8. Editieren von Files

vi und andere Scheußlichkeiten, Reguläre Ausdrücke

Klassische Editoren − eine Übersicht:

ed − Sehr einfacher zeilenorientierter Editor. Funktioniert eigentlich immer. Ist nicht sehr benutzerfreundlich.

sed − Steam−Editor. Nimmt einen Eingabedatenstrom und bearbeitet diesen. Wird üblicherweise nur von der Kommandozeile aus aufgerufen.

vi − Sehr alter bildschirmorientierter Editor. Nahezu überall verfügbar. Die Bedienung ist gewöhnungsbedürftig, aber trotzdem alltagstauglich.

emacs − Ursprüglich tastaturgesteuerter Editor. Unterstützt zahlreiche Funktionen außerhalb des Editierens von Dateien.

nano − Leicht zu benutzender Editor für die Kommandozeile. Unter Linux fast immer verfügbar. Die Bedienung erklärt sich von selbst.

Zahlreichre GUI−Editoren − xedit, kwrite, gedit, kate, ...

Es gibt noch viele weitere Editoren für verschiedene Zwecke und mit verschiedenen Fähigkeiten.

Zwischen vi (bzw. vim)−Nutzern und emacs−Nutzern gibt es regelrechte Religionskriege.

Hier betrachtet:

- **nano**
- **vi**
- **emacs**

Der Editor nano

Motivation:

Sehr einfach zu bedienen. Die gebräuchlichsten Tastenkombinationen werden ständig am unteren Bildschirmrand angezeigt. Nano ist auf vielen Systemen vorinstalliert.

Aufruf:

```
nano <Dateiname>
```

Steuerkommandos:

Cursorpositionierung mit den Pfeiltasten.

Tastenkombinationen:

Ctrl−O	Speichern
Ctrl−X	Schließen

Der Editor vi

Motivation:
Auf den meisten Systemen verfügbar. Sehr mächtig. Kann sehr stark an die eigenen Bedürfnisse angepasst werden.

Varianten von vi: vim, gvim

Aufruf:
vi [−r] [−R] <datei>
−r − recover − Editieren der Datei <datei>, deren Änderung vorher nicht beendet wurde (Systemabsturz, vi−Abbruch)
−R − read only − Datei nur zum Lesen öffnen (view)
<datei> − Datei, die bearbeitet werden soll

z.B.
vi b1
view b1
vi −r b2

vi ist vom Termnial abhängig und wird nicht überall gleich dargestellt!

Prüfen ob das Terminal vom System unterstützt wird:
tput −T $TERM longname

Umgebungsvariable TERM setzen (sh, bash):
export TERM=<Terminal>

```
Arbeitszustände des vi
----------------------
vi ist ein Modusbasierter Editor.

    Befehlsmodus         - Navigation im File, Ausführen verschiedener
     (Normal mode)         Operationen auf Text, Standardmodus (wird nach
                           dem Start angezeigt)
    Einfügemodus         - Schreiben von Text
     (Insert mode)
    Kommandozeilenmodus - Ausführen von ex-Kommandos (zeilenorientiert,
     (Command mode)       ähnlich ed)
Moduswechsel:
    Normal mode -> Insert mode
        i, a, o, R, ...
    Normal mode -> Command mode
        :
    Insert Mode -> Normal mode
        <ESC>, <CTRL>-C
    Command Mode -> Normal mode
        <ESC>, [Kommando]<ENTER>, <CTRL>-C
```

Wichtige vi−Kommandos

Cursor positionieren: (Kommando wird nicht angezeigt):

h	− Cursor ein Zeichen nach links (Cursor−Taste erlaubt)
j	− Cursor eine Zeile runter (Cursor−Taste erlaubt)
k	− Cursor eine Zeile hoch (Cursor−Taste erlaubt)
l	− Cursor ein Zeichen nach rechts (Cursor−Taste erlaubt)
w	− Cursor ein Wort nach rechts (über Zeilenende hinaus)
G	− Cursor auf letzte Zeile positonieren
<nn>G	− Cursor auf <nn>−te Zeile positionieren.
0	− Cursor am Zeilenanfang positionieren
$	− Cursor am Zeilenende positionieren

Cursor positionieren durch Suchen:

