Awk Quick Reference

Last modified: Fri Nov 27 09:44:57 2020

Awk Quick Reference - by Bruce Barnett @grymoire

AWK can be thought of as a program that can read rows and columns of information, and generate data - like a spreadsheet. It can also be thought of as a simple C interpretor, as AWK and C have similar features.

MAWK Usage

From mawk(1) mawk [-W option] [-F value] [-v var=value] [--] 'program text' [file ...] mawk [-W option] [-F value] [-v var=value] [-f program-file] [--] [file ...]

GAWK Usage

From gawk --help:

```
Usage: gawk [POSIX or GNU style options] -f progfile [--] file ...
Usage: gawk [POSIX or GNU style options] [--] 'program' file ...
POSIX options:          GNU long options:
        -f progfile             --file=progfile
        -F fs                   --field-separator=fs
        -v var=val              --assign=var=val
        -m[fr] val
        -O                      --optimize
        -W compat               --compat
        -W copyleft             --copyleft
        -W copyright            --copyright
        -W dump-variables[=file]        --dump-variables[=file]
        -W exec=file            --exec=file
        -W gen-po               --gen-po
        -W help                 --help
        -W lint[=fatal]         --lint[=fatal]
        -W lint-old             --lint-old
        -W non-decimal-data     --non-decimal-data
        -W profile[=file]       --profile[=file]
        -W posix                --posix
        -W re-interval          --re-interval
        -W source=program-text  --source=program-text
```

```
-W traditional          --traditional
-W usage                --usage
-W use-lc-numeric       --use-lc-numeric
-W version              --version
```

Program

There are only a few commands in AWK. The Tables below are from my awk tutorial. Check this out if you need a beter explanation. The basic operation of AWK is that a line from the input file is read, and for each line, the AWK script is executed.

Basic Structure

The basic structure of an AWK script consists of one or more of the following types of lines:

```
pattern { statements }
function name(parameter_list) { statements }
```

Patterns

If a pattern is not specified, it defaults to be "true", and every line read will cause the starement to be executed,

A pattern can have the following form.

```
BEGIN
END
/regular expression/
relational expression
pattern && pattern
pattern || pattern
pattern ? pattern : pattern
(pattern)
! pattern
pattern1, pattern2 - Range pattern
```

Statements

Statements have the following syntax, separated by a new line or a semicolon.

if (*conditional*) *statement* [else *statement*]
while (*conditional*) *statement*
for (*expression* ; *conditional* ; *expression*) *statement*
for (*variable* in *array*) *statement*
break
continue
{ [*statement*] ...}
variable=expression
print [*expression-list*] [> *expression*]
printf *format* [, *expression-list*] [> *expression*]

next
exit

Special Variables

AWK Table 14 Special Variables				
Variable	**Purpose**	**AWK**	**NAWK**	**GAWK**
FS	Field separator	Yes	Yes	Yes
NF	Number of Fields	Yes	Yes	Yes
RS	Record separator	Yes	Yes	Yes
NR	Number of input records	Yes	Yes	Yes
FILENAME	Current filename	Yes	Yes	Yes
OFS	Output field separator	Yes	Yes	Yes
ORS	Output record separator	Yes	Yes	Yes
ARGC	# of arguments		Yes	Yes
ARGV	Array of arguments		Yes	Yes
ARGIND	Index of ARGV of current file			Yes
FNR	Input record number		Yes	Yes
OFMT	Ouput format (default "%.6g")		Yes	Yes
RSTART	Index of first character after match()		Yes	Yes
RLENGTH	Length of string after match()		Yes	Yes
SUBSEP	Default separator with multiple subscripts in array (default "\034")		Yes	Yes
ENVIRON	Array of environment variables			Yes
IGNORECASE	Ignore case of regular expression			Yes
CONVFMT	conversion format (default: "%.6g")			Yes
ERRNO	Current error after getline failure			Yes
FIELDWIDTHS	list of field widths (instead of using FS)			Yes
BINMODE	Binary Mode (Windows)			Yes
LINT	Turns --lint mode on/off			Yes
PROCINFO	Array of informaiton about current AWK program			Yes
RT	Record terminator			Yes
TEXTDOMAIN	Text domain (i.e. localization) of current AWK program			Yes

Variables $1, $2, etc.

