Textbearbeitung im Befehlsfenster

Achim Stein

December 19, 2015

Abstract

Hier wird erklärt, wie mit einigen Befehlen auf der Kommandozeile Text verändert werden kann (Umformatieren, Suchen und Ersetzen, Wortlisten). Dabei werden unterschiedliche Betriebssysteme berücksichtigt (Linux, Windows, Mac).

Diese Dokumentation wurde für eine Lehrveranstaltung erstellt. Sie wird nicht systematisch gepflegt und nur nach Bedarf erweitert.

Contents

1 Grundlagen: Shell und Arbeitsumgebung **2**

1.1 Befehlsfenster und Kommandozeile . . . 2

1.2 Erste Schritte in der Shell . . . 2

2 Die Befehle **4**

2.1 Grundlagen . . . 4

2.1.1 Befehlssyntax und Dateinamen . . . 4

2.1.2 Verketten, anzeigen, umleiten . . . 5

2.2 Einfache Textmanipulationen . . . 5

2.2.1 cat . . . 5

2.2.2 tr: einzelne Zeichen ersetzen/löschen . . . 6

2.2.3 sort . . . 6

2.2.4 uniq . . . 7

2.3 Weitere Befehle . . . 7

2.3.1 wc . . . 7

2.3.2 head / tail . . . 7

2.3.3 split . . . 7

2.3.4 cut . . . 7

2.3.5 join / paste . . . 8

2.4 Umgang mit Dateien . . . 9

2.4.1 ls: Verzeichnisse und Berechtigungen auflisten . . . 9

2.4.2 chmod: Berechtigungen ändern . . . 9

2.4.3 rm, mv: Löschen, Bewegen und Umbenennen . . . 10

2.4.4 zip und gzip: Komprimieren . . . 10

2.5 Komplexeres Suchen und Ersetzen . . . 11

2.5.1 grep: Zeilen suchen . . . 11

2.5.2 sed: Ersetzen von Zeichenketten . . . 11

1 Grundlagen: Shell und Arbeitsumgebung

1.1 Befehlsfenster und Kommandozeile

Jedes Betriebssystem kann über Kommandos gesteuert werden, die im Befehlsfenster eingegeben werden (man sagt auch "über die Kommandozeile" oder "in einer *Shell*"). Das Befehlsfenster ist über das Startmenü aufrufbar, oft unter einer Kategorie wie "Zubehör" oder "Dienstprogramme".

- Auf Linux, z.B. Ubuntu: Anwendungen–Zubehör–Terminal
- Auf Mac: Programme–Dienstprogramme-Terminal
- Auf Windows XP: Programme–Zubehör–Eingabeaufforderung

Unix und Windows: In dieser Anleitung werden Befehle benutzt, die auf Unix-Betriebssystemen vorinstalliert sind. Linux und Mac OS X sind Unix-Systeme und erfordern meist keine zusätzlichen Installationen. Sie verwenden in der Regel die sogenannte *Bash-Shell*.

Unter Windows können diese Befehle zwar auch als einzelne Programme nachinstalliert (und dann in der "Eingabeaufforderung" benutzt) werden, aber es ist einfacher und komfortabler, sich unter Windows ein komplettes "Unix-Terminal" zu beschaffen, also eine *Bash-Shell*. Die in dieser Anleitung beschriebenen Befehle können dann unter allen Systemen nahezu gleich eingegeben werden.

TIPP: für Windows: Cygwin installieren: http://www.cygwin.com. Dieses Paket ist kostenlos und enthält eine Reihe von Unix-Programmen für Windows.

Im folgenden wird das Befehlsfenster kurz *Shell* genannt.

1.2 Erste Schritte in der Shell

Grundlegende Befehle: In den Beschreibungen der Befehle stehen obligatorische Argumente zwischen Spitzklammern, z.B. `<Verzeichnisname>`, und Optionen zwischen eckigen Klammern, z.B. `[-a]`. Bei der Anwendung dürfen die Klammern nicht mitgeschrieben werden.

