

Bearbeitung von Textfiles mit Bash Kommandos



Einführung

- Es existieren viele einfache Bash Kommandos, die - mit Filter (=Pipes) verknüpft - Textdateien manipulieren können.
- Durch die Kombination mehrerer solcher Bash Kommandos über Pipes bzw. E/A Umleitungen können viele Aufgaben zur Bearbeitung von Textfiles einfach gelöst werden.

Übersicht über Bash Kommandos zur Bearbeitung von Textfiles

- `cut`
- `sort`
- `uniq`
- `head`
- `tail`
- `wc`
- `sed`

cut-Kommando

- cut schneidet Spalten oder Felder aus Textfiles aus
- Die Spalten oder Felder sind ab 1 nummeriert, es können spezifische Trennzeichen verwendet werden
- Standard Trennzeichen ist <TAB>

```
$ cat names.txt
Emma Thomas:100:Marketing
Alex Jason:200:Sales
Madison Randy:300:Product Development
Sanjay Gupta:400:Support
Nisha Singh:500:Sales
```

-d legt das Trennsymbol fest, z.B. :
-f legt die Spalte fest z.B. -f 1 oder
-f 1,2

```
$ cut -d: -f 1 names.txt
Emma Thomas
Alex Jason
Madison Randy
Sanjay Gupta
Nisha Singh
```

Ausschneiden 1.Feld
Trennzeichen ist :

cut-Kommando Beispiele

- Schneide das 1. und 3. Feld aus dem Textfile, Trennzeichen wieder :

```
$ cut -d: -f 1,3 names.txt  
Emma Thomas:Marketing  
Alex Jason:Sales  
Madison Randy:Product Development  
Sanjay Gupta:Support  
Nisha Singh:Sales
```

Ausschneiden 1.und 3.Feld
Trennzeichen ist :

- Schneide einzelne Spalten aus dem Textfile

```
$ cut -c 1-8 names.txt  
Emma Tho  
Alex Jas  
Madison  
Sanjay G  
Nisha Si
```

Ausschneiden Spalten 1 bis 8
Trennzeichen keine Bedeutung !

cut- Kommando mit Pipes

- Das cut Kommando kann auch in Verbindung mit Pipes verwendet werden

- Beispiele:

- Ausschneiden der Berechtigungen für das VZ Documents im Homeverzeichnis:

```
ls -ld Documents/ | cut -c2-10
```

liefert: `rwxr-xr-x`

- Ausschneiden der Uhrzeit aus dem aktuellen Datum:

```
date | tr -s " " | cut -f4 -d " "
```

Anzeige Datum
und
Uhrzeit

Reduziere
Blanks zwischen
Feldern auf eins

Schneide 4.Feld aus, liefert
Uhrzeit

cut- Kommando mit Pipes

Folgende File pubs ist gegeben:

- [1] Schaffer, A.A., Aravind, L., Madden, T.L., Shavirin, S., Spouge, J.L., Wolf, Y.I., Koonin, E.V. and Altschul, S.F. (2001) **"Improving the accuracy of PSI-BLAST protein database searches with composition-based statistics and other refinements,"** Nucleic Acids Res. 29:2994-3005.
- [2] Yu, Y.-K., Wootton, J.C. and Altschul, S.F. (2003) **"The compositional adjustment of amino acid substitution matrices,"** Proc. Natl. Acad. Sci. USA 100:15688-15693.
- [3] Yu, Y.-K. and Altschul, S.F. (2005) **"The construction of amino acid substitution matrices for the comparison of proteins with non-standard compositions,"** Bioinformatics 21:902-911.
- [4] Altschul, S.F., Wootton, J.C., Gertz, E.M., Agarwala, R., Morgulis, A., Schaffer, A.A. and Yu, Y.-K. (2005) **"Protein database searches using compositionally adjusted substitution matrices,"** FEBS J 272(20):5101-9.

Schneide nur die Titel der Publikationen heraus:

```
cut -f2 -d\""\" pubs | cut -d\", \" -f1
```

Liefert:

```
Improving the accuracy of PSI-BLAST protein database searches with composition-based  
statistics and other refinements
```

```
The compositional adjustment of amino acid substitution matrices
```

```
The construction of amino acid substitution matrices for the comparison of proteins with  
non-standard compositions
```

```
Protein database searches using compositionally adjusted substitution matrices
```

cut- Kommando mit Pipes

File pubs :

- [1] **Schaffer, A.A., Aravind, L., Madden, T.L., Shavirin, S., Spouge, J.L., Wolf, Y.I., Koonin, E.V. and Altschul, S.F.** (2001) "Improving the accuracy of PSI-BLAST protein database searches with composition-based statistics and other refinements," *Nucleic Acids Res.* 29:2994-3005.
- [2] **Yu, Y.-K., Wootton, J.C. and Altschul, S.F.** (2003) "The compositional adjustment of amino acid substitution matrices," *Proc. Natl. Acad. Sci. USA* 100:15688-15693.
- [3] **Yu, Y.-K. and Altschul, S.F.** (2005) "The construction of amino acid substitution matrices for the comparison of proteins with non-standard compositions," *Bioinformatics* 21:902-911.
- [4] **Altschul, S.F., Wootton, J.C., Gertz, E.M., Agarwala, R., Morgulis, A., Schaffer, A.A. and Yu, Y.-K.** (2005) "Protein database searches using compositionally adjusted substitution matrices," *FEBS J* 272(20):5101-9.

