
4 Fortgeschrittene Textfilter

In diesem Kapitel lernen Sie



- ▶ die Filterprogramme **expand**, **fmt**, **join**, **nl**, **paste**, **split**, **od**, **pr** und **tac** zu benutzen (LPI 1: 103.2).

In diesem Abschnitt werden folgende Textverarbeitungsprogramme als bereits bekannt vorausgesetzt:

- **cat** zum Aneinanderhängen/Ausgeben/Anlegen von Dateien
- **grep** zum Durchsuchen von Dateien
- **cut** zum Ausschneiden von Textteilen/-spalten
- **sort** zum Sortieren der Zeilen einer Datei
- **head** und **tail** zur Anzeige von Dateianfang bzw. -Ende
- **tr** zur zeichenweisen Ersetzung
- **wc** zum Zählen von Zeichen, Zeilen oder Wörtern in einer Datei

4.1 Übersicht

Die Grundphilosophie von Unix-Tools ist: „Do one thing and do it well“. Damit ist gemeint, dass jedes Tool nur eine ganz einfache Aufgabe erledigt, und diese in vielen denkbaren Variationen und effizient.

Dann kann man mit Hilfe der Shell diese Tools bausteinartig zu einer größeren Lösung zusammensetzen, die in herkömmlichen Programmiersprachen viele Seiten Quellcode lang wäre, und als Folge von Shellbefehlen nur einige wenige Zeilen umfasst.

Am Ende dieses Abschnitts werden wir ein Beispiel kennenlernen, in dem wir, nur in einer (wenn auch längeren) Zeile Programmtext, eine Hitliste der in einem Text am häufigsten vorkommenden Wörter, zusammen mit der Anzahl der vorkommenden Wörter erstellen.

Wichtig ist zunächst, dass man weiß, welches Tool welche Aufgabe erledigt. In der Praxis liest man die Details einfach in der Man-Page nach. Für die Prüfung jedoch sollte man im Idealfall die für die Praxis wichtigsten Optionen wissen. Im Notfall jedoch wird einem nichts anderes übrig bleiben, als zu raten.


Fortgeschrittene Textfilter

Zuerst also eine Übersicht darüber, welches Tool welche Aufgabe erledigt:

expand	ersetzt alle Tabulatorzeichen im Text durch eine Anzahl Leerzeichen.
fmt	ist ein simpler Textformatierer, der ASCII-Text auf eine vorgegebene Textbreite hin umbricht.
paste	ist das Gegenteil zu cut . Während man mit cut aus einer tabellenartigen Datei Spalten ausschneiden kann, kann man mit paste die Zeilen einer oder mehrerer Dateien aneinanderhängen.
join	verknüpft zwei tabellenartige Dateien anhand eines gemeinsamen Feldes zu einer kombinierten Tabelle.
nl	(number lines) fügt Zeilennummerierung hinzu.
split	ist das Gegenteil zu cat und teilt eine Datei in gleich große Portionen auf. Empfehlenswert, um große Dateien auf mehrere Disketten zu verteilen.
od	(octal dump) formatiert eine Datei oder die Standardeingabe im Oktal-, Dezimal-, Hexadezimal- oder ASCII-Zeichen-Format. Nützlich, um Binär-Dateien in lesbarer Form darzustellen.
pr	formatiert ASCII-Text in Seiten und/oder Spalten zum Ausdruck.
tac	kehrt die Zeilen-Reihenfolge in einer Datei um. (tac = cat rückwärts)

4.2 Die Textfilter im Detail


4.2.1 **expand** — Tabulatoren durch Leerzeichen ersetzen

Das Kommando **expand** (LPI 1: 103.2) ersetzt Tabulatorzeichen im Text durch eine Reihe von Leerzeichen (standardmäßig 8 Leerzeichen). Alternativ lassen sich für jedes Tabulatorzeichen in einer Zeile auch unterschiedliche Werte angeben. Das Ergebnis des Kommandos wird auf die Standardausgabe geschrieben. Als Eingabe dienen eine oder mehrere Dateien oder die Standardeingabe, falls ein Minuszeichen (-) als Eingabedatei angegeben ist. 

