robotron

# Systemhandbuch

# MUTOS – 8000

## Teil II Abschnitt 2

## Text-Editor

Streichhahn

robotron

Programmtechnische Beschreibung
fuer Buerocomputer A5120.16

# SYSTEMHANDBUCH

# MUTOS 8000

## TEIL II

## Abschnitt 2 - Text-Editor

VEB Robotron-Buchungsmaschinenwerk
Karl-Marx-Stadt
Stand: 12/85

Die vorliegende Dokumentation entspricht dem Stand 12/85. Nachdruck, jegliche Vervielfaeltigung oder Auszuege daraus sind unzulaessig.

Im Interesse einer staendigen Weiterentwicklung werden alle Leser gebeten, ihre Vorschlaege bzw. Hinweise dem
VEB Robotron-Buchungsmaschinenwerk
9010 Karl-Marx-Stadt
Annabergerstrasse 93
mitzuteilen.

Fuer das Betriebssystem MUTOS 8000 wurden folgende Dokumentationen erarbeitet:

Anwendungsbeschreibung MUTOS 8000

Systemhandbuch MUTOS 8000 Teil I
Abschnitt 1 Kommandos
Abschnitt 2,3 Systemrufe, Bibliotheksfunktionen
Abschnitt 4, 5, 7, 8 Special Files, Fileformate, Makropakete, Systemunterstuetzung

Systemhandbuch MUTOS 8000 Teil II
Abschnitt 1 Kommandointerpreter Shell
Abschnitt 2 Editor
Abschnitt 3 Testhilfsprogramm adb

Systemhandbuch MUTOS 8000 Teil III
Abschnitt 1 Textverarbeitung nroff

Sprachbeschreibungen
Assemblersprachbeschreibung U 8000
Sprachbeschreibung C-Sprache

Das Sachwortverzeichnis ist als Anlage in der Anwenderbeschreibung enthalten.

Zur Programmentwicklung und Programmtestungen fuer den Prozessor U 8000 werden zusaetzlich folgende Dokumentationen angeboten:
Monitortestsystem MON8000

CROSS-Software U8000 zum Betriebssystem UDOS
Assembler U8000ASM
Binder ZLINK
Absolutbinder IMAGER
Preprozessor INCLUDE

Inhaltsverzeichnis

## 1. Einleitung

**ed** ist ein 'text-editor', d.h. ein interaktives Programm, das mit Hilfe von Anweisungen, die es von einem Nutzer an einem Terminal erhaelt, 'text' erzeugen und veraendern kann. Der zu bearbeitende Text kann dabei eine Beschreibung (wie diese hier) oder ein Programm sein. Er kann auch Daten fuer ein Programm enthalten.

Diese Einfuehrung soll es Ihnen erleichtern, den Umgang mit dem **ed** zu erlernen. Die guenstigste Moeglichkeit dafuer ist, diese Beschreibung zu lesen und gleichzeitig am Buerocomputer **ed** zu verwenden, um die angegebenen Beispiele nachzuvollziehen.

Loesen Sie die Uebungsaufgaben! Sie enthalten Material, das im Text nicht vollstaendig beschrieben wird.
In einem Anhang werden die Kommandos zusammengefasst.

Diese Beschreibung ist eine Einfuehrung mit Lehrbuchcharakter. Es wird also nur auf einen Teil der Moeglichkeiten des Text-Editors eingegangen (es werden aber die gebraeuchlichsten und am haeufigsten verwendeten Moeglichkeiten des **ed** erlaeutert).

Die Taste, mit der man die Eingabe am Buerocomputer beenden kann, ist die Taste 'ENTER'. Wir werden uns also im folgenden mit dem Begriff <ENTER> auf diese Taste ('Eingabeende') beziehen.

## 2. Aufruf des Text-Editors

Wir setzen voraus, dass Sie das **login**-Kommando fuer Ihr System bereits eingegeben haben und auf Ihrem Bildschirm das Prompt-Zeichen, i.a. ein $-Zeichen, erschienen ist.

Die Eingabe des Kommandos

**ed** (gefolgt von <ENTER>)

ist die einfachste Moeglichkeit, den Text-Editor zu aktivieren. Dann koennen Sie mit dem Editor arbeiten. Er erwartet von Ihnen, dass Sie ihm 'mitteilen', was zu tun ist.

## 3. Erzeugen von Text - das append-Kommando 'a'

Nehmen wir als erstes an, dass Sie einen voellig neuen Text eingeben wollen. Es koennte vielleicht der erste Entwurf eines Artikels sein, der spaeter modifiziert werden muss. In diesem Abschnitt wird beschrieben, wie beliebiger Text eingegeben werden kann. Aenderungsmoeglichkeiten sind spaeter erlaeutert.

