awk [options] -f program-file [--] [[-v] var=value] ... [datendatei] ...
awk [options] [[-v] var=value] ...' program-script' [datendatei] ...

ARGC	Anzahl der übergebenen Argumente (einschl. Options)
ARGIND	Index auf ARGV auf Datei die augenblicklich verarbeitet wird
ARGV	Array command line arguments (indiziert von 0 bis ARGC - 1).
CONVFMT	Das Konvertierungsformat für Zahlen (Default "%.6g")
ENVIRON	Assoziative Tabelle mit denm Environments. Indizierung z.B. ENVIRON["HOME"]
ERRNO	Enthält Fehlermeldung, wenn Fehler bei getline oder close() ein Fehler auftritt
FILENAME	Name der current input-file.
FNR	Satznummer innerhalb der current file
FS	Der input-Field-Separator, Default ist ein Blank und oder TAB. /[\t]*/
IGNORECASE	Kontrolliert die case-sensitivity aller Regular Expression (Default 0)
NF	Anzahl der Felder im augenblicklichen Satz.
NR	Bisherige Anzahl der Eingabesätze.
OFMT	Ausgabeformat für Zahlen (Default "%.6g")
OFS	Output-Field-Separator (Default Blank)
ORS	Output-Record-Separator (Default newline \n)
RS	Eingabe Record-Separator (default new-line \n)
RSTART	Der Index des ersten Character von einem match(); 0 wenn kein match.
RLENGTH	Die Länge des String matched by match(); -1 wenn kein match.
SUBSEP	Trennzeichen zwischen Indices einer Tabelle

Pattern

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

Steueranweisungen

break
continue
if (condition) statement [else statement]
while (condition) statement
do statement while (condition)
for (expr1; expr2; expr3) statement
for (var in array) statement
delete array[index]
delete array
exit [expression]

Metazeichen

c	matches das non-Metazeichen c.
^	der RE beginnt am Anfang einer Zeile oder eines Strings /^d/
$	der RE steht am Ende einer Zeile oder eines Strings /test$/
.	steht für jedes Zeichen (Ausnahme newline). /^.....w/
\c	Quoten des Metazeichen c, ergibt das Literalzeichen c. /^\./
[abc]	Character Class, jedes der Zeichen abc.... /[xyzabc]/
	Bereiche werden durch Bindestrich gebildet /[a-zA-Z]/
[^abc]	Negation , jedes Zeichen das nicht abc... oder newline ist.
re1|re2	Alternation: steht für entweder re1 oder re2. /root|hahn/
re1re2	Concatenation: erst re1, und dann re2. /[abc][0-9]/
re+	steht für ein oder mehrere re's. /[0-9]+/
re*	steht für Null oder mehrere re's. **/[,[\t]*/**
re?	steht für Null oder ein re's. /[0-9]?/
(re)	Zusammenfassung von re . /(,[\t])|([\t])/

Escapesequenzen die in Strings verwendet werden dürfen, können auch in RE verwendet werden.

\\ , \a , \b , \f , \n , \r , \t , \v , *ddd* , *xhh* , \" , \c

Operatoren

~	enthält	ls -l | awk ' $1 ~ /^d/ {print $1, $9} '
!~	enthält nicht	ls -l | awk ' $1 !~ /^d/ {print $1, $9} '
>	größer als	ls -l | awk ' $5 > 30000 '
>=	größer gleich	ls -l | awk ' $2 >= 2 '
==	gleich	ls -l | awk ' $3 == /root/ '
<	kleiner als	ls -l | awk ' $2 < 2 '
<=	kleiner gleich	ls -l | awk ' $5 <= 1024 '
!=	ungleich	ls -l | awk ' $3 != /root/ '
&&	logisch-UND	ls -l | awk '$5 > 1024 && $3 != "root" '
||	logisch-ODER	ls -l | awk '$3 ~ "root" || $3 ~ "hahn" '
!	logisch-NOT	ls -l | awk ' $3 ~ ! "root" '

= += -= Zuweisungsoperatoren können auch in der verkürzten
*= /= %= ^= Form op= verwendet werden.
?: Der 'conditional expression' in der Form expr1 ? expr2 : expr3.
|| Logical OR.
&& Logical AND. $1 ~ /^d/ && $3 ~ user && $9 ~ **/^[^\.]** /{ S += $5 }
~ Match mit Regular Expression
!~ Negierter Match. Achtung: Der RE sollte nur auf der rechten Seite
verwendet werden.
< > <= >= != == Vergleichsoperatoren
blank String concatenation.
+ - Addition and Subtraktion.
* / % Multiplikation, Division und Modulus (Divisionsrest).
+ - ! Unary Plus, unary Minus, und logische Negation.
^ Potenzierung (** ist möglich sowie **= als Zuweisungsoperator).
++ -- Increment und Decrement, Beide können als prefix und postfix
verwendet werden.
$ Field Reference. $n

