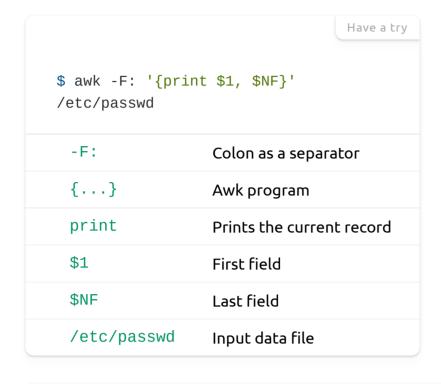
# Awk cheatsheet

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

### # Getting Started



```
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
```

```
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
```

```
Variables
            $1
                    $2/$(NF-1)
                                   $3/$NF
 $0/NR ▶
             ID
                    WEBSITE
                                    URI
 $0/NR ▶
                    quickref.me
                                    awk
             1
 $0/NR ▶
             2
                    google.com
                                    25
 # 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
                                   Conditions
 awk -F: '$3>30 {print $1}' /etc/passwd
See: Conditions
```

```
awk 'BEGIN{
    while (a++ < 1000)
        s=s " ";
    print s
}'</pre>
See: Loops
```

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

Arrays

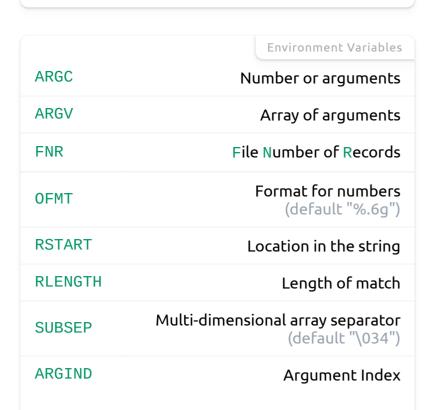
```
# => 5
awk 'BEGIN{print length("hello")}'
# => HELLO
awk 'BEGIN{print toupper("hello")}'
# => hel
awk 'BEGIN{print substr("hello", 1, 3)}
See: Functions
```

#### # Awk Variables

```
Build-in variables
$0
                                        Whole line
$1, $2...$NF
                          First, second... last field
NR
                               Number of Records
\mathsf{NF}
                                 Number of Fields
                           Output Field Separator
0FS
                                       (default " ")
                             input Field Separator
FS
                                      (default " ")
                         Output Record Separator
ORS
                                     (default "\n")
                           input Record Separator
RS
                                     (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)
                         Replace 1th field with
{$1 = NR; print}
                                 line number
$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
```

```
Examples
Print sum and average
 awk -F: \{sum += \$3\}
       END { print sum, sum/NR }
  ' /etc/passwd
Printing parameters
 awk 'BEGIN {
      for (i = 1; i < ARGC; i++)
          print ARGV[i] }' a b c
Output field separator as a comma
 awk 'BEGIN { FS=":";0FS=","}
      {print $1,$2,$3,$4}' /etc/passwd
Position of match
 awk 'BEGIN {
      if (match("One Two Three", "Tw"))
          print RSTART }'
Length of match
```



Environment variables
Ignore case
Conversion format
System errors
Fixed width fields

```
awk -v var1="Hello" -v var2="Wold" '
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
                                Field matches
$1 ~ /regex/
$1 !~ /regex/
                             Field not matches
               More conditions
($2 <= 4 || $3 < 20)
                                          Ог
(\$1 == 4 \&\& \$3 < 20)
                                        And
```

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

#### # 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, or 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
getline
                                                        Set $0 to next input record from current input file.
```

```
User defined function
awk
    # Returns minimum number
   function find_min(num1, num2){
       if (num1 < num2)</pre>
       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)
```

### # Awk Arrays

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

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

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

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

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

```
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

```
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?";
}'
```

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

    Not exists

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

```
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

```
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
}'</pre>
```

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

do...while

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

```
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
```

Break

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

Continue

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

awk 'BEGIN {

print ""

}'

 $break_num = 5$ 

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

### # Awk Formatted Printing

```
Right align

awk 'BEGIN{printf "|%10s|\n", "hello"}'

| hello|

Left align
```

```
Common specifiers

ASCII character

Decimal integer

e, E, f
Floating-point format

Unsigned octal value

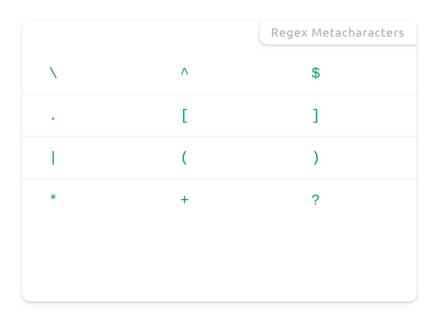
S
String
```

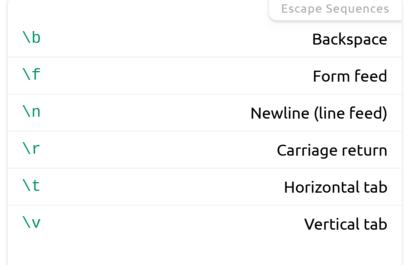
```
awk -F: '{
    printf "%-10s %s\n", $1, $(NF-1)
}' /etc/passwd | head -n 3
Outputs
```

% Literal %

```
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
            /sbin
 daemon
```

## # Miscellaneous







### # Also see

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