Starten Sie **ed** das erste Mal, ist das genau das gleiche, als wuerden Sie mit einem leeren Blatt Papier arbeiten - es ist noch keinerlei Text vorhanden. Er muss vom Anwender des Editors erst geliefert werden. Dafuer wird der Text entweder eingegeben oder von einem File gelesen. Wir beginnen mit der Eingabe von beliebigem Text. Danach wird erlaeutert, wie man im Editor Files lesen kann.

Zunaechst sollen einige Begriffe erlaeutert werden. Im Sinne der **ed**-Terminologie wird der zu bearbeitende Te> "in einem Puffer zwischengespeichert". Diesen Puffer kann man sich z.B. als eine Art Arbeitsspeicher vorstellen. Im Umfang jene Informationsmenge, die gerade editiert wird. Im uebertragenen Sinne entspricht der Puffer einem Stueck Papier, auf das man seinen Text schreiben wuerde, um ihn dann korrigieren oder veraendern zu koennen und ihn dann fuer eine weitere spaetere Bearbeitung abzulegen.

Durch die Eingabe von Befehlen, 'Kommandos' genannt, teilt der Nutzer dem Editor mit, wie sein Text bearbeitet werden soll. Die Kommandos werden durch ein einzelnes Zeichen dargestellt, wobei Buchstaben als Kleinbuchstaben eingegeben werden muessen. Jedes Kommando muss auf einer neuen Zeile eingegeben werden. In manchen Faellen steht vor der Kommandobezeichnung noch eine Information darueber, fuer welche Zeile oder Zeilen des Textes das Kommando angewendet werden soll. In den meisten Faellen liefert **ed** dem Nutzer nach der Abarbeitung eines Kommandos keine 'Rueckmeldung' - d.h. es wird weder ein Prompt-Zeichen noch irgendeine Nachricht, wie z.B. 'ready', ausgegeben.

Als erstes wollen wir uns mit dem Kommando **append** vertraut machen, das nur durch den Buchstaben

```
a
```

aktiviert wird. Es bedeutet, dass die nachfolgend eingegebenen Textzeilen in den Puffer geschrieben ('an den Puffer angehaengt') werden. Das Hinzufuegen von Zeilen zum Pufferinhalt entspricht dem Aufschreiben von Text auf ein Stueck Papier.

Wollen Sie also Textzeilen in den Puffer schreibe , geben Sie zuerst ein **a** und <ENTER> ein. Anschliessend werden die gewuenschten Textzeilen eingegeben, z.B.

```
a <ENTER>
Da steh' ich nun, <ENTER>
ich armer Tor,  <ENTER>
und bin so klug <ENTER>
als wie zuvor.  <ENTER>
          (aus "Faust") <ENTER>
. <ENTER>
```

Die Eingabe eines einzelnen Punktes (auf einer neuen Zeile) ist die einzige Moeglichkeit, die Eingabe von Text zu beenden. Durch den . wird **ed** mitgeteilt, dass die Texteingabe beendet ist. (Sogar erfahrene Nutzer vergessen manchmal den abschliessenden Punkt. Scheint es bei der Eingabe weiterer Kommandos so, als wuerde der Editor diese ignorieren, dann geben Sie auf einer neuen Zeile eben diesen . ein. Sie koennen feststellen, dass Ihr Text einige ueberfluessige Zeilen enthaelt, die Sie spaeter wieder entfernen muessen).

Der Puffer enthaelt nach dem Abarbeiten des **append**-Kommandos folgende Zeilen:

```
Da steh' ich nun,
ich armer Tor,
und bin so klug
als wie zuvor.
          (aus "Faust")
```

Das Zeichen **a** und der . sind nicht im Puffer enthalten, denn sie gehoeren nicht zum eingegebenen Text.

Wollen Sie zu dem bereits vorhandenen Text neuen Text hinzufuegen, koennen Sie ein weiteres **a**-Kommando eingeben und dann mit der Texteingabe fortfahren.

## 4. Fehlermeldungen - "?"

Begehen Sie bei der Eingabe von **ed**-Kommandos einen Fehler, teilt Ihnen das der Editor durch Ausgabe eines

```
?
```

mit.

Diese verschluesselte Form der Fehlermeldung ist in der Praxis normalerweise ausreichend, um den entsprechenden Fehler herauszufinden.

## 5. Ausgabe von Text in ein File - das write - Kommando 'w'

Sicherlich wollen Sie sich den eingegebenen Text fuer eine spaetere Weiterverarbeitung aufbewahren. Zur Ausgabe des Pufferinhalts in ein File koennen Sie das **write**-Kommando

```
w
```

verwenden, wobei Sie den Namen des Files, in das Sie schreiben wollen, mit angeben muessen. Dadurch wird der Pufferinhalt in das angegebene File kopiert. Der bisherige Inhalt dieses Files wird zerstoert.

