

# awk

????

- <https://linuxhandbook.com/awk-command-tutorial/>
- [Running Awk in parallel to process 256M records](#)
- [Awk one-liners](#)

????

```
awk '/^this/{print $0}'      # sed -n '/^this/p'
```

????

```
# [ ] [ ] [ ] [ ]
awk { gsub(/;/, "") }
```

????????

[illegible]

```
awk '!x[$0]++' filewithdups > newfile
```

??? disabled ?????? 1, 3 ????

```
awk '/disabled/{print $1, $3}'
```

?????

```
awk '{print "up " $1 /60 " minutes"}' /proc/uptime
```

```
df -IP -text4 |awk '{sum += $4} END {printf "%d GiB\n", sum/1048576}'
```

```
df -IP -text4 |awk '{sum += $4} END {printf "%d GiB\n", sum/2**20}'
```

?????

```
df -k |grep "/dev/" | awk '($2 > 0 && ((1 - $3/$2) > 0.9) ) {print $0 }'
```

```
awk -F" " '{print ($7 != "A")?$0"***":$0}' myfile
```

???????9??????? 0x00000000 ????????

```
cat info.out | awk '($9 != "0x00000000") {print}'
```

?? uid >= 500 ? <= 10000 ??

```
export UGIDLIMIT=500
```

```
awk -v LIMIT=$UGIDLIMIT -F: '($3>=LIMIT) && ($3<=10000)' /etc/passwd
```

??????????

```
$ awk 'BEGIN {print 12345678901234567890}'
```

```
1.23457e+19
```

```
###
```

```
$ awk 'BEGIN {printf("%d\n", 12345678901234567890)}'
```

```
12345678901234567168
```

```
###
```

```
$ awk 'BEGIN {OFMT="%.0f"; print 12345678901234567890}'
```

```
12345678901234567168
```

?? uid=0 ???

```
awk -F: '($3 == "0") {print}' /etc/passwd
```

???????????

```
ls -ltd */ | awk -F ' ' '{print $NF}'
```

??????? 64 ??

```
awk 'length > 64'
```

??????

```
awk '{ printf("1-minute: %s\n5-minute: %s\n15-minute: \n\n", $1, $2, $3); }' /proc/loadavg
```

?????????

```
foldersize() {  
    if [ -d $1 ]; then  
        ls -alRF $1/ | grep '^-' | awk 'BEGIN {tot=0} { tot=tot+$5 } END { print tot }'  
    else  
        echo "$1: folder does not exist"  
    fi  
}
```

??????(??? "?? " ???????)

```
awk '{total=total+NF}; END {print total+0}'
```

??????????????

```
# lspci -v | awk '/ATI/,/^$/'  
01:03.0 VGA compatible controller: ATI Technologies Inc Rage XL (rev 27) (prog-if 00 [VGA])  
    Subsystem: Compaq Computer Corporation: Unknown device 001e  
    Flags: bus master, stepping, medium devsel, latency 64  
    Memory at fc000000 (32-bit, non-prefetchable) [size=16M]  
    I/O ports at 3000 [size=256]  
    Memory at fbff0000 (32-bit, non-prefetchable) [size=4K]  
    Capabilities: [5c] Power Management version 2
```

?? Kill ?????? /plugins/mactrack ???

```
ps -ef | grep "/plugins/mactrack" | awk '{system("kill " $2);}'
```

?????????????: ?? + =

```
#   
ping 8.8.8.8 | awk -F[\ =] '{print $10}'
```

?????????(? 10, 20, 30, ...)

```
awk '!(NR % 10)' file
```

????????????????

```
# Sort nmon files by time, delete a file far from the current time, always keep only one nmon file:
ls -t ~/.nmon | awk '\.nmon/ {if (NR > 1){system ("rm " $1)}}'
```

CSV ?????

- [GoAWK](#) - A POSIX-compliant AWK interpreter written in Go, with CSV support

```
# 13 APPNAME
cat ./ISO27001/db2/fdctest_validate.csv | awk -F, '{a[$13]++} END {for (k in a) print k, a[k]}'

# 3
awk '{ $3 = toupper(substr($3,1,1)) substr($3,2) } $3' FS=, OFS=, file

# 3
awk '$3 { print toupper($0); }' file
```

?????

```
awk NF test.txt
```

?: ???????

script: [aud2csv.sh](#)

Raw Data:

```
timestamp=2023-01-08-23.13.02.322992;
category=CHECKING;
audit event=CHECKING_OBJECT;
event correlator=107;
event status=0;
database=RPTDB;
userid=winmfg;
authid=WINMFG;
application id=10.8.25.30.64020.230108151301;
application name=EXCEL.EXE;
package schema=NULLID;
package name=SYSSH200;
```

