN { print "\n>>>Start" } | | |
| !/(login shutdown)/ { print NR, \$0 } | | |
| END { print "<<END\n" } | | |
| ' /etc/passwd | | |

| Variables | | |
|-----------|--------------|-------------|
| \$1 | \$2/(\$NF-1) | \$3/\$NF |
| \$0/NR | ID | WEBSITE |
| \$0/NR | 1 | quickref.me |
| \$0/NR | 2 | google.com |
| | | 25 |

| Awk program examples | |
|--|-------------------------------|
| awk 'BEGIN {print "hello world"}' | # Prints "hello world" |
| awk -F: '{print \$1}' /etc/passwd | # -F: Specify field separator |
| # /pattern/ Execute actions only for matched pattern | |
| awk -F: '/root/ {print \$1}' /etc/passwd | |
| # BEGIN block is executed once at the start | |
| awk -F: 'BEGIN { print "uid"} { print \$1 }' /etc/passwd | |
| # END block is executed once at the end | |
| awk -F: '{print \$1} END { print "-done-"}' /etc/passwd | |

| |
|---|
| # First and last field |
| awk -F: '{print \$1,\$NF}' /etc/passwd |
| # With line number |
| awk -F: '{print NR, \$0}' /etc/passwd |
| # Second last field |
| awk -F: '{print \$(NF-1)}' /etc/passwd |
| # Custom string |
| awk -F: '{print \$1 "=" \$6}' /etc/passwd |
| See: Variables |

| Generate 1000 spaces | |
|----------------------|--|
| awk 'BEGIN{ | |
| while (a++ < 1000) | |
| s=s " "; | |
| print s | |
| } | |
| See: Loops | |

| Arrays | |
|---------------------------------------|--|
| awk 'BEGIN { | |
| fruits["mango"] = "yellow"; | |
| fruits["orange"] = "orange"; | |
| for(fruit in fruits) { | |
| print "The color of " fruit " is " fr | |
| } | |
| } | |
| See: Arrays | |

| Functions | |
|--|--|
| # => 5 | |
| awk 'BEGIN{print length("hello")}' | |
| # => HELLO | |
| awk 'BEGIN{print toupper("hello")}' | |
| # => hel | |
| awk 'BEGIN{print substr("hello", 1, 3)}' | |
| See: Functions | |

Awk Variables

| Build-in variables | |
|--------------------|---|
| \$0 | Whole line |
| \$1, \$2...\$NF | First, second... last field |
| NR | Number of Records |
| NF | Number of Fields |
| OFS | Output Field Separator (default " ") |
| FS | input Field Separator (default " ") |
| ORS | Output Record Separator (default "\n") |
| RS | input Record Separator (default "\n") |
| FILENAME | Name of the file |

| Expressions | |
|--------------------|------------------------------------|
| \$1 == "root" | First field equals root |
| {print \$(NF-1)} | Second last field |
| NR!=1{print \$0} | From 2th record |
| NR > 3 | From 4th record |
| NR == 1 | First record |
| END{print NR} | Total records |
| BEGIN{print OFMT} | Output format |
| {print NR, \$0} | Line number |
| {print NR " " \$0} | Line number (tab) |
| {\$1 = NR; print} | Replace 1th field with line number |

| Examples | |
|--------------------------------------|--|
| Print sum and average | |
| awk -F: '{sum += \$3}' | |
| END { print sum, sum/NR } | |
| ' /etc/passwd | |
| Printing parameters | |
| awk 'BEGIN { | |
| for (i = 1; i < ARGV; i++) | |
| print ARGV[i] }' a b c | |
| Output field separator as a comma | |
| awk 'BEGIN { FS=":";OFS="," } | |
| {print \$1,\$2,\$3,\$4}' /etc/passwd | |
| Position of match | |

| | |
|-------------------|------------------|
| \$NF > 4 | Last field > 4 |
| NR % 2 == 0 | Even records |
| NR==10, NR==20 | Records 10 to 20 |
| BEGIN{print ARGC} | Total arguments |

| | |
|--------------------|---------------------|
| ORS=NR%5?" ":"\\n" | Concatenate records |
|--------------------|---------------------|

| | |
|-------------------|---|
| Position of match | awk 'BEGIN { if (match("One Two Three", "Tw")) print START }' |
| Length of match | awk 'BEGIN { if (match("One Two Three", "re")) print RLENGTH }' |
| Defining variable | |

| Environment Variables | |
|-----------------------|---|
| ARGC | Number or arguments |
| ARGV | Array of arguments |
| FNR | File Number of Records |
| OFMT | Format for numbers (default "%.6g") |
| RSTART | Location in the string |
| RLENGTH | Length of match |
| SUBSEP | Multi-dimensional array separator (default "\034") |
| ARGIND | Argument Index |

| GNU awk only | |
|--------------|-----------------------|
| ENVIRON | Environment variables |
| IGNORECASE | Ignore case |
| CONVFMT | Conversion format |
| ERRNO | System errors |
| FIELDWIDTHS | Fixed width fields |

| Defining variable | |
|---------------------------------------|---------------------------------|
| awk -v var1="Hello" -v var2="World" ' | END {print var1, var2}' |
| ' </dev/null | |
| Use shell variables | |
| awk -v varName="\$PWD" ' | END {print varName}' </dev/null |