Befehl	Beispiel	Funktion
`exit`		Terminal beenden
`Ctrl-c`		laufenden Befehl unterbrechen
`mkdir <name>`	`mkdir meine_texte`	ein Verzeichnis anlegen
`cd <name>`	`cd meine_texte`	in ein (Unter)verzeichnis wechseln
`cd ..`	`cd meine_texte`	ins nächsthöhere Verzeichnis wechseln
`ls [-l] [<name>]`	`mkdir meine_texte`	Verzeichnisinhalt anzeigen (-l ausführlich)
`pwd`		Pfad des aktuellen Verzeichnisses anzeigen

Ein Arbeitsverzeichnis erstellen: Für die konkrete Arbeit an einem Text ist es empfehlenswert, ein eigenes Verzeichnis zu erstellen. In der Verzeichnisstruktur der Shell werden Pfadnamen durch einfachen Schrägstrich `/` getrennt, und nicht, wie in Windows, durch einen *Backslash* \.

Wenn das Befehlsfenster gestartet wird, befindet sich der Benutzer im *Home*-Verzeichnis. Die Shell symbolisiert es durch eine Tilde.

- Auf Unix-Systemen (Linux, Mac) ist dieses Verzeichnis das normale *Home*-Verzeichnis, in dem auch die anderen Benutzerdateien liegen.

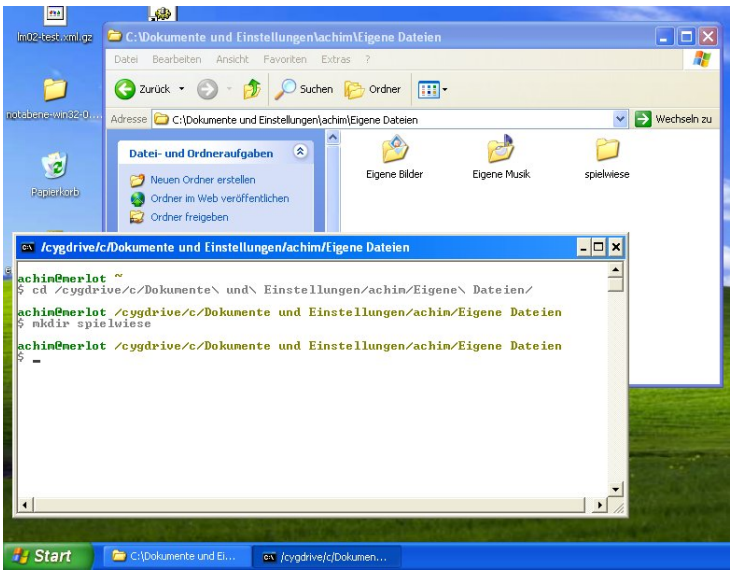

Figure 1: Cygwin in Windows XP: Arbeitsverzeichnis erstellen

- Unter Windows legt Cygwin eine eigene Verzeichnisstruktur `/home/...` für Benutzer an (z.B. `/home/achim`), die sich auf der Festplatte unter unter `C:\cygwin` befindet. Die meisten Benutzer werden aber ihr eigenes Windows-*Home*-Verzeichnis benutzen wollen. Nehmen wir an, der Benutzername ist `achim`. Dann kann man (unter Windows XP) mit
  - mit `cd /cygdrive/c/Dokumente\ und\ Einstellungen/achim/Desktop/` auf den Desktop
  - oder mit `cd /cygdrive/c/Dokumente\ und\ Einstellungen/achim/Eigene\ Dateien` in das Benutzerverzeichnis wechseln.

  `/cygdrive/c` ist also die Windows-Festplatte `C:`. Vor die Leerzeichen in den Pfadnamen muss ein Backslash gesetzt werden. Je nach Windows-Version befinden sich die Benutzerdateien an anderen Orten: unter Windows 7 wäre der Pfad `cd /cygdrive/c/Users/achim`.

Komfort in der Shell: Bei langen Datei- oder Pfadnamen macht man oft Tippfehler. Die Shell bietet eine Erleichterung über die Tabulatortaste (TAB) an: Tippen Sie den Anfang eines Verzeichnis- oder Dateinamens, drücken Sie TAB, und die Shell wird den Namen ergänzen (oder bei mehreren Möglichkeiten diese anzeigen). Tippen Sie z.B. `cd /cy`, dann TAB, und die Shell ergänzt zu `/cygdrive/`, usw.