Z.B. wird durch Eingabe des Kommandos

```
w teil1
```

der Text vom Edit-Puffer in das File mit dem Namen **teil1** gespeichert.

Zwischen der Kommandobezeichnung **w** und dem Filenamen muss genau ein Leerzeichen stehen. **ed** gibt als Antwort auf dieses Kommando die Anzahl der geschriebenen Zeichen aus.

Im angegebenen Beispiel wuerde der Editor die Zahl

```
88
```

ausgeben. (Im Text enthaltene Leerzeichen und das <ENTER>-Zeichen am Ende jeder Zeile werden mitgezaehlt.) Da bei der Ausgabe des Textes in ein File nur eine Kopie erzeugt wird, wird der Inhalt des Puffers nicht zerstoert, sodass Sie mit der Eingabe von Text fortfahren koennen. Das ist ein ganz wichtiger Fakt. D.h., dass **ed** zu jedem Zeitpunkt nur die Kopie eines Files bearbeitet, nicht das File selbst. Solange Sie kein **w**-Kommando eingeben, veraendert sich der Inhalt des entsprechenden Files nicht. (Hinweis: Es ist sinnvoll, bei der Eingabe laengerer Textpassagen den Pufferinhalt von Zeit zu Zeit in ein File auszugeben. So geht Ihnen bei einem eventuellen "Systemabsturz" bzw. nach einem schweren Fehler nicht der gesamte Pufferinhalt verloren, denn bereits in ein File ausgegebener Text ist relativ sicher.)

## 6. Beenden der Editor-Arbeit - das quit-Kommando 'q'

Wollen Sie die Arbeit mit **ed** beenden, muessen Sie sich das Ergebnis Ihrer Arbeit sichern, indem Sie durch Eingabe eines **w**-Kommandos den Inhalt des Puffers in ein File schreiben.

Anschliessend geben Sie das **quit**-Kommando

```
q
```

ein. Das MUTOS-System reagiert darauf mit der Ausgabe des Prompt-Zeichens (*). Bei dieser Operation wird auch der Inhalt des Edit-Puffers geloescht. Das ist der Grund dafuer, dass Sie unbedingt vor einem **quit**-Kommando den Pufferinhalt in ein File schreiben sollten.

Geben sie ein **quit**-Kommando ein, ohne vorher mit **w** den Pufferinhalt abzuspeichern, gibt Ihne der Editor ein ? uf den Bildschirm aus. Zu diesem Zeitpunkt koennen Sie die **write**-Operation nachholen. Wird das nicht gewuenscht, kann die Editorarbeit durch ein weiteres **q**-Kommando, das dann auf jeden Fall ausgefuehrt wird, beendet werden.

**Aufgabe 1**

Rufen Sie den Editor auf (Eingabe von **ed**) und erzeugen Sie dann, entsprechend der Eingabefolge

```
a
... text ...
.
```

einen beliebigen Text. Schreiben Sie mit dem **w**-Kommando den Text in ein File. Anschliessend beenden Sie mit dem **q**-Kommando die Editorarbeit. Zur Kontrolle koennen Sie sich das File ausdrucken. (Dazu muessen Sie nach Erscheinen des Prompt-Zeichens auf dem Bildschirm entweder das Kommando

```
pr filename
```

oder

```
cat filename
```

eingeben. Probieren Sie beide Kommandos aus.)

## 7. Einlesen von Text aus einem File - das edit-Kommando 'e'

Wollen Sie Text, der in einer vorausgegangenen Sitzung mit dem **w**-Kommando in ein File gespeichert wurde, wieder editieren, muss dieser Text in den Edit-Puffer eingelesen werden. Dazu existiert das **edit**-Kommando **e**, das den gesamten Inhalt eines Files in den Puffer einliest. Haben sie z.B. als Uebungsaufgabe den Text "Da steh' ich nun,...usw." mit dem **w**-Kommando in das File **teil1** geschrieben , wuerde das **ed**-Kommando

```
e teil1
```

den gesamten Inhalt des Files **teil1** in den Puffer einlesen. Auf den Bildschirm wird noch die Zahl

```
88
```

ausgegeben, die Anzahl der vom File **teil1** eingelesenen Zeichen.

**Achtung:** Alles was sich vor dieser Operation im Puffer befand, wird vor dem Einlesen des neuen Textes geloescht.

Verwenden Sie am Anfang Ihrer Editorarbeit das **e**-Kommando zum Einlesen eines Files in den Puffer, kann bei einem nachfolgendem **w**-Kommando der Filename weggelassen werden. D.h., der Editor merkt sich den zuletzt in einem **e**-Kommando benutzten Filenamen. Vom **w**-Kommando wird dann in dieses File geschrieben.