Awk Operators

| Operators | |
|------------------------|-------------------|
| {print \$1} | First field |
| \$2 == "foo" | Equals |
| \$2 != "foo" | Not equals |
| "foo" in array | In array |
| Regular expression | |
| /regex/ | Line matches |
| !/regex/ | Line not matches |
| \$1 ~ /regex/ | Field matches |
| \$1 !~ /regex/ | Field not matches |
| More conditions | |
| (\$2 <= 4 \$3 < 20) | Or |
| (\$1 == 4 && \$3 < 20) | And |

| Operations | | |
|-----------------------|----|----|
| Arithmetic operations | | |
| + | - | * |
| / | % | ++ |
| -- | | |
| Shorthand assignments | | |
| += | -= | *= |
| /= | %= | |
| Comparison operators | | |
| == | != | < |
| > | <= | >= |

| Examples | |
|---|-------------|
| awk 'BEGIN { if ("foo" ~ /^fo+\$/) print "Fooy!" }' | Not match |
| awk 'BEGIN { if ("boo" !~ "^fo+\$") print "Boo!" }' | if in array |
| awk 'BEGIN { assoc["foo"] = "bar"; assoc["bar"] = "baz"; if ("foo" in assoc) print "Fooy!" }' | |
| awk ' | |

Awk Functions

| Common functions | |
|---------------------|--|
| index(s,t) | Position in string s where string t occurs, 0 if not found |
| length(s) | Length of string s (or \$0 if no arg) |
| rand | Random number between 0 and 1 |
| substr(s,index,len) | Return len-char substring of s that begins at index (counted from 1) |
| srand | Set seed for rand and return previous seed |
| int(x) | Truncate x to integer value |
| split(s,a,fs) | Split string s into array a split by fs, returning length of a |
| match(s,r) | Position in string s where regex r occurs, 0 if not found |
| sub(r,t,s) | Substitute t for first occurrence of regex r in string s (or \$0 if s not given) |
| gsub(r,t,s) | Substitute t for all occurrences of regex r in string s |
| system(cmd) | Execute cmd and return exit status |
| tolower(s) | String s to lowercase |
| toupper(s) | String s to uppercase |

| User defined function | |
|--------------------------------|--------------------------------|
| awk ' | |
| # Returns minimum number | function find_min(num1, num2){ |
| if (num1 < num2) | return num1 |
| return num2 | } |
| # Returns maximum number | function find_max(num1, num2){ |
| if (num1 > num2) | return num1 |
| return num2 | } |
| # Main function | function main(num1, num2){ |
| result = find_min(num1, num2) | print "Minimum =", result |
| result = find_max(num1, num2) | print "Maximum =", result |
| # Script execution starts here | } |
| BEGIN { | main(10, 60) |

```
getline
```

```
Set $0 to next input record from current input file.
```

```
main(...)
```

Awk Arrays

Array with index

```
awk 'BEGIN {
    arr[0] = "foo";
    arr[1] = "bar";
    print(arr[0]); # => foo
    delete arr[0];
    print(arr[0]); # => ""
}'
```

Array with key

```
awk 'BEGIN {
    assoc["foo"] = "bar";
    assoc["bar"] = "baz";
    print("baz" in assoc); # => 0
    print("foo" in assoc); # => 1
}'
```

Array with split

```
awk 'BEGIN {
    split("foo:bar:baz", arr, ":");
    for (key in arr)
        print arr[key];
}'
```

Array with asort

```
awk 'BEGIN {
    arr[0] = 3
    arr[1] = 2
    arr[2] = 4
    n = asort(arr)
    for (i = 1; i <= n ; i++)
        print(arr[i])
}'
```

Multi-dimensional

```
awk 'BEGIN {
    multidim[0,0] = "foo";
    multidim[0,1] = "bar";
    multidim[1,0] = "baz";
    multidim[1,1] = "boo";
}'
```

Multi-dimensional iteration

```
awk 'BEGIN {
    array[1,2]=3;
    array[2,3]=5;
    for (comb in array) {
        split(comb,sep,SUBSEP);
        print sep[1], sep[2],
        array[sep[1],sep[2]]
    }
}'
```

Awk Conditions

if...else statement

```
awk -v count=2 'BEGIN {
    if (count == 1)
        print "Yes";
    else
        print "Huh?";
}'
```

Ternary operator

```
awk -v count=2 'BEGIN {
    print (count==1) ? "Yes" : "Huh?";
}'
```

Exists

```
awk 'BEGIN {
    assoc["foo"] = "bar";
    assoc["bar"] = "baz";
    if ("foo" in assoc)
        print "Fooy!";
}'
```