Arbeitsverzeichnis: Im gewünschten Verzeichnis angekommen, sollte dann ein eigenes Verzeichnis angelegt werden, z.B. mit `mkdir spielwiese`. Es erscheint dann wie gewohnt im Dateimanager des Sytems, wie in Abb. 1 und 2 gezeigt (es könnte natürlich auch mit dem Dateimanager angelegt werden).

TIPP: In vielen Betriebssystemen zeigen die grafischen Dateimanager deutsche (oder der eingestellten Sprache entsprechende) Pfadnamen an, verwenden intern aber andere Namen. Mac OS X und Windows 7 zeigen ein Verzeichnis `/Benutzer` bzw. `C:\Benutzer` an, verwenden Intern aber `/Users` bzw. `C:\Users`. Analog für Schreibtisch bzw. Arbeitsplatz (Desktop) u.a. Die Shell zeigt immer die *internen* (englischen) Namen, und diese müssen auch bei Befehlen wie `cd` usw. eingegeben werden.

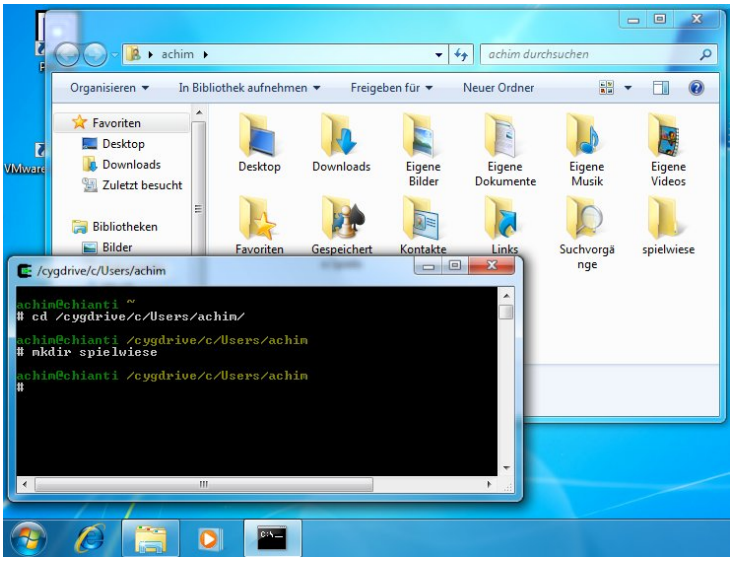

Figure 2: Cygwin in Windows 7: Arbeitsverzeichnis erstellen

Als Arbeitstext für die nächsten Schritte besorgen wir uns "Betrachtung" von Franz Kafka, der auf http://www.gutenberg.org frei herunterladbar ist. Wir laden ihn im Format "Plain Text" und speichern ihn im Verzeichnis `spielwiese` als `kafka.txt`.

2 Die Befehle

2.1 Grundlagen

Es schadet nicht, die hier erklärten Grundlagen vorher zu lesen. Praktische Beispiele folgen bei der Erklärung der Befehle, Praktiker werden dort anfangen und dann zurückblättern.

2.1.1 Befehlssyntax und Dateinamen

Befehlssyntax: Die meisten Befehle haben Optionen. Diese bestehen aus Bindestrich mit einem folgenden Buchstaben (Kurzform), manchmal auch zwei Bindestrichen mit einem Optionsnamen (Langform), und ggf. Argumenten. Danach folgt in der Regel der Dateiname. Beispiele s.u. bei 2.2.1.

Dateinamen sind Zeichenketten, z.B. `kafka.txt`. Sonderzeichen und Leerzeichen sind in den meisten Betriebssystemen möglich, von ihrer Verwendung ist aber abzuraten (Betriebssysteme kodieren die Sonderzeichen unterschiedlich, und bei der Übermittlung im Internet können sie ebenfalls verändert werden).

