```
                12. Editieren von Files
                =======================

          ed, vi und andere Schäußlichkeiten

Übersicht:

ed    - einfachster Editor im UNIX, zeilenorientiert, stellt
        keinerlei Anforderungen an das E/A-Gerät. Auch bei unbekanntem
        Terminal einsetzbar, wenn sonst nichts funktioniert.
        ed ist überall verfügbar.
        Man sollte wissen, daß es ihn gibt und sich im Notfall an die
        elementarsten Kommandos erinnern.
        Nicht geeignet für den täglichen Gebrauch.

sed   - Streameditor. Nicht für den Einsatz im Dialog geeignet.
        Nur für den Kommandozeileneinsatz geeignet. Die Editier-
        kommandos werden in der Kommandozeile angegeben.

vi    - Ältester bildschirmorientierter Editor im Unix.  Sehr verbreitet.
        Es gibt kein Unix-System ohne vi.  Das Unix muß den Bildschirmtyp
        kennen!!  Gewöhungsbedürftige Nutzerschnittstelle.  Es gibt vi-Fans
        und vi-Hasser.  Für den täglichen Gebrauch geeignet.  Für Leute die
        auf vielen verschiedenen Unix-Systemen zu Hause sind - ideal.
        Neuere Fassungen des vi
                       vim  -  in Farbe und mit Syntaxunterstützung
                       gvim -  wie vim aber mit graphischer Unterstützung
```



```
ex    - Erweiterung von ed um vi

emacs - Bildschirmorientierter Editor, der sowohl tastaturgesteuert
        (ursprüngliche Version ohne X-Unterstützung) als auch mausgesteuert
        (neue aktuelle Version mit X-Unterstütung - xemacs) arbeiten kann.
        Mächtigster Texteditor im Unix. Beliebig erweiterbar und
        programmierbar. Kann außer dem gleichzeitigem Editieren von
        mehreren Files noch Mail-Lesen, News-Lesen, Kompilieren u.s.w.
        Vielseitig einsetzbares Werkzeug.
        Leicht gewöhnungsbedürftig. Vorsicht!! Suchtgefahr, es soll
        Informatiker geben, die nicht mehr ohne emacs bzw. xemacs
        leben können.

kwrite - Einfacher graphischer Editor (von kde). Nur bei X-Oberflächen
         einsetzbar (kedit).

xedit - Einfacher graphischer Editor, oft verfügbar

Mit weiteren Editoren ist zu rechnen.
```

Der Editor xedit

Motivation:

Einfacher Editor für grafische Oberlächen. Wird von der Kommandozeile aus aufgerufen. Benötigt keinerlei Vorkenntnisse durch den Nutzer. nicht sehr leistungsstark.

Aufruf:

```
/usr/bin/xedit <Dateiname>           # Linux
/usr/openwin/bin/xedit <Dateiname>  # Solaris
```

Die Datei muß vorhanden sein!!!!

Steuerkommandos:

Kursorpositionierung über die Maus und Tastatur (Pfeiltastern, PgUp, PgDwn).

Tastenkombinationen:

```
Ctrl-a    Zeilenanfang
Ctrl-b    ein Zeichen zurück
Ctrl-d    löschen Zeichen unter Kursor
Ctrl-h    löschen Zeichen vor Kursor
Ctrl-k    löschen vom Kursor bis Zeilenende
Ctrl-m    einfügen neue Zeile
Ctrl-v    eine Seite vorwärts
```

Beispiel:

```
xedit xxxxx
```


Der Editor ed

Motivation:

Universelle Einsatzmöglichkeiten
Leistungsstark durch reguläre Ausdrücke
extrem einfach, aber umständlich.