Die Kommandofolge

```
ed
e file
...
[editorkommandos]
...
w
q
```

kann als Vorbild fuer Ihre eigene Arbeit mit dem Editor dienen.

Auf diese Weise koennen Sie sicher sein, dass bei einem von Zeit zu Zeit eingegebenen **w** genau in das richtige File geschrieben wird - vorausgesetzt, Sie haben am Anfang den richtigen Filenamen eingegeben.

Durch Eingabe des **file**-Kommandos **f** koennen Sie sich immer den Filenamen anzeigen lassen, den sich der Editor gerade merkt. In unserem Beispiel wuerden Sie auf die Eingabe von

```
f
```

die Antwort

```
teil1
```

erhalten.

## 8. Einlesen von Text aus einem File - das read-Kommando 'r'

Manchmal ist es notwendig, den Inhalt eines Files in den Puffer einzulesen, ohne dessen bisherigen Inhalt zu zerstoeren. Zu diesem Zweck gibt es das **read**-Kommando **r**. Durch das Kommando

```
r teil1
```

wird der Inhalt vom File **teil1** in den Puffer so eingelesen, dass der Text vom File **teil1** an den Text, der sich gerade im Puffer befindet, angehaengt wird. Geben Sie also nach einem **edit**-Kommando ein **read**-Kommando ein,

```
e teil1
r teil1
```

enthaelt der Puffer zwei Kopien des Textes von **teil1** (also 10 Zeilen):

```
Da steh' ich nun,
ich armer Tor,
und bin so klug
als wie zuvor.
          (aus "Faust")
```

```
Da steh' ich nun,
ich armer Tor,
und bin so klug
als wie zuvor.
          (aus "Faust")
```

Nach Beenden der **read**-Operation wird, analog zu den Kommandos **w** und **e**, dem Nutzer die Anzahl der gelesenen Zeichen angezeigt. Insgesamt wird **r** jedoch seltener verwendet als **e**.

**Aufgabe 2:**

Experimentieren Sie mit dem **e**-Kommando. Versuchen Sie, unterschiedliche Files zu lesen und auszugeben. Sollten Sie dabei die Fehlermeldung **?name** erhalten, wobei **name** der Name eines Files ist, bedeutet das, dass dieses File nicht existiert. Eine typische Fehlerursache dafuer ist, dass der Filename falsch geschrieben wurde. Es besteht aber auch die Moeglichkeit, dass der Zugriff auf das angegebene File nicht gestattet ist. Versuchen Sie abwechselnd, Text in ein File einzulesen bzw. an dieses anzufuegen, um die analoge Arbeitsweise der entsprechenden Kommandos zu verstehen. Ueberpruefen Sie die Aussage, dass

```
ed filename
```

aequivalent ist zu

```
ed
e filename.
```

Was bewirkt das Kommando

```
f filename      ?
```

## 9. Ausgabe des Pufferinhalts auf den Bildschirm - das Print-Kommando 'p'

Durch Eingabe des **print**-Kommandos

```
p
```

koennen Sie sich den Inhalt des Puffers (oder Teile davon) am Bildschirm anzeigen lassen. Wollen Sie sich nur bestimmte Teile des Pufferinhalts ansehen, koennen Sie vor der Kommandobezeichnung **p** durch Komma getrennt die Zeilen angeben, bei denen die Ausgabe beginnen bzw. enden soll. Folglich koennen Sie sich z.B. die ersten beiden Zeilen des Puffers (d.h. die Zeilen 1 bis 2) anzeigen lassen, indem Sie das Kommando

```
1,2p          (Beginn der Ausgabe bei Zeile  1,
                Ende  der Ausgabe bei Zeile 2)
```

eingeben. **Ed** wird Ihnen fuer unser Beispiel die Zeilen

```
Da steh' ich nun,
ich armer Tor,
```

auf den Bildschirm ausgeben.

Wollen Sie sich alle Zeilen des Puffers ausgeben lassen, koennten Sie analog zum obigen Beispiel, das Kommando **1,5 p** eingeben; aber nur, weil Sie wussten, dass der Puffer genau 5 Zeilen enthaelt. Im allgemeinen wissen Sie aber nicht genau, wieviele Zeilen der Puffer enthaelt, so dass Sie die Zeilennummer der letzten Zeile nicht exakt ngeben koennen. Deshalb kann im Editor als Symbol fuer die Zeilennummer der letzten Zeile des Puffers das Sonderzeichen * verwendet werden. Betrachten Sie dazu das Kommando

```
1,*p
```

Es gibt **alle** Zeilen des Puffers auf den Bildschirm aus (Zeile 1 bis zur letzten Zeile).

