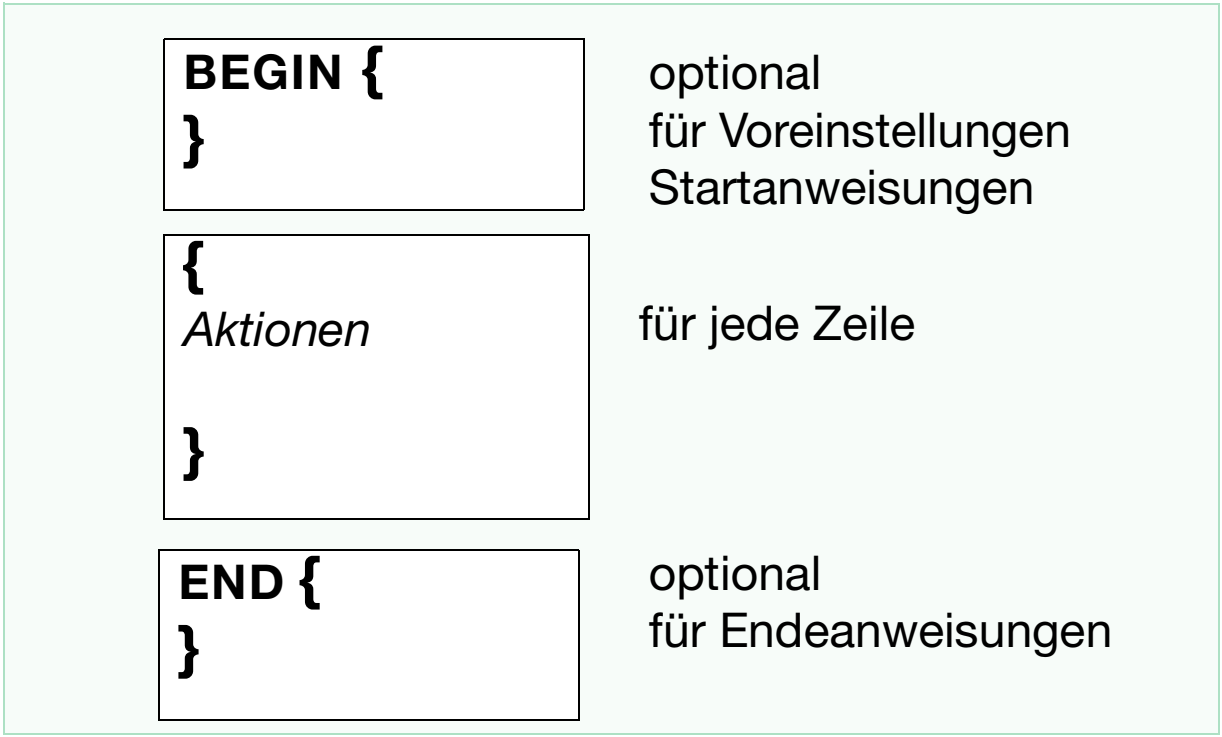
15 Der AWK

(Aho, Weinberger, Kernighan)

- Mächtiges Werkzeug zum Manipulieren von Dateien
  -> Reportgenerator
- Feldweise Bearbeitung von Dateien oder String - liest zeilenweise
- Suchen nach Text mit vorgebenen Muster (regulären Ausdrücken wie in grep)
- Aktionen für gefundene Muster
  - Ausgabe (**print** und **fprint**)
  - Ersetzen
  - Arithmetische Operationen **+, -, *, /**
    Funktionen wie exp(), cos(), atan(), log(), sqrt()
  - Funktionen für Zeichenketten
    length(), split(), subst(), index() ...
- Aktionen können Schleifen, bedingte Anweisungen, Variable
  und andere Operationen enthalten

Struktur des awk-Programms



awk 'BEGIN { Startanweisungen ..} *optional*
[Suchmuster] { … Aktionen .}.
END { … abschließende Anweisungen .}. *optional*
' *[Eingabedatei(en)]*

Variable im awk

Zeile aufgeteilt in Felder (Spalten) .