Viele Befehle können mehrere Dateien gleichzeitig aufrufen. Dafür können Platzhalter eingesetzt werden, die ähnlich wie beim Befehl `grep` funktionieren, nur der Platzhalter für 'irgendetwas' ist `*` (und nicht `.*`). Beispiele:

`*`	alle Dateien (im aktuellen Verzeichnis)
`a*`	alle Dateien, deren Name mit `a` beginnt
`*.txt`	alle Dateien mit Suffix `.txt` beginnt

2.1.2 Verketten, anzeigen, umleiten

Mit einer sog. *pipe* ('Röhre') können Befehle verkettet werden. Die *pipe* ist ein senkrechter Strich zwischen zwei Befehlen, also:

`befehl1 datei | befehl2`

Hier bearbeitet der erste Befehl die Datei. Das Ergebnis wird nicht angezeigt, sondern direkt vom zweiten Befehl verarbeitet. Man spart sich also die Erstellung einer Zwischendatei, die dann vom zweiten Befehl extra aufgerufen werden müsste.

Die Ausgabe eines Befehls wird normalerweise am Bildschirm angezeigt. Bei großen Dateien ist dies nicht praktisch. Es gibt zwei Möglichkeiten, die Ausgabe zu studieren:

Bei der Umleitung in eine Datei mit `>` wird die Datei im Verzeichnis angelegt und kann dann im Editor (oder mit `cat`) angezeigt werden:

`befehl > datei`

Vorsicht! Wenn schon eine Datei mit diesem Namen existiert, wird sie ohne Warnung überschrieben.

Die zweite Möglichkeit ist, die Datei Bildschirm für Bildschirm anzuzeigen, mit `less`:

`befehl | less`

Der erste Bildschirm wird angezeigt, unten steht ein Doppelpunkt für die Eingabe, dort kann mit der Leertaste weitergeblättert oder mit `q` abgebrochen werden.

TIPP: Mit `less` sind auch einfache Suchen möglich:

2.2 Einfache Textmanipulationen

Das erste Ziel ist, aus dem Beispieltext `kafka.txt` eine Frequenzliste zu machen. Dafür sind vier Befehle nötig.

2.2.1 cat

`cat` (*concatenate*) gibt eine oder mehrere Dateien hintereinander aus.

`cat kafka.txt` oder `cat *.txt`

Mit Option `-n` werden die Zeilen numeriert

`cat -n kafka.txt`

Mit einem Platzhalter und einer Umleitung können schnell z.B. alle Textdateien aneinandergehängt werden:

`cat *.txt > alle-texte`

TIPP: Hier sind natürlich nicht alle Möglichkeiten der Befehle erklärt. Im Internet gibt es gute Einführungen zu Unixbefehlen. Aber auch direkt in der Shell kann für fast jeden Befehl eine Hilfeseite mit `man` (*manual*) aufgerufen werden:

`man <befehl>`, z.B. `man cat`

Die Hilfeseite wird dann bildschirmweise angezeigt, wie mit `less` (s. 2.1.2).

2.2.2 tr: einzelne Zeichen ersetzen/löschen

`tr` (*translate*) übersetzt einzelne Zeichen, die in seinen beiden Argumenten angegeben werden. Wegen der beiden Argumente wird `tr` nicht mit Dateinamen aufgerufen, sondern wird an `cat` angehängt. Der folgende Aufruf ersetzt in der Beispieldatei alle *a* durch *b*:

```
cat kafka.txt | tr a b
```

In beiden Argumenten können mehrere Zeichen angegeben werden. Die Ersetzungspaare werden dann jeweils aus den ersten, zweiten usw. Zeichen gebildet. Die Großbuchstaben *A, B, C, D, E* werden mit folgendem Befehl durch Kleinbuchstaben ersetzt:

```
cat kafka.txt | tr ABCDE abcde
```

Außerdem können solche Reihen abgekürzt werden, etwa `[A-Z]` für alle Großbuchstaben, `[a-z]` für alle Kleinbuchstaben und `[0-9]` für alle Zahlen. Diese Ausdrücke sollten dann in Hochkommata stehen. Alle Großbuchstaben werden also durch folgenden Befehl klein geschrieben:

```
cat kafka.txt | tr '[A-Z]' '[a-z]'
```