Aufruf:

```
ed [-s] [-p<promptzeichen> | -p <promptzeichen>] [<Dateiname>]

   -s - silent         - Unterdrückung von Ausschriften einiger Editier-
                         Kommandos
   -p <promptzeichen>
   -p<promptzeichen>   - Definition eines eigenen ED-Prompts
                         Standard ist die leere Zeichenkette.
```

Beispiele:

```
ed
ed  -p "--->"
ed  "-p--->"
ed  xxx
ed  "-p--->" xxx
```

Nach dem Aufruf gibt ED eine Zahl aus - Anzahl der Zeichen im File -
oder
eine Fehlermitteilung: xxxx: No such file or directory
falls die Datei nicht existiert. In jedem Fall kann der im Editier-Puffer befindliche Text editiert werden.

Arbeitsmodi des ED

ED kennt zwei Arbeitszustände:

1. **Kommandomodus**
2. **Eingabemodus**

Nach dem Start befindet sich ED immer im Kommandomodus, was durch die Ausgabe des Promptzeichen (standardmäßig leer) angezeigt wird. Die Datei ist in einen Arbeitspuffer eingelesen. Alle Operationen finden über dem Puffer statt. Wird ein falsches Kommando eingegeben, so quittiert ED dies mit der Ausgabe von "?" ohne das irgendetwas verändert wird.

Eingabemodus

Der Wechsel von Kommandomodus zu Eingabemodus erfolgt durch die Kommandos:

```
    i      -  insert - einfügen vor der aktuellen Zeile
    a      -  append - einfügen nach der aktuellen Zeile
    c      -  change - ändern der aktuellen Zeile, mehrere
                       Zeilen können eingesetzt werden.
```

Im Eingabemodus kann man nur in der aktuellen Zeile Korrekturen vornehmen. Bereits abgeschickte Zeilen können nicht verändert werden.
In der aktuellen Zeile kann man durch

```
        ^U (kill)  - die aktuelle Zeile löschen
        ^? (erase) - das letzte Zeichen in der aktuellen Zeile löschen
```

Der Eingabemodus wird durch Eingabe eines einzelnen Punktes auf einer Zeile beendet.

Kommandomodus

Editierkommandos können nur im Kommandomodus ausgeführt werden!!!
Allgemeine Aufbau eines Edit-Kommandos:

```
   [adresse1[,adresse2]] [edit-kommando [parameter]]
```

adresse1 und adresse2 sind Adressen innerhalb der Datei
Folgende Angaben sind für Adressen zulässig:

```
    .              - Aktuelle Zeile

    $              - letzte Zeile der Datei

    <Zeilennummer> - Zeile mit der Zeilennummer <Zeilennummer>

    /regulärer Ausdruck/
                   - Adressiert die erste Zeile nach der aktuellen Zeile,
                     die den regulären Ausdruch enthält (round robin),
                     vorwärts suchen in der Datei

    ?regulärer Ausdruck?
                   - Adressiert die erste Zeile vor der aktuellen Zeile,
                     die den regulären Ausdruck enthält (round robin),
                     rückwärts suchen in der Datei
```

Schreibweisen für Adressen:

```
[A1[,A2]]  - zwei Adressen angebbar. Das Edit-Kommando wirkt für den
             Bereich von A1 bis A2. Es darf aber auch nur eine Adresse
             oder keine Adresse angegeben werden. Dann wirkt das
             Edit-Kommando nur auf die angegebene bzw. aktuelle Zeile.

[A1]       - eine Adresse angebbar. Das Edit-Kommando wirkt nur für
             die angegebene Zeile. Wenn keine Adresse angegeben ist,
             wirkt das Edit-Kommando auf die aktuelle Zeile.

 -         - Keine Adressangabe, nur für Edit-Kommandos zum verlassen
             des ED.
```

Editierkommandos zur Orientierung

```
<NL>         - die nächste Zeile wird angezeigt

[A1[,A2]]p - anzeigen der Zeilen im Adressbereich

[A1[,A2]]l - anzeigen der Zeilen im Adressbereich mit Zeilenendezeichen

[A1[,A2]]n - anzeigen der Zeilen im Adressbereich mit vorangestellter
              Zeilennummern
```


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 regulären Ausdruck besteht aus:

- ein normales 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

- Metazeiche: [^...] - stellt ein Zeichen aus der Komplementmenge
                       von [...] dar.
                            [^a-z]   - alle Zeichen außer Kleinbuch-
                                       staben
```

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", "anmachen", "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",
                          "allles", "alllllles", ...

            <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
```


Edit-Kommandos

[A1]a - append

Anfügen von Text nach der adressierten Zeile
0a ist erlaubt, bedeutet vor der ersten Zeile einfügen
Beenden des Eingabemodus mit ".".