Syntax:

```
expand [Optionen] [Datei1,...]
```

Beispiele:

Alle Tabulatoren in der Datei `Datei1.txt` durch 8 Leerzeichen ersetzen: 

```
$ expand Datei1.txt
```

Sie können statt dessen auch die Positionen der einzelnen Tabulatoren in einer Liste angeben:

```
cat > Datei1.txt
↵ a ↵ b ↵ c
Strg-D
$ echo 0123456789; expand -t 2,5,9 Datei1.txt
0123456789
  a b c
```

4.2.2 unexpand — Leerzeichen durch Tabulatoren ersetzen

- Ⓛ Das Kommando **unexpand** (LPI 1: 103.2) ersetzt eine Reihe von Leerzeichen am Zeilenanfang durch Tabulatorzeichen. Es werden standardmäßig acht Leerzeichen durch ein Tabulatorzeichen ersetzt, ansonsten benutze man die Option `-t`).

Das Ergebnis des Kommandos wird auf die Standardausgabe geschrieben. Als Eingabe dienen eine oder mehrere Dateien oder die Standardeingabe, falls ein Minuszeichen (`-`) als Eingabedatei angegeben ist.

Syntax:

```
unexpand [Optionen] [Datei1,...]
```

📎 *Beispiele:* Wir legen eine Datei an und wenden **unexpand** darauf an:

```
$ cat > Datei1.txt
$ ██████████A██████████B██████████C
Strg-D
$ od -a Datei2.txt

0000000 sp sp sp sp sp sp sp sp A sp sp sp sp sp sp sp
0000020 sp B sp sp sp sp sp sp sp sp sp C nl

$ unexpand Datei1.txt | od -a

0000000 ht A sp sp sp sp sp sp sp sp B sp sp sp sp sp sp
0000020 sp sp sp C nl
```

Fortgeschrittene Textfilter

Wie man sieht, werden nur die Leerzeichen am Zeilenanfang in Tabulatoren umgewandelt.

Mit der Option `-a` werden alle Leerzeichen in Tabulatoren umgewandelt:

```
$ unexpand -a Datei1.txt | od -a
0000000 ht  A ht sp  B ht sp sp  C nl
```

Dieses Kommando hat die selbe Wirkung wie:

```
$ unexpand -t8 Datei1.txt | od -a
0000000 ht A ht sp B  ht sp sp C nl
```

Verkleinert man die Tabulatorgröße, so erhält man:

```
$ unexpand -t4 Datei1.txt | od -a
0000000 ht ht  A ht ht sp  B ht ht sp sp  C nl
```

Bemerkenswert sind in den letzten drei Beispielen die zusätzlichen Leerzeichen (`sp`). Das kommt daher, dass `unexpand` die Tab-Positionen bei 8, 16, 24, ... hält (von Null an gezählt!). A befindet sich daher auf Position acht, B auf Position 17 und C auf Position 26.

Wollen wir die zusätzlichen Leerzeichen in unserem Beispiel eliminieren, so müssen die Tabulatorpositionen explizit angegeben werden:

```
$ unexpand -t8,17,26 Datei2.txt | od -a
0000000 ht A ht  B ht C nl
```

4.2.3 `fmt` — Text formatieren

Um Text in einer vorgegebenen Breite absatzweise zu formatieren, verwendet man das Kommando `fmt` (LPI 1: 103.2):



Syntax:

```
fmt [-Zahl] [weitere Optionen] [Datei,...]
```

Mit der Option `-wZahl` oder alternativ mit `-Zahl` kann man die Textbreite in Zeichen vorgeben. Die weiteren Optionen sind für spezielle Formatierungswünsche notwendig. Man lese Sie deshalb im Bedarfsfall in der Man-Page nach.

Beispiel:



```
$ fmt -w30
```

Das Kommando `fmt` formatiert normalen ASCII-Text in eine fest vorgegebene Breite um. Dazu kann man mit der Option `-w` die Breite einstellen.

Das Kommando `fmt` formatiert normalen ASCII-Text in eine fest vorgegebene Breite um. Dazu kann man mit der Option `-w` die Breite einstellen.

Im angegebenen Beispiel wurde eine Breite von 30 Textzeichen eingestellt. Bei einer Leerzeile wird ein neuer Absatz angefangen.

Im angegebenen Beispiel wurde eine Breite von 30 Textzeichen eingestellt. Bei einer Leerzeile wird ein neuer Absatz angefangen.

4.2.4 `join` — Tabellen verknüpfen

- Ⓔ Da viele Konfigurationsdateien unter UNIX/Linux in Form von ASCII-Dateien in Tabellenform vorhanden sind, kann man mit `join` (LPI 1: 103.2) gemeinsame Felder solcher Tabellen miteinander verknüpfen.