Not exists

```
awk 'BEGIN {
    assoc["foo"] = "bar";
    assoc["bar"] = "baz";
    if ("Huh" in assoc == 0 )
        print "Huh!";
}'
```

switch

```
awk -F: '{
    switch (NR * 2 + 1) {
        case 3:
        case "11":
            print NR - 1
            break
        case /2[:digit:]+/:
            print NR
        default:
            print NR + 1
        case -1:
            print NR * -1
    }
}' /etc/passwd
```

Awk Loops

for...i

```
awk 'BEGIN {
    for (i = 0; i < 10; i++)
        print "i=" i;
}'
```

Powers of two between 1 and 100

```
awk 'BEGIN {
    for (i = 1; i <= 100; i *= 2)
        print i
}'
```

for...in

```
awk 'BEGIN {
    assoc["key1"] = "val1"
    assoc["key2"] = "val2"
    for (key in assoc)
        print assoc[key];
}'
```

Arguments

```
awk 'BEGIN {
    for (argnum in ARGV)
        print ARGV[argnum];
}' a b c
```

Examples

Reverse records

```
awk -F: '{ x[NR] = $0 }
END {
    for (i = NR; i > 0; i--)
        print x[i]
}'
' /etc/passwd
```

Reverse fields

```
awk -F: '{
    for (i = NF; i > 0; i--)
        printf("%s ", $i);
    print ""
}' /etc/passwd
```

Sum by record

```
awk -F: '{
    s=0;
    for (i = 1; i <= NF; i++)
        s += $i;
    print s
}'
' /etc/passwd
```

Sum whole file

while

```
awk 'BEGIN {
    while (a < 10) {
        print "- " concatenation: " a
        a++;
    }
}'
```

do...while

```
awk '{
    i = 1
}'
```

Break

```
awk 'BEGIN {
    break_num = 5
    for (i = 0; i < 10; i++) {
        print i
        if (i == break_num)
            break
    }
}'
```

```
i = 1
do {
    print $0
    i++
} while (i <= 5)
}' /etc/passwd
```

```
awk 'BEGIN {
    for (x = 0; x <= 10; x++) {
        if (x == 5 || x == 6)
            continue
        printf "%d ", x
    }
    print ""
}'
```

```
awk -F: '
    for (i = 1; i <= NF; i++)
        s += $i;
}
END{print s}
' /etc/passwd
```

Awk Formatted Printing

| Usage | |
|--|--|
| Right align | |
| awk 'BEGIN{printf " %10s \n", "hello"}' | |
| hello | |
| Left align | |
| awk 'BEGIN{printf " %-10s \n", "hello"}' | |
| hello | |

| Common specifiers | |
|-------------------|-----------------------|
| c | ASCII character |
| d | Decimal integer |
| e, E, f | Floating-point format |
| o | Unsigned octal value |
| s | String |
| % | Literal % |

| Space | |
|------------------------------------|-------|
| awk -F: '{ | |
| printf "%-10s %s\n", \$1, \$(NF-1) | |
| }' /etc/passwd head -n 3 | |
| Outputs | |
| root | /root |
| bin | /bin |
| daemon | /sbin |

| Header | |
|--|-------|
| awk -F: 'BEGIN { | |
| printf "%-10s %s\n", "User", "Home" | |
| printf "%-10s %s\n", "----", "----" | |
| { printf "%-10s %s\n", \$1, \$(NF-1) } | |
| ' /etc/passwd head -n 5 | |
| Outputs | |
| User | Home |
| ---- | ---- |
| root | /root |
| bin | /bin |
| daemon | /sbin |

Miscellaneous

| Regex Metacharacters | | |
|----------------------|---|----|
| \ | ^ | \$ |
| . | [|] |
| | (|) |
| * | + | ? |

| Escape Sequences | |
|------------------|---------------------|
| \b | Backspace |
| \f | Form feed |
| \n | Newline (line feed) |
| \r | Carriage return |
| \t | Horizontal tab |
| \v | Vertical tab |

| Run script | |
|--------------------------------|--|
| \$ cat demo.awk | |
| #!/usr/bin/awk -f | |
| BEGIN { x = 23 } | |
| { x += 2 } | |
| END { print x } | |
| \$ awk -f demo.awk /etc/passwd | |
| 69 | |

Also see

The GNU Awk User's Guide (www-zeuthen.desy.de)
 AWK cheatsheet (gist.github.com)

Related Cheatsheet

| |
|---------------------|
| Emacs Cheatsheet |
| Quick Reference |
| |
| Markdown Cheatsheet |
| Quick Reference |

| |
|-----------------|
| Grep Cheatsheet |
| Quick Reference |
| |
| Sed Cheatsheet |
| Quick Reference |

Recent Cheatsheet

| |
|--------------------------|
| Google Search Cheatsheet |
| Quick Reference |
| |
| ES6 Cheatsheet |
| Quick Reference |



Share quick reference and cheat sheet for developers.

中文版 #Notes



SPONSOR Contentful — Make it easy for developers to compose content across channels. Centralize your content in one location.