[A1]c - change

Ändern der adressierten Zeile. Eingabe von mehreren Zeilen erlaubt
Beenden des Eingabemodus mit ".".

[A1[,A2]]d - delete

Streichen der adressierten Zeilen

e <dateiname> - edit

Editieren des angegebenen Files. Die Editierung des aktuellen Files wird beendet. Fall ungesicherte Änderungen vorhanden waren gibt es eine Fehlernachricht. Wird des E-Kommando sofort wiederholt, gehen die ungesicherten Änderungen verloren und das neue File wird editiert.

f <dateiname> – file

Neudefinition des Dateinamen für Schreiboperation.

h – help

Kommentierung des letzten Fehlers

H – help–Modus

Es werden immer Fehlertexte anstelle des "?" ausgegeben.

[A1]i – insert

Einfügen nach der adressierten Zeile. Es können beliebig viele Zeilen eingefügt werden.
Beenden des Eingabemodus mit ".".

[A1,A2]j – join

Zusammenfügen der adressierten Zeilen zu einer Zeile (NL wird entfernt)
Es muß mehr als eine Zeile adressiert sein.

[A1[,A2]]m<Adresse> – move

Transportieren der adressierten Zeilen hinter die Adresse <Adresse>. <Adresse> ist eine Zeilennummer. Die ursprünglichen Zeilen werden gelöscht.

P – promt

Standardmäßig arbeitet ED ohne Prompt. Das Edit–Kommando "P" schaltet den Prompt "*" ein. Erneute Eingabe von P schaltet den Prompt wieder aus.

q – quit

Beeden von ED. Erzeugt einen Fehler, wenn vorher die Änderungen nicht mit dem Edit–Kommando "w" geschrieben wurden. Zweimal "q" hintereinande ignoriert den Fehler.

Q – quit

Wie Edit–Kommando "q" nur ohne Fehlerausschrift.

[A1]r <dateiname> – read

Lesen der Datei <dateiname> hinter die adressierte Zeile.
den Prompt wieder aus.

```
[A1[,A2]]s/<regulärer ausdruck>/<string>/       - substitute
[A1[,A2]]s/<regulärer ausdruck>/<string>/g
[A1[,A2]]s/<regulärer ausdruck>/<string>/<nn>

      Ersetzen des regulären Ausdrucks xregulärer ausdruck> in den
      adressierten Zeilen durch den angegebenen String <string>.
      Standardmäßig erfolgt nur eine Ersetzung pro Zeile. Wird "g"
      spezifiziert werden alle möglichen Ersetzungen pro Zeile durchgeführt.
      Wird eine Nummer (1 <= <nn> <= 512) spezifiziert, so werden maximal
      <nn> Ersetzungen pro Zeile durchgeführt.
      Ersetzungen können sich über mehrere Zeilen erstrecken. Das
      abschließende <NL>-Zeichen sind durch "\" zu maskieren.


[A1[,A2]]t<Adresse>     - transport

      Kopieren der adressierten Zeilen hinter die Adresse <Adresse>.
      <Adresse> ist eine Zeilennummer. Die ursprünglichen Zeilen bleiben
      erhalten.


u                       - undo

      Die letzte Änderung wird rückgängig gemacht.
```



```
[A1[,A2]w [<dateiname>]       - write

      Schreiben der adressierten Zeilen in die Datei <dateiname>.
      Ist kein Dateiname angegeben, wird der Name der zuletzt
      gelesenen Datei benutzt.
      Ist keine Adresse angegeben, werden alle Zeilen geschrieben.

!<unix-kommando>              - exec Unix-Kommando

      Ausführen des Unix-Kommandos <unix-kommando>.


[[+]<nn>]                       - next

      Positionieren auf die neue Zeile.
      Keine Adressangabe     - nächste Zeile
      <nn> als Adressangabe  - Angegebene Zeilennummer
      +<nn> als Adressangabe - aktuelle Zeilennummer plus angegebene
                               Nummer ist neue aktuelle Zeilennummer
```