Mit der Option `-d` (*delete*) werden die Zeichen des einzigen Arguments gelöscht. Der folgende Befehl löscht alle Satzzeichen:

```
cat kafka.txt | tr -d '[\.,;:!\?«»]'
```

Für die Frequenzliste sollten die Satzzeichen entfernt werden, aber darüberhinaus muss der Text in eine Wortliste verwandelt werden. Das heißt: Alle Leerzeichen werden durch Zeilenschaltungen ersetzt. Auch zur Markierung des Leerzeichens sind Hochkommata nötig, die Zeilenschaltung wird unter Unix als `\12` geschrieben. Die "Wortliste" (eigentlich nur ein vertikal Wort für Wort geschriebener Text) entsteht also durch:

```
cat kafka.txt | tr ' ' '\n'
```

2.2.3 sort

`sort` sortiert die Zeilen einer Datei alphabetisch. Angewendet auf die Wortliste ergibt dies die Befehlskette:

```
\verb#$ cat kafka.txt | tr ' ' '\n' ] sort
...
zwingen,
zwingt
zwinkerte,
zwischen
zwischen
zwischen
zwischen
$
```

Das Ende der Ausgabe zeigt uns erstens, dass wir die Satzzeichen noch herausnehmen sollten. Dafür hängen wir den Befehl von oben ein:

```
cat kafka.txt  | tr -d '[\.,;:!\?«»]' | tr ' ' '\n' ] sort
```

Zweitens sollen die gleichen Formen zusammengefasst und dabei gezählt werden. Für das Zusammenfassen würde die Option `sort -u` genügen. Wenn die Formen zusätzlich gezählt werden sollen, wird `uniq` benötigt.

2.2.4 uniq

Mit `uniq` werden gleiche Zeilen der Eingabe zusammengefasst, mit `uniq -c` werden die zusammengefassten Zeilen gezählt. Die resultierende Frequenzliste wird mit >in eine Datei umgeleitet.

```
cat kafka.txt  | tr -d '[\.,;:!\?«»]' | tr ' ' '\n' ] sort | uniq -c > kafka.frq
```

2.3 Weitere Befehle

2.3.1 wc

`wc` (*word count*) gibt die Anzahl der Zeilen, Wörter und Zeichen in einer oder mehreren Dateien aus, z.B.

```
$ wc kafka.txt
    1285    9022   58303 kafka.txt
```

Mit den Optionen `-l`, `-w` und `-c` kann die Ausgabe auf einen bestimmten dieser Werte beschränkt werden. Wenn mehrere Dateien aufgerufen werden, z.B. `wc -w *.txt` wird eine Liste aller Dateien mit dem/den entsprechenden Wert(en) erstellt.

2.3.2 head / tail

Manchmal sind Textdateien so groß, dass sie nicht in einen Editor geladen werden können. Mit diesen Befehlen kann man am Anfang (`head`) oder am Ende (`tail`) ein Stück abschneiden. Default sind 10 Zeilen, mit `-<zahl>` können mehr angegeben werden:

```
head -500 kafka.txt > anfang.txt
```

schreibt die ersten 500 Zeilen der Beispieldatei in eine neue Datei. Mit `tail -500` wären es die letzten 500 Zeilen. Die Kombination von beiden Befehlen erlaubt, einen Bereich aus der Mitte eines Textes herauszuschneiden, z.B. die Zeilen 213 bis 219:

```
head -219 kafka.txt | tail -7
```

2.3.3 split

Mit `split` lassen sich große Dateien in kleine zerteilen. Der Befehl

```
split -l 100 kafka.txt zerlegt-
```

kopiert jeweils 100 Zeilen aus der Beispieldatei in 13 Dateien, die `zerlegt-aa` bis `zerlegt-am` heißen.