/<string>	− suche String vorwärts
?<string>	− suche String rückwärts
n	− wiederholen des letzten Suchkommandos
N	− wiederholen des letzten Suchkommandos rückwärts

File verändern: (Kommando wird nicht angezeigt):

```
x             − Zeichen unter Cursor löschen
dw            − Wort ab Cursor bis Wortende löschen
dd            − Zeile unter Cursor löschen
r<Zeichen>    − Zeichen unter Cursor ersetzen
J             − nachfolgende Zeile wird an die aktuelle Zeile angehangen
u             − undo − letzte Änderung rückgängig machen
                       eventuell mehrfach anwendbar
a<Text><ESC> − einfügen des Textes <Text> nach dem Cursor
                (auch mehrere Zeilen)
A<Text><ESC> − einfügen des Textes <Text> am Ende der Zeile
                (auch mehrere Zeilen)
i<Text><ESC> − einfügen des Textes <Text> vor dem Cursor
                (auch mehrere Zeilen)
I<Text><ESC> − einfügen des Textes <Text> am Anfang der Zeile
                (auch mehrere Zeilen)
o<Text><ESC> − einfügen des Textes <Text> vor der aktuellen Zeile
                (auch mehrere Zeilen)
O<Text><ESC> − einfügen des Textes <Text> nach der aktullen Zeile
                (auch mehrere Zeilen)
R<Text><ESC> − Wort ab Cursor durch Text <Text> ersetzen
```

Änderung speichern und vi verlassen:

ZZ − File schreiben und vi verlassen

:wq − File schreiben und vi verlassen

:w − File schreiben

:q − vi verlassen (wenn nichts geändert wurde)

:q! − vi immer verlassen ohne Speicherung der Änderungen

Weitere sinnvolle vi−Kommandos

s<Text><ESC> − Zeichen an der Cursorposition duch Text <Text> ersetzen
(auch mehrere Zeilen)

<nn>s<Text><ESC> −
<nn> Zeichen an der Cursorposition duch Text <Text> ersetzen
(auch mehrere Zeilen)

S<Text><ESC> − aktuelle Zeile duch Text <Text> ersetzen
(auch mehrere Zeilen)

:s/<alt>/<neu> −
in der aktuellen Zeile Zeichenkette <alt> durch Zeichenkette
<neu> ersetzen (einmalig)

:s/<alt>/<neu>/g −
in der aktuellen Zeile Zeichenkette <alt> durch Zeichenkette
<neu> ersetzen (mehrmalig)

:1,$s/<alt>/<neu> −
im gesammten File Zeichenkette <alt> durch Zeichenkette <neu>
maximal einmal pro Zeile ersetzen.

:1,$s/<alt>/<neu>/g −
im gesammten File Zeichenkette <alt> durch Zeichenkette <neu>
ersetzen (mehrmalig)

:r <Dateiname> −
Einlese der Datei <Dateiname> hinter die aktuelle Zeile

:e <Dateiname> −
Editieren des Files <Dateiname>

. − Ausführen der letzten Änderung im File an der aktuellen
Position des Cursors

:<nn> − Gehe zur <nn>−ten Zeile

Kurzer Ausflug zu den regulären Ausdrücken

Reguläre Ausdrücke werden von vielen Programmen im Unix benutzt um Zeichenketten zu beschreiben. Dabei sollen nicht nur einzelne Zeichenketten beschrieben werden, sondern Mengen von Zeichenketten.

Reguläre Ausdrücke sind also Muster für Zeichenketten.
Sie bestehen aus:

normalen Zeichen des Alphabets, z.B.

0-9, a-z, A-Z, Sonderzeichen ungleich Metazeichen

und

Metazeichen:

. * ^ $ [ ] \

Ein einfacher regulärer Ausdruck besteht aus:

- einem normalen Zeichen des Alphabets

- dem Metazeichen "\" gefolgt von einem weiteren Metazeichen dadurch wird das Metazeichen selbst Zeichen des Ausdrucks (ein Zeichen)

