awk (english) Cheat Sheet

by TME520 (TME520) via cheatography.com/20978/cs/3902/

Usage

```
awk [-v var=val] 'program' [file1 file2...]
awk [-v var=val] -f progfile [file1 file2...]
```

Predefined Variable Summary

FS	Input Field Separator, a space by default.
OFS	Output Field Separator, a space by default.
RS	Record Separator, a newline by default.
ORS	Output Record Separator, a newline by default.
NR	The total Number of input Records seen so far.
NF	The Number of Fields in the current input record.
FILENAME	The name of the current input file. If no files are specified on the command line, the value of FILENAME is "-". However, FILENAME is undefined inside the BEGIN block (unless set by getline).
FNR	Contains number of lines read, but is reset for each file read.
$0	The whole line.
$1, $2...$n	Fields from 1 to *n*.
ARGC	The number of command line arguments (does not include options to gawk, or the program source). Dynamically changing the contents of ARGV control the files used for data.
ARGV	Array of command line arguments. The array is indexed from 0 to ARGC - 1.
ARGIND	The index in ARGV of the current file being processed.

Numeric Functions

atan2(y, x)	Returns the arctangent of y/x in radians.
cos(expr)	Returns the cosine of expr, which is in radians.
exp(expr)	The exponential function.
int(expr)	Truncates to integer.
log(expr)	The natural logarithm function.
rand()	Returns a random number N, between 0 and 1, such that 0 <= N < 1.
sin(expr)	Returns the sine of expr, which is in radians.
sqrt(expr)	The square root function.

Numeric Functions (cont)

srand([expr])	Uses expr as a new seed for the random number generator. If no expr is provided, the time of day is used. The return value is the previous seed for the random number generator.

Bit Manipulation Functions

and(v1, v2)	Return the bitwise AND of the values provided by v1 and v2.
compl(val)	Return the bitwise complement of val.
lshift(val, count)	Return the value of val, shifted left by count bits.
or(v1, v2)	Return the bitwise OR of the values provided by v1 and v2.
rshift(val, count)	Return the value of val, shifted right by count bits.
xor(v1, v2)	Return the bitwise XOR of the values provided by v1 and v2.

String Functions

asort(s [, d])	Returns the number of elements in the source array s. The contents of s are sorted using gawk's normal rules for comparing values, and the indexes of the sorted values of s are replaced with sequential integers starting with 1. If the optional destination array d is specified, then s is first duplicated into d, and then d is sorted, leaving the indexes of the source array s unchanged.
asorti(s [, d])	Returns the number of elements in the source array s. The behavior is the same as that of asort(), except that the array indices are used for sorting, not the array values. When done, the array is indexed numerically, and the values are those of the original indices. The original values are lost; thus provide a second array if you wish to preserve the original.

String Functions (cont)

gensub(r, s, h [, t])	Search the target string t for matches of the regular expression r. If h is a string beginning with g or G, then replace all matches of r with s. Otherwise, h is a number indicating which match of r to replace. If t is not supplied, $0 is used instead. Within the replacement text s, the sequence \n, where n is a digit from 1 to 9, may be used to indicate just the text that matched the n'th parenthesized subexpression. The sequence \0 represents the entire matched text, as does the character &. Unlike sub() and gsub(), the modified string is returned as the result of the function, and the original target string is not changed.
gsub(r, s [, t])	For each substring matching the regular expression r in the string t, substitute the string s, and return the number of substitutions. If t is not supplied, use $0. An & in the replacement text is replaced with the text that was actually matched. Use \& to get a literal & (This must be typed as "\\&").
index(s, t)	Returns the index of the string t in the string s, or 0 if t is not present (this implies that character indices start at one).
length([s])	Returns the length of the string s, or the length of $0 if s is not supplied.

String Functions (cont)

match(s, r [, a])	Returns the position in s where the regular expression r occurs, or 0 if r is not present, and sets the values of RSTART and RLENGTH. Note that the argument order is the same as for the ~ operator: str ~ re. If array a is provided, a is cleared and then elements 1 through n are filled with the portions of s that match the corresponding parenthesized subexpression in r. The 0'th element of a contains the portion of s matched by the entire regular expression r. Subscripts a[n, "start"], and a[n, "length"] provide the starting index in the string and length respectively, of each matching substring.
split(s, a [, r])	Splits the string s into the array a on the regular expression r, and returns the number of fields. If r is omitted, FS is used instead. The array a is cleared first. Splitting behaves identically to field splitting.
sprintf(fmt, expr-list)	Prints expr-list according to fmt, and returns the resulting string.
strtonum(str)	Examines str, and returns its numeric value. If str begins with a leading 0, strtonum() assumes that str is an octal number. If str begins with a leading 0x or 0X, strtonum() assumes that str is a hexadecimal number.
sub(r, s [, t])	Just like gsub(), but only the first matching substring is replaced.
substr(s, i [, n])	Returns the at most n-character substring of s starting at i. If n is omitted, the rest of s is used.