???:

```

f1=1; f2=3; f3=5; f4=7; f5=9; f6=11; f7=13; f8=15; f9=17; f10=19; f11=21; f12=23; f13=25; f14=27;
f15=29; f16=31; f17=33; f18=35; f19=37; f20=39; f21=41; f22=43
}

# CSV Header
if (! headline)
{
    headline = printf( "%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s", $f1,
    $f2, $f3, $f4, $f5, $f6, $f7, $f8, $f9, $f10, $f11, $f12, $f13, $f14, $f15, $f16, $f17, $f18, $f19, $f20, $f21, $f22 );
    print headline;
}

# CSV
dataline = printf("%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s", $TIMESTAMP,
$CATEGORY, $AUDIT_EVENT, $EVENT_CORRELATOR, $EVENT_STATUS, $DATABASE, $USERID, $AUTHID,
$APPLICATION_ID, $APPLICATION_NAME, $PACKAGE_SCHEMA, $PACKAGE_NAME, $PACKAGE_SECTION,
$OBJECT_SCHEMA, $OBJECT_NAME, $OBJECT_TYPE, $ACCESS_APPROVAL_REASON, $ACCESS_ATTEMPTED,
$LOCAL_TRANSACTION_ID, $GLOBAL_TRANSACTION_ID, $INSTANCE_NAME, $HOSTNAME );
print dataline;

```

## ?: ? bash ??

```

# System Memory Section
mem_raw=$(free -b | awk '/Mem:/ {print $2, $3, $4, $7}')
read -r -a mem_array <<< "$mem_raw"

mem_total_bytes=${mem_array[0]}
mem_used_bytes=${mem_array[1]}
mem_free_bytes=${mem_array[2]}
mem_avail_bytes=${mem_array[3]}

mem_total_gb=$(awk "BEGIN {printf \"%.2f\", $mem_total_bytes / 1024 / 1024 / 1024}")
mem_used_gb=$(awk "BEGIN {printf \"%.2f\", $mem_used_bytes / 1024 / 1024 / 1024}")
mem_free_gb=$(awk "BEGIN {printf \"%.2f\", $mem_free_bytes / 1024 / 1024 / 1024}")
mem_avail_gb=$(awk "BEGIN {printf \"%.2f\", $mem_avail_bytes / 1024 / 1024 / 1024}")

avail_mem_percentage=$(awk "BEGIN {printf \"%.2f\", 100 * $mem_avail_bytes / $mem_total_bytes}")
is_low_mem=$(awk "BEGIN {print ($avail_mem_percentage < 10)}")

if (( is_low_mem )); then

```

```

mem_info="Total: ${mem_total_gb} GB, Used: ${mem_used_gb} GB, Free: ${mem_free_gb} GB, Available:
${mem_avail_gb} GB \033[0;31m(Warning!: ${avail_mem_percentage}%)\033[0m"
else
    mem_info="Total: ${mem_total_gb} GB, Used: ${mem_used_gb} GB, Free: ${mem_free_gb} GB, Available:
    ${mem_avail_gb} GB"
fi

```

## Cheatsheet

# Linux AWK Command Cheat sheet

### AWK BASICS

```
$ awk -F ":" '{print $1, $6, $NF}' /etc/passwd
```

-F ":" option sets the field separator to ":"  
 '{...}' block inside single quotes is the awk program to execute.  
 print The print statement is used to output the select fields  
 \$1 Refers to the first field (username username in this case)  
 \$6 Refers to the sixth field (home directory in this case)  
 \$NF Refers to the last field  
 /etc/passwd The input file name/path

### BUILT-IN VARIABLES

\$0 Reference the whole record/line  
 \$N Reference Nth field (i.e. \$1, \$2, \$3....\$NF first, second, third and last field)  
 FS Field separator of input file (default whitespace " ")  
 NR Line number records  
 NF Number of fields in current record  
 OFS Output data field separator (default whitespace " ")  
 ORS Output data record separator (default newline "\n")  
 RS Input record separator (default newline "\n")  
 FILENAME Name of the input file

### ESCAPE SEQUENCE

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

### ARITHMATIC OPERATORS

+ Addition  
 - Subtraction  
 \* Multiplication  
 / Division  
 ++ Increment  
 -- Decrement  
 % Modulo

### ASSIGNMENT OPERATORS

= Assignment  
 += Addition and assignment  
 -= Subtraction and assignment  
 \*= Multiplication and assignment  
 /= Division and assignment  
 %= Modulo and assignment

### COMPARISON OPERATORS

== Equal to  
 != Not equal to  
 > Greater than  
 < Less than  
 >= Greater than or equal to  
 <= Less than or equal to

### BOOLEAN OPERATOR

&& Logical AND  
 || Logical OR  
 ! Logical NOT  
 ?: Ternary Operator

### CONDITIONAL OPERATOR

?: Ternary Operator

### REGEX METACHARACTERS

. Match any Character Except New Line  
 ? Match 0 or One characters  
 \* Match 0 or More characters  
 + Match 1 or More characters  
 ^ Beginning of line.  
 \$ End of line.  
 ^\$ Empty line.  
 < Start of word.  
 > End of word.