- Metazeichen "^" - stellt den Zeilenanfang dar

- Metazeichen "$" - stellt das Zeilenende dar

- Metazeichen "." - stellt ein beliebiges Zeichen dar

- Metazeichen: [...] - stellt ein Zeichen aus der Menge, der durch "[" und "]" eingeschlossenen Zeichen dar
  Abkürzung ist zulässig, z.B.
  [a-z] - alle Kleinbuchstaben
  [a-zA-Z] - alle Buchstaben
  [0-9] - alle Ziffern
  [m-s] - die Buchstaben mnopqrs

- Metazeichen: [^...] - stellt ein Zeichen aus der Komplementmenge von [...] dar.
  [^a-z] - alle Zeichen außer Kleinbuchstaben

Operatoren über einfache reguläre Ausdrücke

- Konkatenation (Verkettung): <Z1><Z2>
  Z1 unmittelbar gefolgt von Z2.
  Z1, Z2 sind Zeichen oder Metazeichen
  "an" beschreibt also eine Zeichenkette, die die Buchstaben "a" und "n" unmittelbar hintereinander enthaelt, also:

  "an", "anders", "Kanne"

- Gruppierungen: \(<regulaerer-Ausdruck>\) bedeuten das Gleiche wie <regulaerer-Ausdruck>
  also:

  "\(abc\)" entspricht "abc"

− Häufigkeiten:

<Z1>* : kein, einmaliges oder mehrmaliges Auftreten des Zeichens <Z1>

"al*es entspricht "aes", "ales", "alles", "alles", "allllles", ...

<Z1>\{m\}: m−maliges Auftreten des Zeichens <Z1>

"al\{2\}es" entspricht nur "alles"

<Z1>\{m,\}: mindestens m−maliges Auftreten des Zeichens <Z1>

"al\{2,\}es" entspricht "alles", "allles", ...

<Z1>\{m,n\}: m− bis n−maliges Auftreten des Zeichens <Z1>

"al\{2,3\}es" entspricht genau "alles" und "allles"

Einige Beispiele:

```
   1,$p
   /^meins/p
   /^$/n
   /^[  ]*$/n
   /ales/,/al*/p
   /ales/,/all*/p
   /ale*/p
   /ale*/n
   /ale./
   /al.les/p
   /al.*les/p
   /a[lb]*es/p
   /.al/p
   /al\{3\}es/p
   /al\{2,3\}es/p
   /a[bl]\{2,3\}es/p
   /\(alles\)/p
   /\(alles \)*/p
   /\(alles \)\{2\}/p
   /\(alles \)\{1,\}/p
   /\(alles \)\{1,2\}/p
 noch eine Steigerung
   /\(alles\) \1/p
   /\([a-zA-Z]\{1,\}\) \1/p
```

Weitere Positionierkommandos zum Blättern

<cntrl>f – Eine Bildschirm vorwärts blättern

<cntrl>b – Eine Bildschirm rückwärts blättern

<cntrl>d – Eine halben Bildschirm vorwärts blättern

<cntrl>u – Eine halben Bildschirm rückwärts blättern

Pufferverwaltung

Der vi benutzt Puffer, um sich einzelne oder mehrere Zeilen zu merken.
Es gibt einen allgemeinen Puffer, der bei bestimmten Kommandos automatisch gefüllt wird und die nutzerspezifischen Puffer mit den Namen a,b,c,d,...,z.

Kommandos für die Pufferverwaltung des allgemeinen Puffers:

yy oder Y	− aktuelle Cursorzeile in den allgemeinen Puffer kopieren
<nn>yy	−
<nn>Y	− <nn> Zeilen einschließlich der aktuellen Cursozeile in den allgemeinen Puffer kopieren
dd	− aktuelle Cursorzeile löschen und in den allgemeinen Puffer übernehmen
<nn>dd	− <nn> Zeilen einschließlich der aktuellen Cursorzeile löschen und in den allgemeinen Puffer übernehmen
p	− (put) allgemeinen Puffer hinter die aktuelle Cursorzeile kopieren
P	− (Put) allgemeinen Puffer vor die aktuelle Cursorzeile kopieren

Kommandos für die Pufferverwaltung von nutzerspezifischen Puffern:
<puffer> (einzelner Buchstabe)

"<puffer>yy − aktuelle Cursorzeile in den Puffer <puffer> übernehmen
"<puffer><nn>yy −
<nn> Zeilen einschließlich der aktuelle Cursorzeile in den Puffer <puffer> übernehmen