Wollen Sie sich die letzte Zeile des Puffers anzeigen lassen koennen Sie das Kommando

```
*,*p
```

eingeben. Dafuer laesst **ed** auch die abkuerzende Schreibweise

```
*p
```

zu. D.h., Sie koennen sich jede beliebige einzelne Zeile nur durch Angabe der Zeilennummer gefolgt von **p** auf den Bildschirm ausgeben lassen.
Folglich wird von

```
1p
```

die Ausgabe

```
Da steh' ich nun,
```

erzeugt, denn das ist die erste Zeile des Puffers.

Es sind sogar noch weiter abgekuerzte Kommandos erlaubt: so kann jede beliebige einzelne Zeile nur durc Eingabe ihrer Zeilennummer auf den Bildschirm ausgegeben werden. Bei Eingabe von

```
*
```

wird die letzte Zeile des Puffers angezeigt.

Das *-Zeichen kann aber auch in Kombinationen wie der folgenden verwendet werden:

```
*-1,*p
```

Damit werden die letzten beiden Zeilen des Puffers angezeigt. Das ist z.B. nuetzlich, um nachzusehen, wo her Text zuletzt eingegeben wurde.

**Aufgabe 3:**

Geben Sie, aehnlich wie in der vorigen Aufgabe, unter Verwendung des a-Kommandos einen beliebigen Text ein. Anschliessend probieren Sie bitte die verschiedenen Moeglichkeiten des p-Kommandos aus.

Dabei werden Sie z.B. feststellen, dass man die Zeile Ø oder eine Zeile nach dem Pufferende nicht ausgeben kann. Auch der Versuch, Zeilen in umgekehrter Reihenfolge, z.B.

```
3,1p
```

auszugeben, wird fehlschlagen.

## 10. Die aktuelle Zeile - DOT oder '.'

Nehmen wir an, dass Ihr Puffer noch die in einem der o.a. Beispiele erzeugten 1Ø Zeilen enthaelt und Sie gerade das Kommando

```
1,3p
```

eingegeben haben, worauf die entsprechenden drei Zeilen ausgegeben wurden. Geben Sie danach nur

```
p                    (ohne Zeilennummer)
```

ein, erscheint auf dem Bildschirm die Zeile

```
und bin so klug
```

Es ist die dritte Zeile des Puffers. Das ist genau die Zeile, mit der Sie zuletzt gearbeitet haben. (Sie haben sich diese ja gerade anzeigen lassen!) Geben Sie sofort anschliessend das p-Kommando ohne Zeilennummern noch ein- oder auch mehrmals ein, wird Ihnen immer die Zeile 3 ausgegeben. Die Ursache dafuer ist, dass sich **ed** genau die Zeile merkt, die Sie zuletzt bearbeitet haben (in unserem Beispiel die Zeile 3, die Sie als letzte ausgegeben haben), so dass man sich auf diese Zeile ohne Angabe einer expliziten Zeilennummer beziehen kann, und zwar durch Angabe des Sonderzeichens

```
('Punkt')
```

Genau wie das *-Zeichen kann man auch den Punkt als Zeilennummer angeben, das bedeutet, dass man sich auf die aktuelle Zeile beziehen moechte, (anders ausgedrueckt: auf die Zeile, mit der man zuletzt gearbeitet hat).

Eine der vielen Moeglichkeiten, dieses Symbol anzuwenden, ist z.B.

```
.,*p
```

Daraufhin werden die Zeilen von der aktuellen Zeile bis zur letzten Zeile des Puffers ausgegeben. In unserem Beispiel sind das die Zeilen 3 bis 10.

Es gibt Kommandos, bei deren Abarbeitung die aktuelle Zeile veraendert wird, waehrend das bei anderen nicht der Fall ist. Nach einem **p**-Kommando z.B. ist die aktuelle Zeile die Zeile, die zuletzt ausgegeben wurde. Nach dem letzten Beispielkommando beziehen sich sowohl . als auch * auf die Zeile 10.

Am haeufigsten wird der Punkt in seiner Sonderbedeutung in Kombinationen wie z.B. der folgenden:

```
.+1        (oder .+1p)
```

angewendet. Es bedeutet "Ausgabe der naechsten Zeile" und ist eine Moeglichkeit, sich den Inhalt des Puffers Zeile fuer Zeile anzeigen zu lassen. Sie koennen ebenso

```
.-1        (oder .-1p)
```

eingeben, was die "Ausgabe der vorhergehenden Zeile" bewirkt. So koennen Sie sich auch rueckwaerts durch den Puffer arbeiten. Ein weiterer haeufiger Anwendungsfall ist

```
.-3,.-1p
```

Dadurch werden die drei Zeilen vor der aktuellen Zeile ausgegeben.

