Das Kommandozeilenwerkzeug AWK
Alfred v. Aho, Peter J. Weinberger und Brian W. Kernigham

Kathrin Kulmus

Technische Universität Chemnitz
Institut für Physik
THUS

10. Januar 2020

- **AWK** & **Shell** sind Skriptsprachen
- UNIX bietet viele Werkzeuge zur Datenverarbeitung:
  `sed, grep, paste, join, cut, diff, head, tail, sort, ...`
- in **UNIX-Shell** können diese zu Programmen kombiniert werden
  Glue-Language
- **AWK** ist vollständige Programmiersprache, die kaum auf dem UNIX Werkzeugkasten zurückgreift

Programm	**Skript**
wird *einmal* durch Compiler übersetzt	wird *jedesmal* durch Interpreter übersetzt

☞ **AWK**-Skript wird *jedesmal* in internen Byte-Code übersetzt

+ Wiederholt ausgeführter Code wird nur *einmal* übersetzt und dann sehr schnell ausgeführt
+ keine Syntaxfehler zur Laufzeit mehr möglich
– kleine Programme: Übersetzungszeit länger als Laufzeit

Allgemeine Eigenschaften

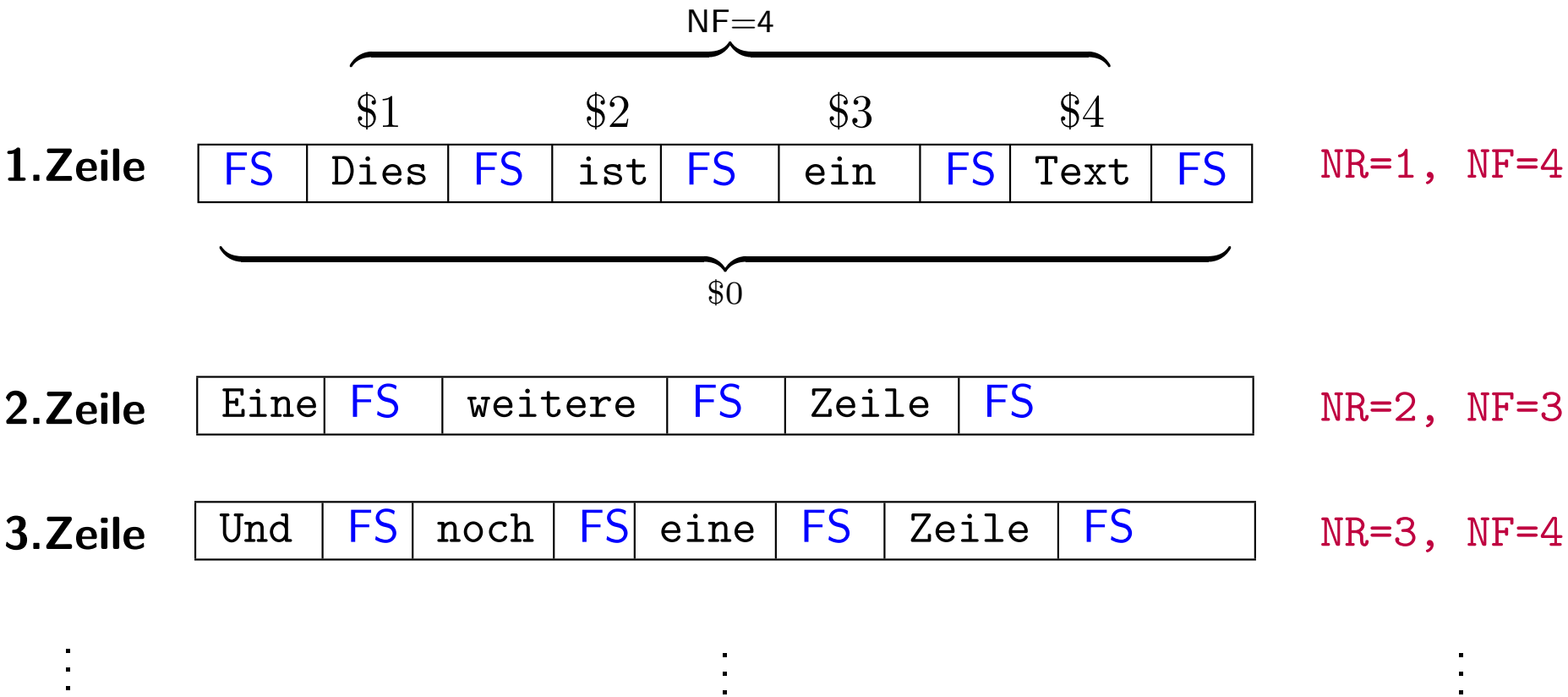
- fasst Fähigkeiten von UNIX-Tools zusammen
- Automatische Speicherverwaltung
- Verwandt mit C
- wenige Datentypen: Text oder Dezimalzahl
- Keine Variablen-Deklaration nötig
- kann Reguläre Ausdrücke

Typische Einsatzgebiete

- Datenauswertung und -aufbereitung
- Datentransformation, Texte umformatieren
- Datenvalidierung (z.B. Sytax, Semantik korrekt?)
- Prototyping und Realisierung von Programmen

```
awk   'BEGIN       { Aktion... }\
      Bedingung1   { Aktion... }\
      Bedingung2   { Aktion... }\
      .....
      BedingungN   { Aktion... }\
      END          { Aktion... }'\
file
```