"<puffer>dd − aktuelle Cursorzeile löschen und in den Puffer <puffer> kopieren
"<puffer><nn>dd −
<nn> Zeilen einschließlich der aktuelle Cursorzeile löschen und in den Puffer <puffer> kopieren

"<puffer>p − (put) Puffer <puffer> hinter die aktuelle Cursorzeile kopieren
"<puffer>P − (put) Puffer <puffer> vor die aktuelle Cursorzeile kopieren

Textobjekte im vi

Textobjekte müssen für folgende vi−Kommandos spezifziert werden:

- c − ändern
- d − löschen
- y − sichern

Folgende Textobjekte werden unterstützt:

Worte, Sätze, Absätze, Abschnitte

Worte − Folge von Buchstaben, Ziffern und Unterstreichungsstrichen

Wort−Textobjekte:

- w − Wort ab Cursorposition bis Interpunktionszeichen nach rechts (ausschließlich)
- W − Wort ab Cursorposition bis Interpunktionszeichen nach rechts (einschließlich)
- b − Wort ab Cursorposition bis Interpunktionszeichen nach links (ausschließlich)
- B − Wort ab Cursorposition bis Interpunktionszeichen nach links (einschließlich)
- e − Ende eines Worts ab Cursorposition nach rechts ohne Interpunktionszeichen
- E − Ende eines Worts ab Cursorposition nach rechts mit Interpunktionszeichen

```
Satz:− Folge von Wörtern ohne ".", "!", "?", Leerzeilen

  Satz−Textobjekte:

     (     − Anfang des momentanen Satzes
     )     − Ende des momentanen Satzes


Absatz − bis zur nächsten Leerzeile oder bis zur nächsten Zeichenfolge,
         die durch den "paragraph"−Wert definiert wird.
         Definition des "paragraph"−Wertes:
               :set paragraphs=

  Absatz−Textobjekte:

     {     − Anfang des momentanen Paragraphs
     }     − Ende des momentanen Paragraphs


Sektion − bis zur nächsten Zeichenfolge, die durch den "sections"−Wert
          definiert wird.
            Definition des "paragraph"−Wertes:
               :set sections=

     [[    − Anfang der momentanen Sektion
     ]]    − Ende der momentanen Sektion
```

Sonstige Positionierkommandos

```
    ^      − rückwärts bis Zeilenanfang
    $      − vorwäts bis Zeilenende
    G      − vorwärts bis Dateiende
  <nn>G    − vorwärts bzw. rückwärts bis zur <nn>−ten Zeile
```

Einige Hilfskommandos

:!<UNIX−Kommando>	− Ausführen des UNIX−Kommandos <UNIX−Kommando>
<cntl>l	− Bildschirm neu aufbauen
z <NL>	− Aktuelle Cursorzeile wird an den oberen Bildschirmrand verschoben (scrollen)
z.	− Aktuelle Cursorzeile wird in die Mitte des Bildschirms verschoben (scrollen)
z+,z−	− aktuelle Cursorzeile vorwärts und rückwärts scrollen

set−Optionen

Durch Optionen werden die Eigenschaften des vi beeinflußt. Die Optionen können im File ~/.exrc oder ~/.vimrc abgelegt werden, so daß sie beim Starten des vi sofort gesetzt sind. Optionen könenn durch ":set" manipuliert werden.

```
   :set <option>=<Wert>  − Wert eintragen
   :set <option>         − Option einschalten
   :set <no−option>      − Option ausschalten
   :set <option>?        − Option anzeigen
   :set all              − Alle Optionen anzeigen
   :set                  − geänderte Optionen anzeigen
   :syntax on            − Syntaxprüfung einschalten
   :gui                  − grafische Oberfläche einschalten
```

Abkürzungen für Optionsnamen sind zulässig.