2.3.4 cut

Etwas komplexer ist die Verarbeitung von Tabellenspalten. Hier möchten wir "vertikal" etwas aus der Datei herausschneiden, z.B. von jeder Zeile das dritte Wort. Typischerweise sind die Dateien Tabellen im Textformat, bei denen die Spalten durch Leerzeichen, Tabulatoren oder ein bestimmtes Zeichen getrennt sind.

Solche Dateien werden z.B. von Textverarbeitungsprogrammen wie *Excel* oder *OpenOffice Calc* erstellt, wenn man im Textformat (csv) exportiert. Aber auch viele sprachverarbeitende Programme benutzen Text-Tabellen, z.B. Tagger oder Parser. Unsere Beispieldatei enthält die Ausgabe eines Parsers, d.h. die syntaktische Analyse von Sätzen. Das Format unserer Beispieldatei `duby.conll` heißt CoNLL und ist z.B. hier erklärt: http://ufal.mff.cuni.cz/conll2009-st/task-description.html. Für unsere Aufgabe ist nur wichtig, dass die Datei 14 Spalten enthält, die durch Tabulatoren getrennt sind (wir stellen hier nur die ersten Spalten dar):

```
Nummer<tab>Wort<tab>Lemma1<tab>Lemma2<tab>Wortart<tab>...
```

Die Wörter des Textes stehen also untereinander, jeweils in der 2. Spalte der Zeile, und in den anderen Spalten stehen weitere Informationen.

TIPP: Um die Datei anzusehen, könnte man sie in einen Editor laden, oder in der Shell mit `head -20 duby.conll` die ersten Zeilen anzeigen lassen, oder eine Kopie namens `duby.csv` erstellen: auf den meisten Computern geht eine Tabellenkalkulation an, wenn man auf csv-Dateien klickt.

Wir möchten nun die Wörter extrahieren, also den Inhalt von Spalte 2 jeder Zeile. Hierfür genügt

```
cut -f2 duby.conll
```

weil das Trennzeichen der Tabulator ist. Andere Zeichen müssten mit Option `-d <Trenner>` (*delimiter*) angegeben werden, z.B. Strichpunkte oder Leerzeichen:

```
cut -d';' -f2 duby.conll
cut -d' ' -f2 duby.conll
```

Mit `-f` können auch mehrere Felder angegeben werden: Der nächste Befehl gibt die Spalten 1-3 und 5 aus (Nummer, Wort, Lemma, Wortart):

```
cut -f1-3,5 duby.conll
```

2.3.5 join / paste

Mit `paste` und `join` lassen sich Tabellen wieder zusammenfügen. Mit `cut` erstellen wir uns zwei Tabellen: `duby1` enthält Nummer, Wort, Lemma. `duby2` enthält Nummer, Wortart:

```
cut -f1-3,5 duby.conll > duby1
cut -f1,5 duby.conll > duby2
```

Entsprechend der Namen der Befehle taugt `paste` nur zum einfachen Zusammenkleben:

```
paste duby1 duby2
```

ergibt (wir zeigen nur 5 Zeilen der Ausgabe):

```
1       L'      le      1       DET:ART
2       office  office  2       NOM
3       royal   royal   3       ADJ
4       s'      se      4       PRO:PER
5       exerce  exercer 5       VER
```

Mit `join` können wir dagegen Zeilen zusammenführen, wenn ein bestimmtes Feld gleich ist. Wenn wir mit Option `-1 <spalte>`die Nummer in der ersten Spalte der ersten Datei als Kriterium für die Zusammenführung definieren

```
join -t$'\t' -1 1 duby1 duby2
```

erhalten wir:

```
1 L’ le DET:ART
2 office office NOM
3 royal royal ADJ
4 s’ se PRO:PER
5 exerce exercer VER
```

`join` nimmt als Default Leerzeichen als Trenner. Wenn wir wieder Tabulatoren haben möchten, geben wir das mit Option `-t '<Trenner>'` an. Dabei ist `\t` der Tabulator.

```
join -t'\t' -1 1 duby1 duby2
```

erhalten wir:

```
1       L’      le      DET:ART
2       office  office  NOM
3       royal   royal   ADJ
4       s’      se      PRO:PER
5       exerce  exercer VER
```

(Wir hätten auch mit Option `-2 1` die Nummer der 2. Datei nehmen können.)