### ENVIRONMENT VARIABLES

FNR Represents the record number within the current input file.  
 CONVFMF The format used for converting numbers to strings. (default %.6g)  
 SUBSEP The separator used for multidimensional array subscripts (default 034)  
 OFMT Controls the format used for numeric output (default %.6g)  
 ARGV Stores the number of command-line arguments passed to awk  
 ENVIRON Contains the command-line arguments passed to awk as an array  
 Provides access to the environment variables as an associative array  
 RSTART Stores the starting position of the most recent string match or substitution operation in awk.  
 RLENGTH Holds the length of the most recent string match or substitution operation in awk.  
 ARGIND Represents the index of the current input file being processed in awk.  
 IGNORECASE Ignore case  
 ERRNO Stores the error number associated with the most recent system error.  
 FIELDWIDTHS Specifies a whitespace-separated list of field widths to split input records into fixed-width fields.

### 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.

### FORMAT SPECIFIERS

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

### AWK FOR-IN LOOP EXAMPLE

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

### AWK WHILE LOOP EXAMPLE

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

### AWK FOR LOOP EXAMPLE

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

### AWK DO LOOP EXAMPLE

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

### AWK IF-ELSE STATEMENT

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

### AWK IF-ELSE STATEMENT

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

### AWK SWITCH-CASE

```
awk '{
    caseValue = $1;
    switch (caseValue) {
        case "apple":
            print "It's an apple!";
            break;
        case "banana":
            print "It's a banana!";
            break;
        default:
            print "Unknown fruit!";
    }
}' fruits.txt
```

### AWK ARRAYS

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

### AWK MULTI-DIMENSIONAL ARRAY

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

@linuxopsys

# AWK COMMAND CHEAT SHEET

## Definition:

'awk' is a powerful text processing tool used for data extraction and manipulation. It operates per line, allowing you to specify patterns and actions on lines matching those patterns.

## Common awk Variables:

**\$0:** Used to specify the whole line  
**\$1:** Specifies the first field  
**\$2:** Specifies the second field

## Basic Syntax:

```
$ awk options 'selection_criteria {action}' input-file > output-file
```

## AWK Command Usefulness

- Changing data file
- Producing formatted reports

## Examples:

### Printing specific columns

```
awk '{print $2 "\t" $3}' file.txt
```

### Printing all lines in a file

```
awk '{print $0}' file.txt
```

### Printing columns that match a specific pattern

```
awk '/a/ {print $3 "\t" $4}' file.txt
```

### Saving output of 'awk' to a different file

```
awk '/a/ {print $3 "\t" $4}' file.txt > Output.txt
```

## Common awk Options:

Options	Description
<b>-F [separator]</b>	Used to specify a file separator. The default separator is a blank space.
<b>-f [filename]</b>	Used to specify the file containing the awk script. Reads the awk program source from the specified file, instead of the first command-line argument.
<b>-v</b>	Used to assign a variable.

## Program:

```
BEGIN    {<initializations>}  
<pattern 1> {<program actions>}  
<pattern 2> {<program actions>}  
...  
END      {< final actions >}
```

## Common Functions:

```
index(s,t)  
length(s)  
rand  
substr(s,index,len)  
srand  
int(x)  
split(s,a,fs)  
match(s,r)  
sub(r,t,s)  
gsub(r,t,s)  
system(cmd)  
tolower(s)  
toupper(s)  
getline
```

## Operators:

Arithmetic	Shorthand	Comparison	Regular expression
+ - * / % ++ --	+= -= *= /= % =	== != < > <= >=	<b>/regex/</b> Line matches <b>!/regex/</b> Line not matches <b>\$1 ~ /regex/</b> Field matches <b>\$1 !~ /regex/</b> Field not matches

## Built-in awk Variables:

**NR:** Number of Records  
**NF:** Number of Fields  
**OFS:** Output Field Separator  
**FS:** Input Field Separator  
**ORS:** Output Record Separator  
**RS:** Input Record Separator  
**FILENAME:** Name of the file

## Common Specifiers:

```
c ASCII character  
d Decimal integer  
e,E,F Floating-point format  
o Unsigned octal value  
s String  
% Literal %
```

## Escape Sequences:

```
\b Backspace  
\f Form feed  
\n Newline  
\r Carriage return  
\t Horizontal tab  
\v Vertical tab
```

## Condition:

```
awk -F: '$3>30 {print $1}' /etc/passwd
```

## Regex Metacharacters:

```
\ ^ $ . [ ] | ( ) * + ?
```

Revision #37

Created 1 June 2020 06:57:14 by Admin

Updated 2 April 2025 15:52:02 by Admin