Denken Sie daran, dass alle zuletzt genannten Kommandos den Wert der aktuellen Zeile (also den Wert von .) veraendern. Durch Eingabe des Kommandos

```
.=
```

koennen Sie sich jederzeit die Zeilennummer der Zeile ausgeben lassen, die gerade die aktuelle Zeile ist.

Wir wollen jetzt einiges von dem, was ueber das **p**-Kommando und den Punkt gesagt wurde, nochmals zusammenfassen. Vor der Kommandobezeichnung **p** koennen 0, 1 oder 2 Zeilennummern angegeben werden. Wurde keine Zeilennummer vor **p** angegeben,

wird die aktuelle Zeile angezeigt, auf die man sich ueber den . beziehen kann. Ist eine Zeilennummer angegeben (mit oder ohne den Buchstaben **p**), wird die gewuenschte Zeile (die damit auch zur aktuellen Zeile wird) ausgegeben. Bei der Angabe von zwei Zeilennummern werden alle Zeilen von der ersten bis zur zweiten Zeilennummer ausgegeben, wobei die zweite Zeilennummer immer groesser oder gleich der ersten Zeilennummer sein muss (siehe Aufgabe 2). In diesem Fall wird die zuletzt angezeigte Zeile zur aktuellen Zeile.

Geben Sie auf einer Zeile nur ein <ENTER> ein, entspricht das dem Kommando .+1p, und es wird die naechste Zeile ausgegeben. Geben Sie dagegen nur ein einzelnes - ein, werden Sie feststellen, dass diese Operation die gleiche Wirkung hat wie **.-1p**.

## 11. Loeschen von Zeilen - das Delete-Kommando 'd'

Nehmen wir an, dass Sie die fuenf zusaetzlichen Zeilen, die der Puffer noch enthaelt, wieder entfernen wollen. Dafuer koennen Sie das **delete**-Kommando

```
d
```

benutzen. Die Wirkungsweise dieses Kommandos ist mit der des **p**-Kommandos vergleichbar, mit dem Unterschied, dass man mit **d** die angegebenen Zeilen loescht, wohingegen sie mit **p** ausgegeben werden. Die zu loeschenden Zeilen werden auf die gleiche Art festgelegt, wie Sie das schon vom **p**-Kommando her kennen:

```
startzeile,endezeile d
```

Folglich loescht das Kommando

```
6,*d
```

die Zeilen ab Zeile 6 bis zum Ende des Pufferinhalts. Fuenf Zeilen sind noch im Puffer verblieben, was sich durch das Kommando

```
1,*p
```

leicht feststellen laesst.

Beachten Sie auch, dass * jetzt auf die Zeile 5 verweist. Nach einem **d**-Kommando wird die naechste Zeile nach der zuletzt geloeschten Zeile zur aktuellen (.); ausser, wenn die letzte geloeschte Zeile auch die letzte Zeile des Puffers war. In diesem Fall verweisen dann . und * auf die gleiche Zeile.

Aufgabe 4

Ueben Sie mit den Kommandos a, e ,r, w, p und d bis Sie sicher wissen, wie diese wirken und bis Sie ebenso sicher wissen, wie man mit Zeilennummern bzw. den Symbolen . und $ arbeitet.

Sind Sie wissbegierig, probieren Sie auch aus, welche Wirkung die Angabe von Zeilennummern bei einem a-, r- bzw. w-Kommando hat. Sie werden herausfinden, dass a den text **nach** der adressierten Zeile einfuegt (anstatt nach der aktuellen Zeile). r liest das angegebene File **nach** der adressierten Zeile ein (und nicht unbedingt immer an's Ende des Puffers) und w gibt genau die adressierten Zeilen aus und nicht immer den gesamten Pufferinhalt. Diese Moeglichkeiten koennen mitunter sehr nuetzlich sein; z.B. koennen Sie damit ein File an den Pufferanfang einlesen, indem Sie

```
Ør filename
```

eingeben. Ebenso koennen Sie mit

```
Øa
... text ...
.
```

Zeilen am Anfang des Puffers einfuegen.

Beachten Sie, dass sich .w **wesentlich** von

```
.
w
```

unterscheidet.

## 12. Veraendern von Text - das Substitute-Kommando 's'

An dieser Stelle nun das wichtigste aller Kommandos - das **substitute**-Kommando

```
s
```

Es ist das Kommando, mit dem man einzelne Woerter oder Buchstaben innerhalb einer Zeile oder in einer Gruppe von Zeilen veraendern kann. Sie muessen es z.B. dann benutzen, wenn Sie in Ihrem Text Schreib- oder Eingabefehler korrigieren wollen.

