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(/\;/, "") }
```

????????

?????? sort ? uniq ??????????????????????????????????????? awk ????????????? awk ??**?????**??????????

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

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

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

?????

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


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

```
# 統計第 13 欄 APPNAME 每個值的計數
cat ./ISO27001/db2/fdctest_validate.csv |  awk -F, '{a[$13]++} END {for (k in a) print k, a[k]}'


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


# 欄位 3 變更成大寫
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;
```

```
package section=4;
object schema=ISTRPT;
object name=FHOPEHS;
object type=TABLE;
access approval reason=OBJECT;
access attempted=SELECT;
local transaction id=0x00000001b6550792;
global transaction id=0x0000000000000000000000000000000000000000;
instance name=istrpt;
hostname=BSMDB_B;
```

????:

```
# 宣告: 紀錄的間隔符號為 空行, 欄位間隔符號為 =
# 一個段落文字為一筆紀錄, 每一行以 = 為間隔區別不同欄位
BEGIN {
    FS="=";
    RS="";
}

# 過濾條件: 欄位總數是 nfp2 的值的資料
# 此實例包含了有不一致欄位數的紀錄, 所以必須先做過濾
NF==nfp2 {

}

# 移除 分號字元
{ gsub(/\;/, "") }

# TIMESTAMP = 欄位 2
# CATEGORY = 欄位 4
# f1 = 欄位 1
# f2 = 欄位 3
{
    TIMESTAMP=2; CATEGORY=4; AUDIT_EVENT=6; EVENT_CORRELATOR=8; EVENT_STATUS=10;
DATABASE=12; USERID=14; AUTHID=16; APPLICATION_ID=18; APPLICATION_NAME=20; PACKAGE_SCHEMA=22;
PACKAGE_NAME=24; PACKAGE_SECTION=26; OBJECT_SCHEMA=28; OBJECT_NAME=30; OBJECT_TYPE=32;
ACCESS_APPROVAL_REASON=34; ACCESS_ATTEMPTED=36; LOCAL_TRANSACTION_ID=38;
GLOBAL_TRANSACTION_ID=40; INSTANCE_NAME=42; HOSTNAME=44;
```

```
    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 = sprintf( "%s,%s,%s,%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 = sprintf("%s,%s,%s,%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

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.

Basic Syntax:

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

AWK Command Usefulness

- Changing data file
- Producing formatted reports

Common awk Variables:

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

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
-F [separator]	Used to specify a file separator. The default separator is a blank space.
-f [filename]	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.
-v	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

/regex/	Line matches
!/regex/	Line not matches
$1 ~ /regex/	Field matches
$1 !~ /regex/	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:

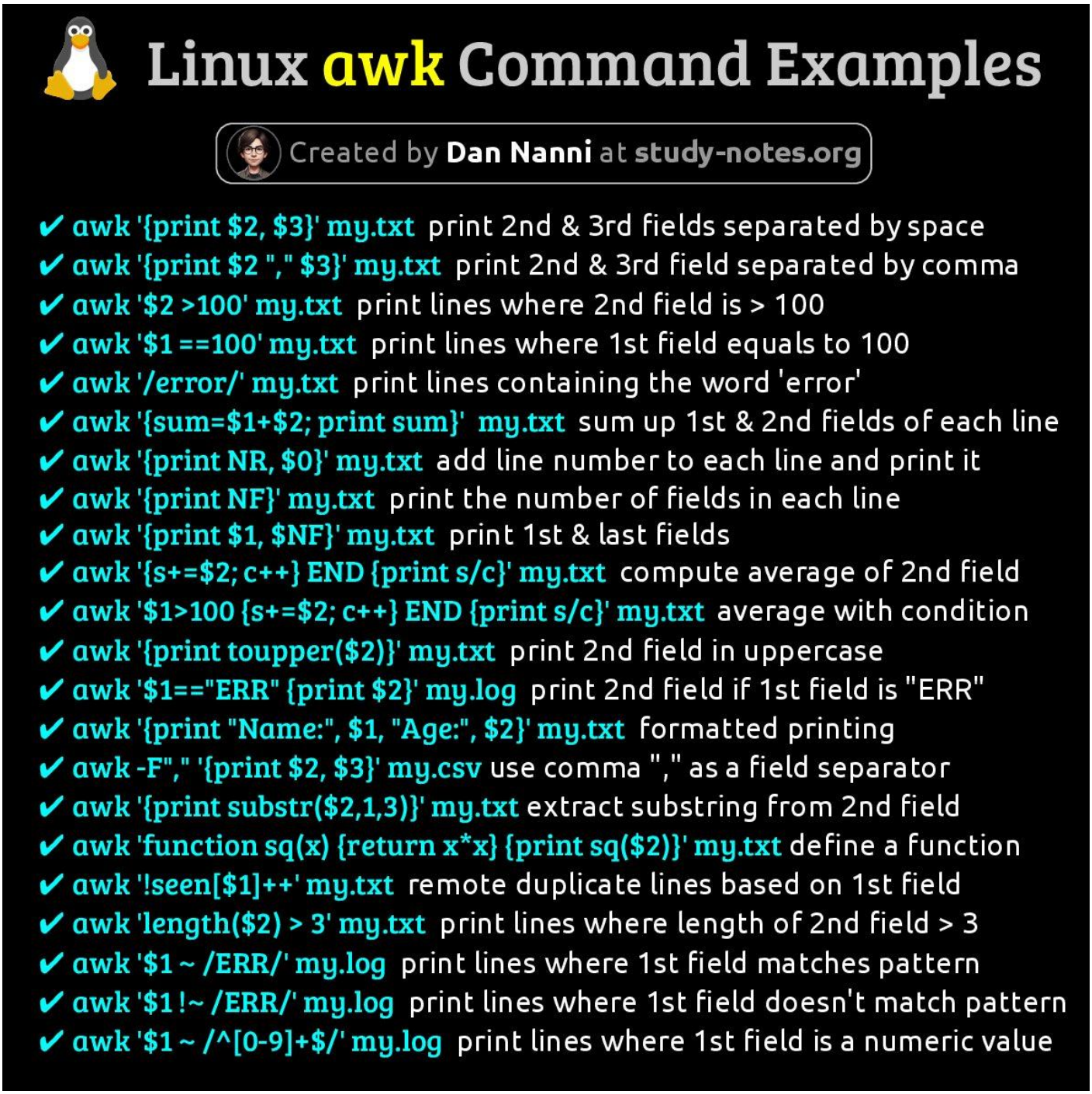
\b	Backspace
\f	Form feed
\n	Newline
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab

Condition:

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

Regex Metacharacters:

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



Revision #38
Created 1 June 2020 06:57:14 by Admin
Updated 28 April 2025 20:48:24 by Admin