Schneide nur die Autoren der Publikationen heraus:

```
cut -f1 -d"(" " pubs | cut -f2 -d"]" | tr -s " "
```

Liefert:

```
Schaffer, A.A., Aravind, L., Madden, T.L., Shavirin, S., Spouge, J.L., Wolf, Y.I., Koonin, E.V. and Altschul, S.F.  
Yu, Y.-K., Wootton, J.C. and Altschul, S.F.  
Yu, Y.-K. and Altschul, S.F.  
Altschul, S.F., Wootton, J.C., Gertz, E.M., Agarwala, R., Morgulis, A., Schaffer, A.A. and Yu, Y.-K.
```


sort - Kommando

- `sort` wird zum Sortieren von Textfiles benutzt
- `sort` verändert die angegeben Datei nicht und schreibt den sortierten File auf die Standardausgabe
- Durch E/A Umlenkung kann der sortierte File gespeichert werden:
- Beispiel:

```
sort myfile > sorted_myfile  
mv sorted_myfile myfile
```
- `sort` kann mit vielen Optionen verwendet werden, genaue Details siehe **man sort**

sort - Kommando Beispiele

- Sortiere `names.txt` aufsteigend

```
$ cat names.txt  
Emma Thomas:100:Marketing  
Alex Jason:200:Sales  
Madison Randy:300:Product Development  
Sanjay Gupta:400:Support  
Nisha Singh:500:Sales
```

```
$ sort names.txt  
Alex Jason:200:Sales  
Emma Thomas:100:Marketing  
Madison Randy:300:Product Development  
Nisha Singh:500:Sales  
Sanjay Gupta:400:Support
```

- Sortiere `names.txt` absteigend

```
$ cat names.txt  
Emma Thomas:100:Marketing  
Alex Jason:200:Sales  
Madison Randy:300:Product Development  
Sanjay Gupta:400:Support  
Nisha Singh:500:Sales
```

```
$ sort -r names.txt  
Sanjay Gupta:400:Support  
Nisha Singh:500:Sales  
Madison Randy:300:Product Development  
Emma Thomas:100:Marketing  
Alex Jason:200:Sales
```

sort – Kommando Beispiele

- Sortiere den File nach `employee-id` (Feld 2) und verwende den Delimiter (:) als Trennzeichen zwischen den Feldern des Files.

```
$ sort -t: -k 2n names.txt      -k 2n sortiert immer numerisch, ohne n vielleicht als string
Emma Thomas:100:Marketing
Alex Jason:200:Sales
Madison Randy:300:Product Development
Sanjay Gupta:400:Support
Nisha Singh:500:Sales
```

- Sortiere den File `/etc/passwd` numerisch nach der User-id (Feld 3)

```
$ sort -t: -k 3n /etc/passwd | more
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
```

sort – Kommando Beispiele


- Weitere Beispiele für die Verwendung von sort

```
ls -al | sort -k 5n
```

Sortiert das Long Listing eines Verzeichnisses numerisch nach der Größe der Files bzw. Verzeichnisse

- Zugriff auf einzelne Spalten eines bestimmten Feldes

```
ls -al | sort -k 1.1
```



Sortiert das Long Listing eines Verzeichnisses nach der ersten Spalte im ersten Feld (Typ des Datenobjekts)

- Viele weitere Optionen vhd. (man sort)

uniq - Kommando

- `uniq` wird meist in Kombination mit `sort` verwendet, um mehrfach vorkommende gleiche Zeilen auf eine zu reduzieren
- `uniq` funktioniert **nur** auf bereits sortierte Files, sonst werden Textzeilen mit gleichem Inhalt nicht gefunden.

nur unmittelbar folgende Werte verglichen

`uniq -c` gibt die Anzahl an, wie oft was vorgekommen ist

uniq – Kommando Beispiel

- Suche Zeilen mit gleichem Inhalt, reduziere die Anzahl zu 1 und gib zu jeder Zeile die gefundene Anzahl von Zeilen mit gleichen Inhalt aus. (Option -c)

```
$ cat namesd.txt
Alex Jason:200:Sales
Emma Thomas:100:Marketing
Madison Randy:300:Product Development
Emma Thomas:100:Marketing
Nisha Singh:500:Sales
Sanjay Gupta:400:Support
Alex Jason:200:Sales
```

```
$ sort namesd.txt | uniq -c
  2 Alex Jason:200:Sales
  2 Emma Thomas:100:Marketing
  1 Madison Randy:300:Product Development
  1 Nisha Singh:500:Sales
  1 Sanjay Gupta:400:Support
```

tail und head

- tail gibt - sofern nicht anders angegeben - die **letzten** 10 Zeilen einer Datei auf die Standard-Ausgabe.
- Mit der Option `-n1` werden statt 10 die letzten *n* Zeilen ausgegeben
- Beispiel

```
tail -51 /etc/passwd
```

Hier werden die letzten 5 Zeilen der Datei `/etc/passwd` ausgegeben

tail und head

- head gibt - sofern nicht anders angegeben - die **ersten** 10 Zeilen einer Datei auf die Standard-Ausgabe.
- Mit der Option `-n1` werden statt 10 die ersten *n* Zeilen ausgegeben
- Beispiel

L evtl nicht mehr gültig.
es reicht: -5 oder -n5

```
head -51 /etc/passwd
```

Hier werden die ersten 5 Zeilen der Datei `/etc/passwd` ausgegeben

WC

- `wc` - word count dient zum Zählen von Wörtern, Zeichen und Bytes in Textdateien

`wc OPTIONEN filename`

- Folgende Optionen sind möglich
 - l zählt die Zeilen in der Datei
 - c zählt die Bytes in der Datei
 - L gibt die Länge der längsten Zeile aus
 - w zählt die Worte in der Datei