Nehmen wir an, dass durch einen Tippfehler in der Zeile 1

```
Da seh' ich nun,
```

steht - bei **seh** wurde das **t** vergessen. Sie koennen das mit dem **s**-Kommando in Ordnung bringen:

```
1s/seh/steh/
```

Das bedeutet: 'in Zeile 1 ersetze die Zeichenfolge **seh** durch **steh'**. Da **ed** das Ergebnis der Operation nicht automatisch anzeigt, sollten Sie es durch Eingabe von

```
p
```

kontrollieren. Es wird daraufhin die veraenderte Zeile

```
Da steh' ich nun,
```

ausgegeben. Haben Sie bemerkt, dass mit **p** (das ja in dieser Form eigentlich die aktuelle Zeile ausgibt) die veraenderte Zeile ausgegeben wurde? D.h., durch das **s**-Kommando wird die zu modifizierende Zeile zur aktuellen Zeile (nachdem die Substitution ausgefuehrt wurde).

Die allgemeingueltige Form des **substitute**-Kommandos sieht so aus:

**startzeile,endezeile** s **/zeichenkette/ersetzungs-zeichenkette/**

Die zwischen den ersten beiden Schraegstrichen (/) angegebene Zeichenkette wird in **jeder** der adressierten Zeilen (von der 'start-zeile' bis zur 'ende-zeile') durch die 'ersetzungs-zeichenkette' ersetzt. Sie wird jedoch in jeder Zeile nur einmal (beim ersten Auftreten) ersetzt. Soll die Zeichenkette in jedem Fall ersetzt werden (auch mehrmals in einer Zeile), schauen Sie sich dazu bitte Aufgabe 5 an. Die Zeilennummern koennen wie beim **p**-Kommando angegeben werden. Nach dem **s**-Kommando ist die letzte der veraenderten Zeilen die aktuelle Zeile. (Es gibt noch eine Falle fuer Unvorsichtige: wuerde keine Substitution ausgefuehrt, wird die aktuelle Zeile nicht veraendert. Als Warnung ein ? ausgegeben.)

Mit dem Kommando

```
1,$s/so das/so dass/
```

koennen Sie z.B. auf einmal in jeder Zeile des Puffers diesen Rechtschreibefehler korrigieren.

Ist das **s**-Kommando ohne Zeilennummern angegeben, wird die Substitution nur in der aktuellen Zeile ausgefuehrt. Daraus laesst sich eine verallgemeinerte Schreibweise ableiten,

```
s/etwas/etwas anderes/p
```

, wodurch die aktuelle Zeile korrigiert und zur Kontrolle auf den Bildschirm ausgegeben wird. (Beachten Sie bitte, dass auf der Zeile des **s**-Kommandos ein **p**-Kommando steht Bis auf wenige Ausnahmen kann das **p** nach jedem Kommando stehen; andere Kommandoverbindungen auf einer Zeile sind nicht gestattet.)

Auch die Eingabezeile

```
s/.../
```

ist sinnvoll. Sie bedeutet, dass die angegebene Zeichenkette in der Zeile geloescht, d.h. durch "nichts" ersetzt wird. So koennen ueberfluessige Worte in einer Zeile oder ueberfluessige Buchstaben in einem Wort geloescht werden. Haben Sie z.B. irrtuemlicherweise

```
Daaa steh' ich nun,
```

eingegeben, koennen Sie mit dem Kommando

```
s/aa//p
```

diesen Fehler berichtigen (vorausgesetzt, diese Zeile war die aktuelle !). Die korrigierte Zeile wird angezeigt:

```
Da steh' ich nun,
```

Beachten Sie, dass // (zwei aufeinanderfolgende Schraegstriche) fuer 'kein Zeichen' steht, nicht fuer 'ein Leerzeichen' ! Das ist ein wesentlicher Unterschied. (Spaeter werden wir noch eine weitere Bedeutung von // kennenlernen.)

**Aufgabe 5:**

Probieren Sie verschiedene Moeglichkeiten des **substitute**-Kommandos aus. Was passiert, wenn Sie ein Wort ersetzen wollen, das mehrmals auf einer Zeile vorkommt?
Geben Sie z.B.

```
a
er rennt, er springt, er laeuft
.
s/er/Max/p
```

ein, erscheint daraufhin auf dem Bildschirm die Zeile

```
Max rennt, er springt, er laeuft
```

Durch das **substitute**-Kommando wird die erste Zeichenkette nur an der Stelle ersetzt, wo sie das erste Mal gefunden wird. Sie koennen die Zeichenkette auch in jedem Fall ersetzen lassen, wenn Sie im **s**-Kommando noch ein **g** (fuer 'global') mit angeben:

```
s/.../.../g
```

Zur Begrenzung der beiden Zeichenketten koennen ausser dem Schraegstrich auch andere Zeichen verwendet werden, ausser Leer- und Tabulatorzeichen.