- Interpreter ist **awk** „' '“ schützen vor Interpretation durch Shell
- `BEGIN`, i.e. vor Verarbeitung der ersten Eingabezeile und `END`, i.e. nach Verarbeitung der letzten Eingabezeile sind optional
- Aktionen stehen in „{ }“
- mehrere Aktionen durch „;“oder „\n“trennen


NF=4
$1
$2
$3
$4
1.Zeile
FS
Dies
FS
ist
FS
ein
FS
Text
FS
NR=1, NF=4
$0
2.Zeile
Eine
FS
weitere
FS
Zeile
FS
NR=2, NF=3
3.Zeile
Und
FS
noch
FS
eine
FS
Zeile
FS
NR=3, NF=4

Automatische Leseschleife:

0.) Datei Zeilenweise einlesen

1.) `BEGIN` Aktion ausführen

2.) Bedingungen der Reihe nach überprüfen
   - Bedingung für aktuelle Zeile ($\$0$) erfüllt $\rightarrow$ Aktion(en) ausführen

3.) alle Bedingungen geprüft
   - Nächste Zeile einlesen und Start bei 2.)

4.) `END` Aktion ausführen

✎ Keine Aktion: Zur Bedingung passende Zeilen werden ausgegeben

✎ Keine Bedingung: Aktion wird auf alle Zeilen angewandt

Operator	**Bedeutung**
(...)	Klammerung
$	Feldzugriff(`$1`,`$2`, ... , `$NF`)
++ --	Inkrement/Dekrement
^	Potenzierung x^y
+ - !	Unäres Plus/Minus, logisch **Nicht**
* / %	Multiplikation, Division, Modulo
+ -	Addition, Subtraktion
== != < (=) > (=)	Vergleich (Zahlen & Strings)
~ !~	Vergleich (Regex)
&& ||	logisches **und / oder**
= += -= *= /= %=	Operation + Zuweisung

- **Reguläre Ausdrücke** *regex* beschreiben eine Menge von Zeichenketten
- in **awk** durch „/`regex`/“ als Bedingung
- Folge von Nummern und/oder Buchstaben, Metazeichen, Zeichenlisten, Escape-Sequenzen
- Vorteil gegenüber Zeichenketten („foo“):
  - Zeichenketten schwieriger zu schreiben/lesen
  - *regex* in `awk` effizienter (wird intern ohne Umwandlung gespeichert)

regex	**Bedeutung**
\n \t \v	Zeilenumbruch, horizontaler/vertikaler Tabulator
\	Unterdrückt Sonderbedeutung: /\„Hallo\“/
^, $, .	Zeilenanfang, -ende, beliebiges Zeichen /^. $ /
[...], [^ ...]	positive/negative Zeichenliste
[:alnum:]	„A-Za-z0-9“
[:alpha:]	„A-Za-z“
[:lower:]	„a-z“
[:digit:]	„0-9“
\w, \W gawk	„[[:alnum:]_]“, „^[[:alnum:]_]“

Standartvariable	**Bedeutung**
`NF`	Anzahl der Felder
`NR`	Anzahl der Zeilen ingesamt
`FNR`	Nummer der aktuell bearbeiteten Zeile
`FS`	Feldtrenner („ “)
`RS`	Zeilentrenner („\n“)
`FILENAME`	aktueller Dateiname

Standartfunktionen	**Bedeutung**
`print` *LISTE*	*LISTE* ausgeben
`length(`*TEXT*`)`	Länge von *TEXT*
`substr(`*TEXT*`,` *START*`,` *LEN*`)`	Teilstück von *TEXT*
`sub(`*REGEX*`,`*TEXT*`,`*VAR*`)`	*REGEX* in *VAR* durch *TEXT* 1x ersetzen
`gsub(`*REGEX*`,`*TEXT*`,`*VAR*`)`	Analog aber global

man awk, man gawk, info awk, info gawk
https://www.ostc.de/awk.pdf
https://www-user.tu-chemnitz.de/~hot/unix_linux_werkzeugkasten/awk.html
https://linux-club.de/wiki/opensuse/Awk
http://mikiwiki.org/wiki/awk

Beispiele

Bearbeite das Gedicht:

1. Welche zeilen sind leer? Entferne alle Leerzeilen, speichere den Text in jandl2.txt
2. Spiele mit dem Feldtrenner: ändere den Standartwert, gibt die neue Anzahl der Felder und den Text ohne Feldtrenner aus.

   behandle das Gedicht als Einzeiler/Einspalter (Gib NF,NR als Test aus)
3. analysiere das Gedicht:
   Wie viele Wörter, Buchstaben, Zeichen, Zeilen?
4. Ersetze jedes o durch ein i.
5. Wie oft kommt das Wort mops vor?

1. Alle Spiele mit Auswärtssieg oder mehr als 30000 Zuschauern
2. Wo waren die meisten Zuschauer und wie viele?
3. Wo waren die Werbeeinnahmen je Zuschauer am geringsten, wie wenig?
4. Statistik:

   Es gab NUM Spiele.
   Zuschauer (gesamt) ± Änderung (gesamt): ANZAHL ± DIFF
   Zuschauer pro Spiel (Durchschnitt) ± Änderung (Durchschnitt): ANZAHL2 ± DIFF2
   Tore (gesamt): TORE
   Tore pro Spiel (Durchschnitt): TORE2