The variables $1, $2, etc created by spliting up each line into fields. $1 is the first field (i.e. the first column), $2 is the second, etc.

Relational expressions are created using unary, binary, relational, the following operators:

Unary variables change the value of a variable.

Unary Operators ***variable operator*** ***operator variable***	
Operator	**Meaning**
++	Increment by 1
--	Decrement by 1

Binary operators combine values.

AWK Table 1 **Binary Operators** ***expression operator expression***		
Operator	**Type**	**Meaning**
+	Arithmetic	Addition
-	Arithmetic	Subtraction
*	Arithmetic	Multiplication
/	Arithmetic	Division
%	Arithmetic	Modulo
<space>	String	Concatenation

Assignment variables change the values of variables.

AWK Table 2 **Assignment Operators** ***variable operator expression***	
Operator	**Meaning**
+=	Add result to variable
-=	Subtract result from variable
*=	Multiply variable by result
/=	Divide variable by result
%=	Apply modulo to variable

Relational operators compare values.

AWK Table 3 **Relational Operators** ***expression operator expression***	
Operator	**Meaning**
==	Is equal
!=	Is not equal to
>	Is greater than
>=	Is greater than or equal to
<	Is less than
<=	Is less than or equal to

Certain characters that follow a '\' have a special meaning.

AWK Table 5 Escape Sequences	
Sequence	**Description**
\a	ASCII bell (NAWK/GAWK only)
\b	Backspace
\f	Formfeed
\n	Newline
\r	Carriage Return
\t	Horizontal tab
\v	Vertical tab (NAWK only)
\ddd	Character (1 to 3 octal digits) (NAWK only)
\xdd	Character (hexadecimal) (NAWK only)
\<Any other character>	That character

The printf or sprintf statement generates a string using a format field and variables.

`printf(`*`Format`*`,variable, variable,...) statement,`

Inside the format field, you can define how the variables should be output.

AWK Table 6 Format Specifiers	
Specifier	**Meaning**
%c	ASCII Character
%d	Decimal integer
%e	Floating Point number (engineering format)
%f	Floating Point number (fixed point format)
%g	The shorter of e or f, with trailing zeros removed
%o	Octal
%s	String
%x	Hexadecimal
%%	Literal %

Here are some examples of format conversions.

AWK Table 7 Example of format conversions		
Format	**Value**	**Results**
%c	100.0	d
%c	"100.0"	1 (NAWK?)

%c	42	"
%d	100.0	100
%e	100.0	1.000000e+02
%f	100.0	100.000000
%g	100.0	100
%o	100.0	144
%s	100.0	100.0
%s	"13f"	13f
%d	"13f"	0 (AWK)
%d	"13f"	13 (NAWK)
%x	100.0	64

Here are more complex format conversion examples

AWK Table 8 Examples of complex formatting		
Format	**Variable**	**Results**
%c	100	"d"
%10c	100	" d"
%010c	100	"000000000d"
%d	10	"10"
%10d	10	" 10"
%10.4d	10.123456789	" 0010"
%10.8d	10.123456789	" 00000010"
%.8d	10.123456789	"00000010"
%010d	10.123456789	"0000000010"
%e	987.1234567890	"9.871235e+02"
%10.4e	987.1234567890	"9.8712e+02"
%10.8e	987.1234567890	"9.87123457e+02"
%f	987.1234567890	"987.123457"
%10.4f	987.1234567890	" 987.1235"
%010.4f	987.1234567890	"00987.1235"
%10.8f	987.1234567890	"987.12345679"
%g	987.1234567890	"987.123"
%10g	987.1234567890	" 987.123"
%10.4g	987.1234567890	" 987.1"
%010.4g	987.1234567890	"00000987.1"
%.8g	987.1234567890	"987.12346"

%o	987.1234567890	"1733"
%10o	987.1234567890	" 1733"
%010o	987.1234567890	"0000001733"
%.8o	987.1234567890	"00001733"
%s	987.123	"987.123"
%10s	987.123	" 987.123"
%10.4s	987.123	" 987."
%010.8s	987.123	"000987.123"
%x	987.1234567890	"3db"
%10x	987.1234567890	" 3db"
%010x	987.1234567890	"00000003db"
%.8x	987.1234567890	"000003db"

The AWK variants have build-in functions. There are numeric, string, and miscellaneous functions.