$1	$2	$3	$4	$5
Name	*Ort*	*Telefon*	*Umsatz*	*Vorjahr*
Albers	Hamburg	040-213344	30000	20000
Mayer	München	089-432678	60000	55000
Huber	München	089-915530	25000	22000

$0	gesamte Zeile
S1	1. Feld (Spalte)
$n	n. Feld (Spalte)
FS	Field Separator (wie Leerzeichen oder :)
NF	Number Fields (Anzahl der Felder je Zeile)
NR	aktuelle Zeilennummer des aktuellen Satzes
RS	Record Separator (Satztrennzeichen)

Beispiel einer Suche mit Ausgabe

Datei statistik:

$1	$2	$3	$4	$5
Albers	Hamburg	040-213344	30000	20000
Mayer	München	089-432678	60000	55000
Huber	München	089-915530	25000	22000

```
awk '/München/  { print $1 }'  statistik
```

Mayer

Huber

Beispiel vorab Setzen des Trennungszeichens (Fieldseparator)

```
sort /etc/passwd | awk '
BEGIN {
        FS=":"
}
# Main Loop (Aktionen)
{
print $1, $4,$5
}
'
```

Formatierte Druckausgabe

printf (%formatwert1 ["%formatwert2] [%formatwertn]", wert1,wert2,..wertn)

Beispiel: printf ("%-20s %10.2f\n", $7, $3/1000)

Formatwerte	Bedeutung
-	Formatierungskennzeichen optional linksbündige Ausrichtung
anzahl	max. Stellenanzahl
anzahl.anzahl	Nachkommstellen einer Gleitpunktzahl
s	String
f	Gleitkomma
d	Ganzahliger Wert mit Vorzeichen (decimal)
\n	Zeilenumbruch (new line)

Erstellen eigener Variable

Variable = Wert

```
#!/bin/ksh
#@(#) meindf V1.0 Anzeige von diskfree in MegaBytes
df -k | tail +2 | awk '
BEGIN {
        print "_____________________________________________"
        printf ("%-20s %10s\n","Dateisystem","Frei")
        freie_bytes = 0
        insgesamt = 0
}
# hier beginnen die Anweisungen pro Zeile
{
        freie_bytes = $4/1024
        ingesamt+= freie_bytes
        printf("%-20s %10.2f\n",$7, freie_bytes)
}
END {
print "_____________________________________________"
printf ("%.20s %10.2f Summe \n", "Summe:" , insgesamt)
}
'
```

Überprüfen Sie beim Testen vorab die Ausgabe von df -k (z.T. unterschiedlicher Aufbau je UNIX-Derivat)

AWK Konstante

- Numerische Konstante (
  - Ganze Zahlen 127,
  - Gleitpunktzahlen 1.2, .3, 23e2 (Angabe von + oder - erlaubt)
- Textkonstante durch mit " ... " geklammert
  - \" " im Text
  - \n neue Zeile
  - \t Tabulator
  - \\ das Zeichen \

AWK Ausdrücke

- Numerische Ausdrücke - Operatoren
  +, -, *, /, % (für Modulo)
- Textausdrücke verbinden durch Leerzeichen:
  "30 °" " Celsius"
- Logische Ausdrücke
  - Vergleichsoperatoren **<, <=, ==, !=, >=, >>**
  - logische Verknüpfungen
    && für UND (beide Operanten wahr)
    || für ODER (einer der Operanten wahr)
    ! Negation
  - Operatoren
    ~ für "ist enthalten in"
    !~ "ist nicht enthalten in"