Spezifikation	Konvertierung
%c	Character, Einzelzeichen
%d	Integer
%e	Gleitkomma, Eponentialschreibweise [-]d.dddddE[+-dd]
%f	Gleitkomma, Ausgabe mit 6 Nachkommastellen [-]ddd.dddddd
	abhängig von CONVFMT)
%g	Gleitkomma, kürzere Darstellung (e oder g)
%s	Stringausgabe
%o	Oktale Ausgabe
%x	Hexadezimale Ausgabe

close(filename) Schließt Datei oder Pipe.
getline Setzt $0 vom nächsten Input-Record. Setzt NF, NR, FNR.
getline < "file" Setze $0 vom nächsten Satz von file. Setzt NF. file muß in "file" Quotes
getline var Setzt var vom nächsten Input-Record. Setzt NF, FNR.
getline var < "file" Setzt var vom nächsten Satz von file. " quoten sonst als awk-Variable interpretiert
next Der nächste Satz wird gelesen und awk beginnt die Auswertung wieder beim ersten Pattern. Bei EOF wird der END Block(s) bearbeitet.
next "file" Bearbeitung curr. file beenden. Nächster Satz wird von der nächsten Eingabedatei gelesen FNR wird zurückgesetzt auf 1. awk beginnt die Aus- wertung wieder mit dem ersten Pattern.
print Gibt den augenblicklichen Satz aus.
print expr-list durch Leerzeichen getrennte Para. werden bei der Ausgabe verbunden. ls -l | awk '{print $1 $3 $9}'
durch Komma getrennte Par., wird der expression durch **OFS** vom nächsten expression getrennt. ls -l | awk '{print $1, $3, $9}' Der Satz wird mit dem mit dem Inhalt der Variablen **ORS** abgeschlossen.
print expr-list > "file" Jeder expression wird durch den Inhalt von **OFS** vom
nächsten expression getrennt. Der Satz wird mit dem mit dem Inhalt der Variablen **ORS** abgeschlossen.
printf fmt, expr-list Formatierte Ausgabe, (siehe Formatierte Ausgabe)

Form	**verändert**
getline	$0, NF, NR, FNR
getline *var*	*var,* NR, FNR
getline < file	$0, NF
getline *var* < file	*var*
cmd | getline	$0, NF
cmd | getline *var*	*var*

if (val in array) statement
for (var in array) statement
delete arry

gsub(r,s, [t]) global ersetzen alle r (RE) durch s (String) in $0 oder in t (Textvariablen)
geb="29.01.44" ; gsub(/[0-9]+/, "(&) ", geb) --> "(29).(01).(44)"
index(s,t) liefert die Position vom ersten t im String s. Wenn t
nicht in s enthalten ist, wird 0 zurückgegeben
index("Reportgenerator","or") -> 4

length(s) liefert die Länge von String s zurück
x = length("Reportgenerator") -> 15
print length($0)

match(s,r) liefert die Position von r im String s. Wenn r nicht in s
enthalten ist, wird 0 zurückgegeben. Im Unterschied
zu index() wir als Suchmuster ein regular expression
akzeptiert. Zusätzlich werden die Systemvariablen RSTART (Anfangsposi. des gefundenen Strings) und
RLENGTH (Länge des gefundenen Musters) gesetzt.
r nicht gefunden ist RSTART=0 und RLENGTH=1.

```
{ if (match($0, /a.?i/))
        print RSTART, RLENGTH, $0 }
```

split(s,a,[r}) splittet s in Einzelfelder. Zur Feldtrennung wird die Variable FS b.z.w. r herangezogen.
split($0,zeile,/[0-9]:/)

sprintf(s,fmt[,expr-list]) formatiert die expr-list gemäß der Format
spezifikationen im fmt String und legt das Ergebnis
im String s ab statt es wie printf auf stdout auszugeben.

sub(r,s, [t]) ersetzt nur den ersten gefundenen r (RE) durch s (String) im Satz $0 oder in der angegebenen- t (Textvariablen)

substr(s,p,[n]) substr extrahiert aus dem String s ab der Position p n Zeichen oder den Rest des Strings.
substr("extrahieren",6,4) --> hier
substr("extrahieren",1,5) --> extra

toupper(s) liefert den String als Großbuchstaben zurück.
tolower(s) liefert den String als Kleinbuchstabenzurück.

Arithmetische Funktionen

atan2(y,x)	Arcus Tangens von y /x im Bereich -π bis π
cos(x)	Cosinus von x. x muß asl Bogenmaß übergeben werden
exp(x)	Exponentialfunktion von x
int(x)	Ganzzahligen Teil, abgeschnitten in Richtung Null
log(x)	Natürlicher Logarithmus von x
rand()	Zufallszahl im Bereich 0 bis 1
sin(x)	Sinus von x. x muß als Bogenmaß übergeben werden
sqrt(x)	Quadratwurzel von x
srand(x)	x ist ein neuer Startwert für die Funktion rand()

Zeitfunktionen

Da it dem awk häufig Log-Dateien ausgewertet werden die Zeitangaben (timestamps) enthalten, bietet der awk zwei Zeitfunktionen an.

systime() liefert die Zeit in Sekunden seit dem UNIX-Zero-Date
Midnight UTC, 1.1.1970 (UTC Universal Time Coordinate)

strftime(format,timestamp) formatiert eine Zeitmarke die als timestamp
vorliegt mit Hilfe eines Formatstrings und liefert einen
String zurück. Im Formatstring können alle Spezi-fikationen von `date` verwendet werden.

print strftime(" Datum : %d.%m.%y Zeit : %H:%M", systime())

Parameterübernahme ARGC, ARGV

ARGV[0] bis ARGV[ARGC-1] enthalten die übergebenen Parameter.
Achtung: Die optionalen Parameter (z.B. -v var=val -FS:) sind nicht in der
Parametertabelle enthalten.
ARGV[0] ist der Name des Programms (i.d.R. awk
ARGV[1] ist der erste Dateiname der hinter dem Script angegeben wird.