

1.6.F Anwendung/Umleitung und Verkettung – Ergänzungen und Bilder

1.6.F.1 sort

Mit `sort` kann man Dateien und Datenströme sortieren. Gegeben ist folgende Geburtstagsliste eines Vereins (erste Spalte ist die Mitgliedsnummer):

```
1 97 Franz Meier 12.01.1987
2 68 Anton Lehmann 22.11.2000
3 112 Paul Meier 25.04.1993
4 48 Egon Schulz 21.03.1978
```

Diese Datei soll sortiert werden:

```
schueler@debian964:~$ sort gebtag.txt
112 Paul Meier 25.04.1993
48 Egon Schulz 21.03.1978
68 Anton Lehmann 22.11.2000
97 Franz Meier 12.01.1987
```

Es wird nach der ganzen Zeile sortiert, und zwar alphabetisch.

Die numerische Sortierung gelingt mit der Option `-n` (*numeric*):

```
schueler@debian964:~$ sort -n gebtag.txt
48 Egon Schulz 21.03.1978
68 Anton Lehmann 22.11.2000
97 Franz Meier 12.01.1987
112 Paul Meier 25.04.1993
```

Will man nach dem Nachnamen sortieren, muss man gezielt einen Suchschlüssel angeben. Dazu dient die Option `-k` (*key*):

```
schueler@debian964:~$ sort -k3,3 gebtag.txt
8 Anton Lehmann 22.11.2000
112 Paul Meier 25.04.1993
97 Franz Meier 12.01.1987
48 Egon Schulz 21.03.1978
```

Standardmäßig werden Leerzeichen und Tabs als Trenner gewertet, wobei mehrere davon wie ein einziger Trenner gelten.

Die Zählung der Schlüsselfelder beginnt mit der Nummer eins. Die beiden Zahlen geben einen Bereich von Schlüsselfeldern an, die vordere Zahl gibt das erste, die hintere Zahl das letzte Schlüsselfeld an. Hier wird also Feld 3 bis Feld 3 benutzt. Lässt man die zweite Zahl weg, werden alle Felder bis zum Zeilenende verwendet.

Da Paul und Franz den gleichen Nachnamen haben, braucht man noch einen Vornamen zur Sortierung. Er liegt aber in der Zeile vor dem Nachnamen. Deshalb kann man hier keinen Bereich nehmen. Stattdessen wird eine zweite Option `-k` für den Vornamen hinzugefügt:

```
schueler@debian964:~$ sort -k3,3 -k2,2 gebtag.txt
68 Anton Lehmann 22.11.2000
97 Franz Meier 12.01.1987
112 Paul Meier 25.04.1993
48 Egon Schulz 21.03.1978
```

Nun bleibt noch der Wunsch, die Liste nach Geburtstagen sortieren zu können, zuerst nach Monat und dann nach Tag. Dazu bietet `sort` die Möglichkeit, ein Schlüsselfeld nach Bytes zu unterteilen: `-k4.7,4.8` meint die Bytes Nr. 7 bis Nr. 8 des vierten Feldes, in unserem Fall das Jahrhundert. Eine 0 steht dabei für das letzte Bytes; `-k4.7,4.0` meint daher alle Bytes ab Nr. 7 des vierten Feldes, in diesem Fall die Jahreszahl. Führende Leerzeichen und Tabs gehören (leider) mit zum Schlüsselfeld. Das wird hier mit der Option `-b` (*-ignore-leading-blanks*) verhindert.

```
Terminal
schueler@debian964:~$ sort -k4.4b,4.5b -k4.1b,4.2b gebtag.txt
97 Franz Meier 12.01.1987
48 Egon Schulz 21.03.1978
112 Paul Meier 25.04.1993
68 Anton Lehmann 22.11.2000
```

Falls man Probleme mit Sortierschlüsseln hat, empfiehlt sich die Option `--debug`:

```
Terminal
schueler@debian964:~$ sort --debug -k4.4b,4.5b -k4.1b,4.2b gebtag.txt
sort: es werden die Sortierregeln für „de_DE.UTF-8“ verwendet
97 Franz Meier 12.01.1987
    _____
    _____
48 Egon Schulz 21.03.1978
    _____
    _____
112 Paul Meier 25.04.1993
    _____
    _____
68 Anton Lehmann 22.11.2000
    _____
    _____
```

Hier werden nach jeder Zeile zeilenweise der erste (Tag), der zweite (Monat) und der dritte (ganze Zeile) Schlüsselbereich unterstrichen.

Es gibt übrigens auch eine alte Schreibweise für Schlüsselfelder, die deutlich abweicht:

```
Terminal
schueler@debian964:~$ sort +1 -2 gebtag.txt
68 Anton Lehmann 22.11.2000
48 Egon Schulz 21.03.1978
97 Franz Meier 12.01.1987
112 Paul Meier 25.04.1993
```

Hier beginnt der Schlüsselbereich ab Feld 1 (beginnend mit 0!) und endet *vor* Feld 2.

