AWK REFERENCE CARD

Command Line

awk PROGRAM FILENAMES...
awk -f PROGRAM-FILE FILENAMES...
awk -F*s*
(sets field separator to string *s*, **-Ft** sets separator to tab)

Limits and Restrictions

100 fields, 2500 characters per input/output line, 1024 characters per field and printf-string, 400 characters maximum quoted string and character class, 15 open files, 1 pipe

Control Flow Statements

if (*expr*) *statement* [**else** *statement*]
if (subscript **in** array) *statement* [**else** *statement*]
while (*expr*) *statement*
for (*expr* ; *expr* ; *expr*) *statement*
for (var **in** array) *statement*
do *statement* **while** (*expr*)
break
continue
next
exit [*expr*]
return [*expr*]

User-Defined Functions

func name(*a, b, c*) { *statement* }
function name(*a, b, c*) { *statement* }

function-name(*expr*, *expr*, ...)

String Functions

gsub(r,s,t)	substitutes *s* for *r* in *t* globally, returns number of substitutions
index(s,t)	returns position of string *t* in *s*, and 0 if not present
length(s)	returns length of *s*
match(s,r)	returns position in *s* where *r* occurs, and 0 if not present
split(s,a,r)	splits *s* into array *a* on *r*, returns number of fields
sprintf(f,e...)	returns *e...* formatted according to format string specified in *f*
sub(r,s,t)	substitutes *s* for first *r* in *t*, returns number of substitutions
substr(s,p,n)	returns substring of *s* length *n* starting at offset position *p*

Arithmetic Functions

atan2(y,x)	arctangent of *y/x* in radians
cos(x)	cosine of *x*, with *x* in radians
exp(x)	exponential function of *x*
int(x)	integer part of *x* truncated towards 0
log(x)	natural logarithm of *x*
rand()	random number between 0 and 1
sin(x)	sine of *x*, with *x* in radians
sqrt(x)	square root of *x*
srand(x)	*x* is new seed for *rand()*

Patterns

BEGIN
END
/regular expression/
relational expression
pattern && pattern
pattern || pattern
(pattern)
!pattern
pattern, pattern

Input/Output

close(filename)	close file
getline	set $0 from next input line, and set NF, NR, FNR
getline < file	set $0 from next input line of file, and set NF
getline var	set var from next input line, and set NR, FNR
getline var < file	set var from next input line of file
print	print current input line
print expr-list	print expressions
print expr-list > file	print expressions to file
printf fmt, expr-list	format and print
printf fmt, expr-list > file	format and print to file
system(cmd-line)	execute command cmd-line, return status

In ***print*** and ***printf*** above, **>>** appends to a *file*, and **| command** writes to a pipe. Similarly, **command | getline** pipes into ***getline***. ***getline*** returns 0 on the end of file, -1 on an error.

Build-In Variables

ARGC	number of command-line arguments
ARGV	array of command-line arguments (0..ARGC-1fR)
FILENAME	name of current input file
FNR	input line number number in current file
FS	input field separator (default blank)
NF	number of fields in input line
NR	number of input lines read so far
OFMT	output format for numbers (default=%.6g)
OFS	output field separator (default=space)
ORS	output line separator (default=newline)
RS	input line separator (default=newline)
RSTART	index of first character matched by match()
RLENGTH	length of string matched by match()
SUBSEP	subscript separator (default="\034")

Operators

= += -= *= /= %= ^=	assignment
?:	conditional expression
|| &&	logical or, logical and
~ !~	regular expression match, negated match
< <= > >= != ==	relationals
(blank)	string concatenation
+ -	add, subtract
*** / %**	multiply, divide, modulus
+ - !	unary plus, unary minus, logical negation
^	exponentional
++ --	increment, decrement
$num	field *num*

Regular Expressions

c	matches no-metacharacter *c*	**^**	matches begin of line or string	**r1|r2**	matches either *r1* or *r2*
\c	matches literal character *c*	**$**	matches end of line or string	**r1r2**	concatenation: matches *r1*, then *r2*
.	matches any character, except newline	**[ab...]**	character-class match any of *abc..*	**(r)**	grouping: matches *r*
		[^ab...]	negated character-class, match any except *abc...* and newline	***r?***	matches zero or one *r*
				r+	matches one or multiple *r*'s
				r*	matches zero or multiple *r*'s