Kommandos, die das Leben leichter machen:

```
<nn1>,<nn2>m<A1>   - Verschieben der angegebenen Zeilen nn1 bis nn2 an die
                     Adresse A1

<nn1>,<nn2>t<A1>   - Kopieren der angegebenen Zeilen nn1 bis nn2 an die
                     Adresse A1

m+                 - aktuelle Zeile mit der nachfolgenden Zeile vertauschen

m--                - aktuelle Zeile mit der vorangehenden Zeile vertauschen

m$                 - aktuelle Zeile ans Dateiende verschieben

m0                 - aktuelle Zeile an den Dateianfang verschieben

t.                 - aktuelle Zeile doppeln


g/^/m0             - File umdrehen
```


Der Bildschirmeditor vi

Varianten des vi: vim, gvim, elvis, xvi

Ist ein Standardeditor des UNIX-Systems
Arbeitet wie der ED pufferorientiert, d.H. die Daten werden in einen Puffer gelesen und beim Beenden der Editierarbeiten in das File zurückgeschrieben. Während der Editierarbeiten wird eine Kopie angelegt, die einen fast aktuelle Stand der Editierarbeiten enthält.

Der VI ist vom Terminaltyp abhängig!!!!!!!
Umgebungsvariable TERM muß richtig gesetzt sein, d.h. muß dem Typ des benutzten Terminals entsprechen.
Anschauen der Umgebungsvariablen TERM:

```
echo $TERM
```

Prüfen ob das Terminal vom System unterstützt wird:

```
tput -T <Terminalname> longname
```

z.B.

```
tput -T xterm longname
    xterm terminal emulator (X Window System)
tput -T vt100 longname
    dec vt100 (w/advanced video)
```

Datenbasis für Terminalinformationen:

```
    /etc/termcap  ( /usr/share/misc/termcap )  - Alte Datenbasis
                                  Enthält alle Beschreibungsinformationen in
                                  einem File - lesbar

    /usr/share/terminfo                        - Neue Datenbasis
                Enthält pro Anfangsbuchstabe oder Anfangsziffer von Terminaln
                ein Directory. In diesen Directories ist dann für jedes Termi
                ein File mit Beschreibungsdaten enthalten (nicht lesbar).

    Anzeigen von terminfo-Dateien:

         infocmp      z.B.   infocmp  vt100
                             infocmp
                             infocmp  $TERM

    Erzeugen einer terminfo-Datei aus einer Quelle:

         tic

    Initialisieren, Prüfen eines Terminals

        tput

           tput -T <Terminalname> longname   - s.o.
           tput [-T <Terminalname>] init     - initialisieren
           tput [-T <Terminalname>] reset    - rücksetzen
```


Prüfen der Umgebungsvarialben TERM

```
   echo $TERM
   printenv TERM
   env | grep TERM
```