Mit der Option `-t` (*separator*) gibt man an, welches Zeichen ein Trennzeichen sein soll. Dann allerdings ist es nicht mehr erlaubt, mehrere Leerzeichen als einen einzigen Trenner zu verwenden. Stattdessen sind so auch leere Felder möglich.

```
Terminal
schueler@debian964:~$ sort -t ":" -k3,3n /etc/passwd|head -1
root:x:0:0:root:/root:/bin/bash
```

Weitere wichtige Optionen für `sort` zeigt Tabelle 1.

Option	Bedeutung
-r	Rückwärts sortieren (<i>reverse</i>)
-f	Groß- und Kleinschreibung ignorieren (<i>fold</i>)
-d	Nur Buchstaben, Ziffern, Leerzeichen und Tabs ansehen (<i>dictionary-order</i>)
-g	Numerisch sortieren, auch Gleitkommazahlen (<i>general-numeric</i>)
-z	Jede Eingabezeile endet mit \0 (<i>zero-terminated</i>)

Tabelle 1: Weitere Optionen für `sort`

1.6.F.2 Probleme mit `sort` und `join`

`join` benötigt stets sortierte Eingabedateien, aber die von `sort` gelieferte sortierte Reihenfolge passt manchmal nicht. Woran liegt das? In

<http://unix.stackexchange.com/questions/12942/join-file-2-not-in-sorted-order> findet man einen Hinweis: "*Sort by default uses the entire line as the key; join uses only the specified field as the key.*"

`sort` sortiert standardmäßig so, dass die ganze Eingabezeile als Schlüssel verwendet wird (Tabelle 2). Dabei werden leer- oder Interpunktionsfelder nicht mitgezählt.

Zeile	<code>sort</code> liest
100:x:y	100xy
101:x:y	101xy
1:x:y	1xy
200:x:y	200xy
29:x:y	29xy
2:x:y	2xy

Tabelle 2: Wie `sort` standardmäßig sortiert

`join` erwartet dagegen, dass nur nach dem Schlüsselfeld sortiert wurde (Tabelle 3). Es gibt

Zeile	<code>join</code> liest
1:x:y	1
100:x:y	100
101:x:y	101
2:x:y	2
29:x:y	29
200:x:y	200

Tabelle 3: Was `join` standardmäßig einliest

zwei Wege, das Problem zu lösen:

- Simple Abhilfe: Numerische Werte im Schlüsselfeld sind mit führenden Nullen aufzufüllen (Tabelle 4). Nachteil: Das hilft nur in diesem Sonderfall, nicht aber bei anderen Arten von Schlüsselfeldern, z. B. Nachnamen als Schlüsselfeldern. Z. B. heißt ein Nachname Müller, einer heißt Müller-Lüdenscheid; schon ist bei `sort` die Sortierreihenfolge vom Inhalt des nächsten Feldes abhängig, bei `join` dagegen nicht.
- Nachhaltige Abhilfe: Bei `sort` immer das zu sortierende Feld *explizit* angeben:

```

Terminal
schueler@debian964:~$ sort -t ":" -k1,1b < abc.txt > abc.srt
schueler@debian964:~$ join -t ":" abc.srt anderedatei.srt

```

Zeile	sort liest	join liest
001:x:y	001xy	001
002:x:y	002xy	002
029:x:y	029xy	029
100:x:y	100xy	100
usw.		

Tabelle 4: Wie man sort und join zusammenbringt

1.6.F.3 Hintergrund-Programme ohne Ausloggen

Normalerweise werden beim Ausloggen alle Prozesse der aktuellen Shell geschlossen – auch die, die im Hintergrund laufen. Das passiert durch das Signal SIGHUP (=hangup, Nr. 1). Um dies zu vermeiden, kann man einen Hintergrundprozess mit einem Programm aufrufen, das dieses Signal abfängt. Dazu gibt es folgende Programme:

- nohup – für neue Prozesse
- disown -h %1 – für laufende Prozesse, eingebauter Shell-Befehl
- screen – kann mehr
- tmux – kann mehr

Ein Beispiel für das Programm xeyes (funktioniert nur mit GUI oder SSH-Tunnel und Xming):

```
Terminal
schueler@debian964:~$ xeyes &
schueler@debian964:~$ exit
```

Jetzt verschwindet das Fenster von xeyes.

```
Terminal
schueler@debian964:~$ nohup xeyes &
schueler@debian964:~$ exit
```

Jetzt bleibt das Fenster von xeyes weiterhin sichtbar. Übrigens fangen manche Programme das Signal SIGHUP schon von selbst ab, so dass die Hilfe durch nohup gar nicht brauchen; es schadet dann aber nichts.