Beispiel mit if-Abfrage

```
#!/usr/bin/ksh # # @(#) mydf2 V1.0 Verbessertes Disk-Free Command mit awk
#  Aufruf: mydf2
# df -k an awk "pipen". Achtung: Überschriftszeile von df muss raus:
df -k | tail +2 | awk '
BEGIN {
        print "----------------------------------------"
        printf ("%-20s %10s\n", "Dateisystem:", "MB Frei:")
        print "----------------------------------------"
        freie_bytes = 0
       summe = 0
}
# Main Loop
{      freie_bytes = $4/1024
        summe += freie_bytes
        if ( freie_bytes < 10 )
                printf("%-20s %10.2f *** Achtung \n", $6, freie_bytes)
         else printf("%-20s %10.2f\n", $6, freie_bytes)
}
END {
print "----------------------------------------"
printf("%-20s %10.2f\n", "Summe:", summe)
}
 '
```

AWK-Aktionen

- Weitere Zuweisungen von Variablen

Form	Bedeutung
variable += ausdruck	variable = variable + ausdruck
variable -= ausdruck	variable = variable - ausdruck
variable *= ausdruck	variable = variable * ausdruck
variable /= ausdruck	variable = variable / ausdruck
variable ++	variable =variable + 1
++variable	variable =variable + 1
variable --	variable =variable - 1
-- variable	variable =variable - 1
variable % ausdruck	variable = variable modulo ausdruck

Kontrollstrukturen innerhalb des AWK

- **if** (*bedingung*) *anweisung_1* **{** *anweisung_2* **}**
- **while** (*bedingung*) *anweisung*
- **for** (*ausdruck_1; bedingung; ausdruck2)* *anweisung*
- **for** (*variabel* **in** *feld*) *anweisung*

Zusätzlich

break	beendet eine for- oder while Schleife
continue	springt an das Ende einer Schleife
next	überspringt evtl. weitere Aktionen für die aktuelle Zeile geht zur nächsten Zeile
exit	Rest der Eingabedatei(en) (bzw. String) wird übersprungen END-Teil wird ausgeführt und awk beendet
return	Rücksprung aus einer awk-Funktion (Ergebniswert ausdruck)
#	Kommentar

Einige Funktionen innerhalb des AWK

Funktion	Bedeutung
gsub (*ra, neu, text*)	(global substitution) Zeichenkette "text" wird nach "ra" (regulärer Ausdruck) durchsucht und durch "neu" ersetzt
sub (*ra, neu, text*)	wie oben - doch Ersetzung nur beim 1. Vorkommen
index (*t1, t2)*	Sucht im Text "t1" nach Text "t2", liefert Anfangsposition zurück
length *(ausdruck)*	Liefert die Anzahl der Zeichen in Ausdruck zurück
split (*ausdr, bezeich, trenn*) **split** (*ausdr, bezeich*) *trenn entspricht FS*	Der Textausdruck "ausdr" wird entsprechend dem Trennzeichen "trenn" in Feldelemente zerlegt und dem Array "bezeich" zugewiesen. Die einzelnen Elemente können mit bezeich[1] bis bezeich[n] abgerufen werden
match (*t, ra)*	Die Zeichenkette "t" wird nach "ra" durchsucht und die Position des ersten Zeichens zurückgegeben
substr(*zk, m, n)* **substr**(*zk, m)*	Schneidet einen n-langen Text, beginnend beim m.ten Zeichen. Ohne "n" wird von m bis Ende ausgeschnitten

Einige Beispiele

ls -l | awk '/^d/ {print $9, $1}

awk '/Anfang/,/Ende/' info

awk '/^[0-9]+/ {print $1} ' info

Zählt alle Adressen aus München (Musterdatei statistik)

```
awk '
/München/ { count ++ }
END {
print "Anzahl der Münchner“, count }
' statistik
```

Gibt Anzahl der Leerzeilen aus:

```
awk '
BEGIN {N = 0}
{length == 0
N++ }
END { print N }
' text.dat
```

Literatur-Hinweise

- sed & awk, Dale Dougherty, Verlag O'Reilly, ISBN 0-937175-59-6
- UNIX, J. Gulbins, K. Obermayr, Springer, ISBN3-540-58864-7