<option>

```
    autoident  (ai,noai)     − Automatisches einrücken
    autowrite  (aw,noaw)     − Automatisches sichern (verstärkt)
    ignorecase (ic,noic)     − Groß− und Kleinbuchstaben beim Suchen ignorieren
    list       (list,nolist) − Anzeigen von Tabulatorenzeichen
    magic      (magic,nomagic) − Bedeutung von Metazeichen für reguläre
                                 Ausdrücke einschalten, sonst nur ^ und $
                                 unterstützt
    number     (nu,nonu)     − Zeilennummerierung einschalten
    shiftwidth  (sw=,sw=8)   − Länge der Softwaretabulatoren bestimmen
```

Zum Üben für jeden zu Hause:

```
vimtutor
```

Der Editor emacs

emacs ist einer der mächtigsten Editoren im UNIX. Er ist beliebig erweiterbar und programmierbar.
Erfinder: Richard Stallmann (Free Software Foundation)
Ähnlich vi mit gvim gibt es eine GUI−Variante von emacs − xemacs.

emacs befindet sich nach dem Start immer im Eingabemodus
Cursortasten, Scroll−Tasten und Maus können zur Positionierung benutztwerden. Steuerkommandos beginnen mit:

<CNTL>
<ALT>
<CNTL>+<SHIFT>
<ALT>+<SHIFT>

Für emacs−Kommandos gibt es zwei Eingabemöglichkeiten:

1. Steuerkommando−Anfang + Kommandobuchstabe
2. <ALT>+X <Kommandoname> <Enter>

z.B. streichen eines Zeichens:
<CNTL>D
<ALT>+X delete−char

Sollte <ALT>+X nicht unterstützt werden, so funktioniert <ESC>+X

Starten und Beenden des emacs

starten:
  emacs <dateiname>

beenden:
  <CNTL>X<CNTL>C
  emacs fragt nach, ob gespeichert werden soll

Laden und Speichern von Dateien

<CNTL>X<CNTL>F <dateiname>	- Laden einer Datei
<CNTL>X<CNTL>W <dateiname>	- Schreiben einer Datei
<CNTL>X<CNTL>S	- Schreiben aller Dateien
<CNTL>X S	- Schreiben aller Dateien
<CNTL>X I	- Datei einfügen
<CNTL>Z	- emacs in den Hintergrund

Cursor positionieren

Cursortasten, Seite vorwärts, Seite rückwärts

<ALT> <	- Dateianfang
<ALT> >	- Dateiende
<ALT>X gotoline <enter> nn	- gehe zu Zeile nn (<ALT>M?)
<ALT>X line-number-mode	- Zeilennummerierung ein-/ausschalten

Löschen

```
  <Backspace−Taste>   − Zeichen löschen
  <CNTL>D             − Zeichen löschen
  <ALT>D              − Wort löschen
  <CNTL>K             − bis Zeilenende löschen
  <ALT>0<CNTL>K       − bis Zeilenanfang löschen (<ALT> NULL <CNTL>K)
```

Suchen

```
  <CNTL>S <Suchtext>  − vorwärts suchen
  <CNTL>S <ALT>N      − weiter vorwärts suchen
  <CNTL>R <Suchtext>  − rückwärts suchen
  <CNTL>S <ALT>P      − weiter rückwärts suchen
                    Suchtext ist String
  <CNTL><ALT>S <Suchtext>  − vorwärts suchen
  <CNTL><ALT>R <Suchtext>  − rückwärts suchen
                    Suchtext ist regulärer Ausdruck
```

Editierkommandos

<ALT>X overwrite−mode − umschalten zwischen Overwrite− und Insert−Modus
<ALT>C − Buchstabe unter Kursor groß, den Rest des Wortes klein
<ALT>L − Buchstabe unter Kursor klein, den Rest des Wortes klein
<CNTL>T − Vertauschen von zwei Buchstaben
<ALT>T − Vertauschen von zwei Wörtern

Online−Hilfe

<F1> − F1−Taste
<F1><F1> − Übersicht über Hilfsmemü
<F1> a <Schlüsselwort> − Hilfstexte zum Thema des Schlüsselwortes
<F1> b − Übersicht Tastenkürzel
<F1> f <Kommando> − Übersicht zum Kommando <Kommando>

Aufruf von UNIX−Kommandos

<ESC>!<kommando> − Ausführen des Kommandos <Kommando>
Ergebnis wird in einem anderen Fenster angezeigt
Fenster kann mit <CNTL>X0 gelöscht werden.

<ALT>x shell − Starten einer Shell im Fenster

und vieles Andere mehr