Syntax:

```
join [-tTrennzeichen] [-1 Spalte Datei 1] \  
      [-2 Spalte Datei 2] [-o Datei.Spalte ...] \  
      Datei1 Datei2
```

Mit der Option `-t` wird das Tabellen-Feldtrennzeichen angegeben. Meistens ist das ein Doppelpunkt, ein Tabulatorzeichen oder ein Komma.

Die Optionen `-1` und `-2` legen fest, welche Spalte aus der ersten Datei mit welcher Spalte aus der zweiten Datei verknüpft werden soll.

Fortgeschrittene Textfilter

Die Option `-o` legt die Spalten in der Ausgabe fest. Es wird eine durch Leerzeichen getrennte Liste von Datei-/Spaltenbezeichnern erwartet, die angibt, aus welcher Datei (1 oder 2) welche Spalte (Spaltennummer) ausgegeben werden soll.

Datei 1 und *Datei 2* sind die zwei Tabellendateien, deren Spalten miteinander verknüpft werden sollen. Wird anstatt eines Dateinamens ein Minuszeichen (-) angegeben, so steht das für die Standardeingabe.

Beispiel:



Die Dateien `/etc/passwd` und `/etc/group` stellen die Benutzer/Gruppendatenbank unter Linux dar. Sie sind in Form einer ASCII-Tabelle mit dem Doppelpunkt als Feldtrenner abgelegt.

In der Datei `/etc/passwd` ist im Feld 1 der Benutzername, und im Feld 4 die Gruppennummer der Standardgruppe des Benutzers gespeichert. In der Datei `/etc/group` ist im Feld 1 der Gruppenname und im Feld 3 die zugehörige Gruppennummer abgelegt.

Wir wollen nun eine Tabelle darstellen, bei der alle Benutzernamen und alle zugehörigen Gruppennamen dargestellt werden.

Wir verknüpfen daher aus Datei 1 (`/etc/passwd`) das Feld 4 (die Gruppennummer des Benutzers) mit dem Feld 3 aus Datei 2 (`/etc/group`), das die Gruppennummer der entsprechenden Gruppe enthält.

Ausgegeben (`-o`) werden soll aus der Datei 1 das erste Feld (der Benutzername) und aus Datei 2 das zur Gruppennummer aus Datei 1 zugehörige erste Feld (nämlich der Gruppenname).

```
$ join -t: -1 4 -2 3 -o 1.1 2.1 /etc/passwd /etc/group
root:root
daemon:daemon
bin:bin
sys:sys
lp:lp
...
joe:autoren
...
```

Achtung: Die Dateien sollten bereits nach dem gemeinsamen Feld sortiert sein und möglichst keine Zeilen enthalten, die keiner Zeile in der anderen Datei zugeordnet werden können, sonst kann es zu unerwarteten Ergebnissen kommen.



4.2.5 nl — Zeilen nummerieren

- Ⓛ Mit **nl** kann man jedem beliebigen ASCII-Text eine Zeilennummerierung hinzufügen. Das Kommando **nl** (LPI 1: 103.2) beherrscht eine Menge von Formatierungsoptionen, von denen hier nur einige grundlegende beschrieben werden können.

Syntax:

```
nl [Optionen] [Dateien]
```

Das Kommando **nl** unterscheidet zwischen Kopf- und Fußzeilennummerierung und beherrscht auch die Nummerierung von logischen Seiten, deren Seitentrenner frei definierbar ist. Es würde den Rahmen dieser Unterlage sprengen, dies alles im Detail zu behandeln, deswegen wollen wir die wichtigste Option betrachten:

-bpRegulärer Ausdruck (body parse) Nummeriert nur Zeilen in denen der reguläre Ausdruck vorkommt



Beispiele:

Normale Zeilennummerierung:

```
$ nl
Dies ist die erste Zeile
  1 Dies ist die erste Zeile
Dies ist die zweite Zeile
  2 Dies ist die zweite Zeile
Dies ist die dritte Zeile
  3 Dies ist die dritte Zeile
```

Alle Zeilen im Textkörper einer Man-Page nummerieren (= alle Zeilen, die mit mindestens einem Tabulator- oder Leerzeichen beginnen):

```
$ man nl | nl -b p'^[ \t]+'
NL(1)                               FSF                               NL(1)

NAME
1      nl - number lines of files

SYNOPSIS
2      nl [OPTION]... [FILE]...

DESCRIPTION
```

Fortgeschrittene Textfilter

```
3      Write each FILE to standard output, with line numbers
4      added. With no FILE, or when FILE is -, read standard
5      input.


6      -b, --body-numbering=STYLE
7          use STYLE for numbering body lines

8      -d, --section-delimiter=CC
9          use CC for separating logical pages
...