TIPP: Falls `join` die Fehlermeldung *illegal character specification* ausgibt, hilft ein Dollarzeichen vor dem Argument von `-t`, also:

```
join -t$'\t' -1 1 duby1 duby2
```

2.4 Umgang mit Dateien

Diese Befehle sollten beherrscht werden, wenn man nicht von Dateiverwaltungsfenstern abhängig sein möchte (Arbeitsplatz, Finder usw.). Sie sind nur kurz erklärt.

2.4.1 ls: Verzeichnisse und Berechtigungen auflisten

`ls` zeigt den Inhalt des aktuellen Verzeichnisses an. `ls -1` zeigt nur die Dateinamen, `ls -l` eine ausführliche Liste mit Zugriffsrechten, Größe, Änderungsdatum und -zeit, z.B.:

```
-rw-------   1 achim  ilr  16512 19 Dez 16:48 duby1
-rw-r-----   1 achim  ilr   9516 19 Dez 16:51 duby2
-rw-rw-r--   1 achim  ilr  59101 19 Dez 15:46 kafka.txt
drwxr-x---  25 achim  ilr    850 19 Dez 16:35 latex2html
```

2.4.2 chmod: Berechtigungen ändern

Falsche Rechte (Berechtigungen) sind oft ein Grund von Fehlern. Daher hierzu eine Kurzanleitung. Oben sehen wir in der ersten Spalte eine Kombination aus 10 Zeichen. Sie bedeuten:

- `d`: directory (Verzeichnis)
- `r`: read (Leserecht)
- `w`: write (Schreibrecht)
- `x`: execute (Ausführungsrecht)
- `-`: nicht gesetzt

Zeichen 1 unterscheidet nur Verzeichnisse (`d`) von Dateien. Die folgenden 9 Zeichen können drei Mal `rwx` sein (oder eben nicht gesetzt). Sie geben für drei Benutzergruppen die Rechte an, also:

1: Verzeichnis
2-4: Rechte für Benutzer
5-7: Rechte für Gruppe
8-10: Rechte für andere

Im Beispiel oben sehen wir, dass die Datei `duby1` nur für den Benutzer *achim* lesbar und schreibbar (d.h. auch löschbar) ist. Die Datei `duby2` ist zusätzlich für die Gruppe lesbar. Und `kafka.txt` ist zusätzlich für die Gruppe schreibbar (löschbar) und für alle anderen lesbar.

Befehle und Verzeichnisse müssen zusätzlich mit `x` als ausführbar markiert sein. Das Verzeichnis `latex2html` kann vom Benutzer und von der Gruppe geöffnet (eingesehen) werden, aber nur der Benutzer kann neue Dateien darin erstellen (schreiben).

TIPP: Welche Rechte als Default für neue Dateien gesetzt werden und welche Gruppen es gibt definiert das System (oder die Systemverwaltung).

Zum Ändern der Rechte muss man Schreibrechte für die Datei haben `:-)` Das erste Argument von `chmod` gibt an, ob die direkt angehängten Rechte für *user* (`u`), *group* (`g`) oder *others* (`o`) gelten. Die Gruppe und alle anderen erhalten z.B. mit

```
chmod go+rw duby1
```

Lese- und Schreibrechte. Den 'anderen' können wir z.B. mit

```
chmod o-r duby1
```

das Schreibrecht wieder entziehen.

2.4.3 rm, mv: Löschen, Bewegen und Umbenennen

Löschen: `rm <datei>`
Umbenennen: `mv <datei> <neuer Name>`

Für beides sind Schreibrechte nötig.
WARNUNG: Beide Befehle können Dateien unwiderbringlich löschen, `rm` löscht direkt, und `mv` überschreibt Dateien, wenn der neue Name schon existiert. Diese Dateien sind dann **weg**, d.h. auch nicht mehr im Papierkorb des Systems zu finden! Bei beiden Befehlen sollte man Option `-i` verwenden: sie fragt vorher nochmal nach.