AWK Table 9
Numeric Functions

Name	Function	Variant
cos	cosine	GAWK,AWK,NAWK
exp	Exponent	GAWK,AWK,NAWK
int	Integer	GAWK,AWK,NAWK
log	Logarithm	GAWK,AWK,NAWK
sin	Sine	GAWK,AWK,NAWK
sqrt	Square Root	GAWK,AWK,NAWK
atan2	Arctangent	GAWK,NAWK
rand	Random	GAWK,NAWK
srand	Seed Random	GAWK,NAWK

AWK Table 10
String Functions

Name	Variant
index(string,search)	AWK, NAWK, GAWK
length(string)	AWK, NAWK, GAWK
split(string,array,separator)	AWK, NAWK, GAWK
substr(string,position)	AWK, NAWK, GAWK
substr(string,position,max)	AWK, NAWK, GAWK
sub(regex,replacement)	NAWK, GAWK
sub(regex,replacement,string)	NAWK, GAWK
gsub(regex,replacement)	NAWK, GAWK

gsub(regex,replacement,string)	NAWK, GAWK
match(string,regex)	NAWK, GAWK
tolower(string)	GAWK
toupper(string)	GAWK
asort(string,[d])	GAWK
asorti(string,[d])	GAWK
gensub(r,s,h [,t])	GAWK
strtonum(string)	GAWK

AWK Table 11
Miscellaneous Functions

Name	Variant
getline	AWK, NAWK, GAWK
getline <file	NAWK, GAWK
getline variable	NAWK, GAWK
getline variable <file	NAWK, GAWK
"command" | getline	NAWK, GAWK
"command" | getline variable	NAWK, GAWK
system(command)	NAWK, GAWK
close(command)	NAWK, GAWK
systime()	GAWK
strftime(string)	GAWK
strftime(string, timestamp)	GAWK

The *strftime*function has special formats.

AWK Table 12
GAWK's strftime formats

%a	The locale's abbreviated weekday name
%A	The locale's full weekday name
%b	The locale's abbreviated month name
%B	The locale's full month name
%c	The locale's "appropriate" date and time representation
%d	The day of the month as a decimal number (01--31)
%H	The hour (24-hour clock) as a decimal number (00--23)
%I	The hour (12-hour clock) as a decimal number (01--12)
%j	The day of the year as a decimal number (001--366)
%m	The month as a decimal number (01--12)
%M	The minute as a decimal number (00--59)
%p	The locale's equivalent of the AM/PM
%S	The second as a decimal number (00--61).
%U	The week number of the year (Sunday is first day of week)
%w	The weekday as a decimal number (0--6). Sunday is day 0

%W	The week number of the year (Monday is first day of week)
%x	The locale's "appropriate" date representation
%X	The locale's "appropriate" time representation
%y	The year without century as a decimal number (00--99)
%Y	The year with century as a decimal number
%Z	The time zone name or abbreviation
%%	A literal %.

Modern versions of GAWK (Gnu AWK) have additional functions.

AWK Table 13 Optional GAWK strftime formats	
%D	Equivalent to specifying %m/%d/%y
%e	The day of the month, padded with a blank if it is only one digit
%h	Equivalent to %b, above
%n	A newline character (ASCII LF)
%r	Equivalent to specifying %I:%M:%S %p
%R	Equivalent to specifying %H:%M
%T	Equivalent to specifying %H:%M:%S
%t	A TAB character
%k	The hour as a decimal number (0-23)
%l	The hour (12-hour clock) as a decimal number (1-12)
%C	The century, as a number between 00 and 99
%u	is replaced by the weekday as a decimal number [Monday == 1]
%V	is replaced by the week number of the year (using ISO 8601)
%v	The date in VMS format (e.g. 20-JUN-1991)