Setzen der Umgebungsvariablen TERM

```
   sh-Umgebung (bash-Umgebung)
       TERM=xterm
       export TERM
       tput init

   csh-Umgebung (tcsh-Umgebung)
       setenv TERM xterm
       tput init
```

```
VI besitzt zwei Darstellungsmodi:

     1. vi-Modus   -  bildschirmorientiert
     2. ex-Modus   -  zeilenorientiert
  zwischen den Modies kann gewechselt werden:
      vi-Modus --> ex-Modus     -   Q   für mehrere ex-Kommandos
                                    :   für ein ex-Kommando
      ex-Modus --> vi-Modus     -   vi

  Der Leistungsumfang und der Kommandoaufbau im ex-Modus entspricht im
  wesentlichen den Arbeitsmöglichkeiten des ED.

Aufruf des VI

  vi  [-r <datei>] [-w<nn>] [-R] [-L] [-c <kommando> [<datei1>] [<datei2>] ..

     -r <datei> - recover - Editieren der Datei <datei>, deren Änderung vorhe
                  nicht beendet wurde (Systemabsturz, vi-Abbruch)
     -w <nn>    - window size - <nn> Anzahl der Zeilen im Fenster
     -R         - read only - der VI eröffnet das File nur zum Lesen
                  (view)
     -c <kommando> - ausführen des EX-Kommandos <kommando> beim Start des VI
     -L         - list - auflisten aller "abgestürzten" VI-Files
     <datei1>   - zu editierendes File

  z.B.
     vi b1
     view b1
     vi -L
     vi -r cb3
```



```
Arbeitszustände des VI

   direkter Kommandomodus (Kommandomodus)
   Eingabemodus
   Zeilen-Kommandomodus

Nach dem Starten befindet sich der VI im Kommandomodus.
Der Wechsel in den Eingabemodus erfolgt durch eines der folgenden Kommandos:

   i -  einfügen vor dem Cursor  (insert)
   I -  einfügen am Zeilenanfang der aktuellen Zeile
   a -  einfügen nach dem Cursor (append)
   A -  einfügen am Zeilenende der aktuellen Zeile
   o -  einfügen vor der aktuellen Zeile (open)
   O -  einfügen nach der aktuellen Zeile
   c -  ersetzen eines Textobjektes
          z.B.
           cw   - eines Worte
   C -  ersetzen in der aktuellen Zeile bis Zeilenende
   s -  ersetzen des Zeichens unter dem Cursor
   S -  ersetzen der ganzen Zeile unter dem Cursor
   R -  Überschreiben des Textes unter dem Cursor
```

Im Eingabemodus können Korrekturen nur an der aktuell eingegeben Zeile vorgenommen werden:

```
erase      -  löschen des letzen Zeiches (DEL-Taste)
kill       -  löschen der zuletzt eingegebenen Zeile  (<cntl>+U)
<cntl>+W   -  löschen des zuletzt eingegeben Wortes (mehrfach)
```

Der Wechsel vom Eingabemodus in den Kommandomodus (beenden des Eingabemodus) erfolgt durch:

```
ESC-Taste
```

Folgende Tasten haben mehrfache Funktionen:

```
ESC-Taste:   Eingabemodus:  Eingabeende
             Zeilen-Kommandomodus: Abbruch des Kommandos
             sonst:  Fehlersignal

<cntl>+C     - beenden der momentan laufenden Aktion - immer

<NL>         - Zeilen-Kommandomodus: Abschluß einer Kommandoeingabe
                             und Wechsel in den Kommandomodus
               EX-Modus: Abschluß einer Kommandoeingabe
               Eingabemodus: neue Zeile erzeugen, alte Zeile in den Puff
               Kommandomodus: Positionieren des Cursors am nächsten Zeil
               anfang.
```


Wichtige VI-Kommandos

Cursor positionieren durch Suchen:

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

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
```

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 löschen
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)
c<Text><ESC> - Wort ab Cursor durch Text <Text> ersetzen
               (auch mehrere Zeilen)
```


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
```

Weiter 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
```


Weiter 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 oder
<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 (kleines p)  - (put) allgemeinen Puffer hinter die aktuelle Cursorzeile
                 kopieren

P (großes P)   - (Put) allgemeinen Puffer vor die aktuelle Cursorzeile
                 kopieren
```


Textobjekte im VI

Textobjekte müssen für folgende VI-Kommandos spezfziert 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
```

Sonstiges

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

Einiger 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 ignorier
    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 derSoftwaretabulatoren bestimmen
```

Zum Üben für jeden zu Hause:

vimtutor

Der Editor emacs / xemacs

emacs ist einer der mächtigsten Editoren im UNIX. Er ist beliebig erweiterbar und programmierbar.
Erfinder: R.Stallmann (Free Software Foundation)
xemacs Ableitung von emacs mit gleicher Funktionalität.

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

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

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