2.4.4 zip und gzip: Komprimieren

`zip <archiv> <dateien>` erstellt zip-Archive, d.h. fasst Dateien in einem komprimierten Archiv zusammen. `unzip <archiv>` packt solche Archive wieder aus. Dafür gibt es meist auch andere Möglichkeiten im Betriebssystem.

`gzip` komprimiert nur *eine* Datei, und hängt dabei `.gz` an den Dateinamen an. `gunzip` stellt die ursprüngliche Datei wieder her. Also:

`gzip kafka.txt` (erzeugt `kafka.txt.gz`)
`gunzip kafka.txt.gz` (stellt `kafka.txt` wieder her)

Bei großen Textdokumenten ist praktisch, dass man die Datei nicht dekomprimieren muss, um sie zu verarbeiten. Mit Option `-c` wird der Text der komprimierten Datei am Bildschirm ausgegeben.

Nützlicher ist, dass er dann direkt mit anderen Programmen weiterverarbeitet werden kann, z.B. mit `tail`, um die letzten 50 Zeilen anzuzeigen:

```
gunzip -c kafka.txt.gz | tail -50
```

Auf ähnliche Weise kann `gzip` die Ausgabe einer Datei direkt in eine komprimierte Datei schreiben. Der folgende Befehl packt die Beispieldatei aus, nummeriert die Zeilen mit `cat -n` und packt sie wieder ein:

```
gunzip -c kafka.txt.gz | cat -n | gzip > kafka-nummeriert.txt.gz
```

2.5 Komplexeres Suchen und Ersetzen

2.5.1 grep: Zeilen suchen

Das Programm `grep` ist ein Suchprogramm, das eine Datei nach einem Suchausdruck durchsucht und alle Zeilen ausgibt, die auf diesen Ausdruck passen. Dies Syntax ist

```
grep [Optionen] 'Suchausdruck' Datei
```

Die Suche nach "Garten" im Kafka-Text gibt drei Zeilen aus:

```
$ grep 'Garten' kafka.txt
Ich hörte die Wagen an dem Gartengitter vorüberfahren, manchmal sah ich
den Bäumen im Garten meiner Eltern.
öffne und daß in einem Garten die Musik noch spielt.
```

Sehr vielseitig ist `grep` durch die Möglichkeit, sogenannte *reguläre Ausdrücke* zu benutzen. Das sind Ausdrücke mit Platzhaltern.

Befehl	Bedeutung	Suchausdruck	Findet...
`.`	beliebiges Zeichen	b.ten	*baten, beten, boten*
`+`	vorhergehendes Zeichen min. einmal	be*ten	*beten, beeten, beeeten, ...*
`*`	vorhergehendes Zeichen beliebig oft	be*ten	*bten, beten, beeten, beeeten, ...*
also: `.*`	beliebige Zeichen	b.*ten	*beten, bluten, bearbeiten*
`[ ]`	mögliche Zeichen an einer Position	b[eo]ten	*beten, boten*, aber nicht *buten*

TIPP: Diese Tabelle ist nicht vollständig. Nützlich sind z.B. die Abkürzungen für bestimmte Zeichentypen (`\w` für alphabetische Zeichen, `\s` für Leerzeichen, Tabs usw.). Reguläre Ausdrücke können sehr komplex sein.

2.5.2 sed: Ersetzen von Zeichenketten

ein Leerzeichen vor jedes Satzzeichen stellen:

```
cat kafka.txt | sed -e 's/\([\.,;:!\?«]\)/ \1/g'
```

Um auch die Anführungszeichen zu berücksichtigen, müssen zwei Ersetzungsausdrücke kombiniert werden. Man könnte zwei sed-Befehle aneinanderhängen, aber effektiver ist es, sed nur einmal aufzurufen und zwei Ausdrücke jeweils mit der Option `-e` anzugeben. Im Beispiel wird ein Leerzeichen *vor* jedem Satzzeichen eingefügt und ein Leerzeichen *nach* dem öffnenden Anführungszeichen:

```
cat kafka.txt | sed -e 's/\([\.,;:!\?«]\)/ \1/g' -e 's/\([»]\)/\1 /g'
```