```

Im dargestellten Beispiel werden in der Man-Page von **n1** alle Zeilen, die mit mindestens einem (+) Leer- oder Tabulatorzeichen ([\t]) beginnen (^), nummeriert. Die Syntax der erweiterten regulären Ausdrücke entspricht der von **egrep**.

4.2.6 paste — Tabellenspalten aneinanderfügen


Das Kommando **paste** ist das Gegenstück zu **cut**. Während **cut** Spalten ausschneidet, fügt **paste** (LPI 1: 103.2) die Zeilen mehrerer Dateien spaltenweise zusammen. 

Syntax:

```
paste [-dFeldtrenner] [-s] [Datei1 ...]
```

Mit **-d** lässt sich wie oben das Feldtrennzeichen festlegen. Vorgabewert ist ein Tabulatorzeichen.

Die Option **-s** (seriell) bewirkt, dass alle Zeilen einer Datei zu der ersten Spalte der Ausgabe werden. Damit vertauscht man Zeilen und Spalten.

Beispiel: 

```
$ cat > datei1
1
2
3
Strg-D
$ cat > datei2
eins
zwei
drei
Strg-D
```

```

$ cat > datei3
Oans
Zwoa
Drei
Strg-D
$ paste datei1 datei2 datei3
1      eins   Oans
2      zwei   Zwoa
3      drei   Drei
$ paste -d: datei1 datei2 datei3
1:eins:Oans
2:zwei:Zwoa
3:drei:Drei
$ paste -d: -s datei1 datei2 datei3
1:2:3
eins:zwei:drei
Oans:Zwoa:Drei

```

4.2.7 split — Datei aufteilen

- Ⓛ Das Kommando **split** (LPI 1: 103.2) ist das Gegenstück zu **cat**. Mit **split** kann man eine Datei in gleich große Teile zerlegen, etwa, um eine große Datei auf mehrere Disketten oder mehrere Mails zu verteilen. Mit **cat** lassen sich diese Stücke dann wieder zusammenfügen.

Syntax:

```
split [-b Größe[k|m|b]] [-l Zeilen] Datei [Präfix]
```

Das Kommando **split** kann wahlweise nach Zeilen (`-l=lines`) oder nach der Dateigröße der Teildateien (`-b=bytesize`) aufspalten. Die Dateigröße kann man in Bytes angeben, oder in Blöcken von 512, 1024 oder 1048576 Byte, indem man ein `b`, `k` oder `m` hinzufügt. Das optionale Präfix, das auf der Kommandozeile angegeben werden kann, wird allen Teildateinamen vorangestellt. Wird das Präfix weggelassen, so ist das Präfix lediglich ein `x`. Die Nummerierung der Spaldateien erfolgt alphabetisch.

Beispiel:

Wir erzeugen eine Datei mit der Größe von 3 MB, die nur aus Nullbytes besteht, und splitten sie in Portionen zu je 1400 kB auf, damit die Teile die Kapazität einer Diskette nicht überschreiten.

Fortgeschrittene Textfilter

```
$ dd if=/dev/zero of=Image bs=1024 count=3000
3000+0 records in
3000+0 records out
$ ls -l Image
-rw-r--r--  1 jack      skrdev    3072000 Jun 26 12:34 Image
$ split -b1400k Image Img.
$ ls Img* -l
-rw-r--r--  1 jack      skrdev    1433600 Jun 26 12:35 Img.aa
-rw-r--r--  1 jack      skrdev    1433600 Jun 26 12:35 Img.ab
-rw-r--r--  1 jack      skrdev     204800 Jun 26 12:35 Img.ac
```

Die so erzeugten Dateien können auf Disketten kopiert werden, und auf dem Zielsystem mit **cat** wieder zusammengesetzt werden:

```
$ cat Img.* > Image
$ ls -l Image
-rw-r--r--  1 jack      skrdev    3072000 Jun 26 16:22 Image
```

4.2.8 od — Oktal/Dezimal/Hex/ASCII-Dump

Mit dem Kommando **od** (LPI 1: 103.2) kann man Datenströme in Form von Oktal-, Dezimal- und Hexadezimalzahlen oder als ASCII-Zeichen ausgeben lassen. Es eignet sich damit sehr gut, um einzelne Bytes zu untersuchen, um damit Dateien und Dateiformate zu analysieren. Auch eignet es sich als Konvertierungstool zwischen der Binärdarstellung und der ASCII-Darstellung von Zahlen und Daten. Ⓛ

Syntax:

```
od [Optionen] [Datei(en)]
```