String Functions (cont)

tolower(str)	Returns a copy of the string str, with all the upper-case characters in str translated to their corresponding lower-case counterparts. Non-alphabetic characters are left unchanged.
toupper(str)	Returns a copy of the string str, with all the lower-case characters in str translated to their corresponding upper-case counterparts. Non-alphabetic characters are left unchanged.

Operators

&& || !	Logical operators : AND, OR, NOT
< <= == != >= > ~ !~	Comparison operators.

I/O Statements

close(file [, how])	Close file, pipe or co-process. The optional how should only be used when closing one end of a two-way pipe to a co-process. It must be a string value, either "to" or "from".
getline	Set $0 from next input record; set NF, NR, FNR. Returns 0 on EOF and 1 on an error. Upon an error, ERRNO contains a string describing the problem.
getline <file	Set $0 from next record of file; set NF.
getline var	Set var from next input record; set NR, FNR.
getline var <file	Set var from next record of file.
command | getline [var]	Run command piping the output either into $0 or var, as above. If using a pipe or co-process to getline, or from print or printf within a loop, you must use close() to create new instances.
command |& getline [var]	Run command as a co-process piping the output either into $0 or var, as above. Co-processes are a gawk extension.

I/O Statements (cont)

next	Stop processing the current input record. The next input record is read and processing starts over with the first pattern in the AWK program. If the end of the input data is reached, the END block(s), if any, are executed.
nextfile	Stop processing the current input file. The next input record read comes from the next input file. FILENAME and ARGIND are updated, FNR is reset to 1, and processing starts over with the first pattern in the AWK program. If the end of the input data is reached, the END block(s), are executed.
print	Prints the current record. The output record is terminated with the value of the ORS variable.
print expr-list	Prints expressions. Each expression is separated by the value of the OFS variable. The output record is terminated with the value of the ORS variable.
print expr-list >file	Prints expressions on file. Each expression is separated by the value of the OFS variable. The output record is terminated with the value of the ORS variable.
printf fmt, expr-list	Format and print.
printf fmt, expr-list >file	Format and print on file.
system(cmd-line)	Execute the command cmd-line, and return the exit status.
fflush([file])	Flush any buffers associated with the open output file or pipe file. If file is missing, then stdout is flushed. If file is the null string, then all open output files and pipes have their buffers flushed.
print ... >> file	Appends output to the file.
print ... | command	Writes on a pipe.

I/O Statements (cont)

print ... |& command	Sends data to a co-process.

Time Functions

systime()	Returns the current time of day as the number of seconds since the Epoch (1970-01-01 00:00:00 UTC on POSIX systems).
mktime(datespec)	Turns datespec into a time stamp of the same form as returned by systime(). The datespec is a string of the form YYYY MM DD HH MM SS[DST].
strftime([format [, timestamp]])	Formats timestamp according to the specification in format. The timestamp should be of the same form as returned by systime(). If timestamp is missing, the current time of day is used.If format is missing, a default format equivalent to the output of date(1) is used.

GNU AWK's Command Line Argument Summary

-F fs *or* **--field-sepearator fs**	Use fs for the input field separator (the value of the FS predefined variable).
-v var=val *or* **--assign var=val**	Assign the value val to the variable var, before execution of the program begins. Such variable values are available to the BEGIN block of an AWK program.
-f program-file *or* **--file program-file**	Read the AWK program source from the file program-file, instead of from the first command line argument. Multiple -f (or --file) options may be used.

GNU AWK's Command Line Argument Summary (cont)

-mf NNN *or* **-mr NNN**	Set various memory limits to the value NNN. The f flag sets the maximum number of fields, and the r flag sets the maximum record size (ignored by gawk, since gawk has no pre-defined limits).
-W compat *or* **-W traditional** *or* **--compat--traditional**	Run in compatibility mode. In compatibility mode, gawk behaves identically to UNIX awk; none of the GNU-specific extensions are recognized.
-W dump-variables[=file] *or* **--dump-variables[=file]**	Print a sorted list of global variables, their types and final values to file. If no file is provided, gawk uses a file named awkvars.out in the current directory.
-W help *or* **-W usage** *or* **--help** *or* **--usage**	Print a relatively short summary of the available options on the standard output.