(Sollten Sie das mit einem der Zeichen

^ . * [ * \ &

probieren und sich ueber das Ergebnis wundern, lesen Sie bitte den Abschnitt ueber: "Zeichen mit Sonderbedeutung".)

## 13. Kontextabhaengige Mustersuche -- /.../

Nachdem Sie das **substitute**-Kommando kennengelernt haben, koennen wir uns nun mit einer weiteren, sehr wichtigen Moeglichkeit, die **ed** bietet, beschaeftigen - mit der Mustersuche.

Nehmen wir an, dass der Puffer zur Zeit unser Anfangsbeispiel enthaelt:

```
Da steh' ich nun,
ich armer Tor,
und bin so klug
als wie zuvor.
        (aus "Faust")
```

Jetzt wird die Zeile gesucht, die das Wort **klug** enthaelt, um dieses in **schlau** zu aendern. Bei nur fuenf Zeilen Pufferinhalt kann man sich noch leicht merken, welche der Zeilen das gesuchte Wort enthaelt. Enthaelt aber der Puffer mehrere hundert Textzeilen, die Sie bereits veraendert, zum Teil geloescht bzw. umgestellt haben, koennen Sie sich an die Nummer dieser Zeile nicht mehr erinnern. Durch die Mustersuche koennen Sie diese Zeile jedoch genau bestimmen, ohne deren Zeilennummer zu kennen, indem Sie die Zeile suchen, die das entsprechende Muster enthaelt. Geben Sie also

```
/gesuchte zeichenkette/
```

ein, 'suchen Sie nach der Zeile, die das angegebene Muster enthaelt'. Das **ed**-Kommando

```
/klug/
```

ist z.B. ausreichend, um die gewuenschte Zeile zu finden: es wird die Stelle im Puffer gesucht, wo die zwischen den Schraegstrichen angegebene Zeichenfolge (**klug**) erstmals vorkommt. Gleichzeitig wird die gefundene zur aktuellen Zeile und zur Kontrolle auf den Bildschirm ausgegeben:

```
und bin so klug
```

'Das naechste mal vorkommen' bedeutet dabei, dass **ed** bei der Zeile **.+1** zu suchen beginnt und bis zum Ende des Puffers weitersucht dann wird von der Zeile 1 bis zur aktuellen Zeile gesucht. Alle Zeilen des Puffers werden einmal durchsucht, bis entweder die gesuchte Zeile gefunden oder die aktuelle Zeile wieder erreicht wird. Kann das angegebene Muster in keiner

Zeile gefunden werden, gibt **ed** die Fehlermeldung

```
?
```

aus. Andernfalls wird die gesuchte Zeile ausgegeben.

Sie koennen auch die Suche nach einer bestimmten Zeile **und** eine Substitution gleichzeitig ausfuehren, z.B.

```
/klug/s/klug/schlau/p
```

Das Ergebnis lautet:

```
und bin so schlau
```

Dieses eine Kommando enthaelt eigentlich drei Kommandos: die Mustersuche nach der gewuenschten Zeile, das Ausfuehren der Substitution und die Ausgabe der Zeile auf den Bildschirm.

Der Ausdruck **/klug/** ist eine Anweisung zur Mustersuche. In ihrer einfachsten Form sehen alle Anweisungen zur Mustersuche wie diese aus - eine in Schraegstriche eingeschlossene Zeichenkette. Die Mustersuche kann auch anstelle einer Zeilennummer verwendet werden, sodass man sie fuer sich allein zum Suchen und Ausgeben einer gewuenschten Zeile verwenden kann, aber auch direkt als Zeilennummer fuer irgendein anderes Kommando, wie z.B. s. Fuer beide Faelle wurde bereits ein Beispiel angegeben.

Nehmen wir an, der Puffer enthaelt die fuenf bekannten Zeilen

```
Da steh' ich nun,
ich armer Tor,
und bin so klug
als wie zuvor.
          (aus "Faust")
```

Dann sind die **ed**-Zeilennummern

```
/klug/+1
/als/
/Faust/-1
```

Anweisungen zur Mustersuche, die sich auf die gleiche Zeile (4) beziehen. Um in Zeile 4 etwas zu veraendern, koennten Sie folgendes Kommando eingeben:

```
/klug/+1s/als/noch/
```

oder auch

```
/als/s/als/noch/
```

oder auch

```
/Faust/-1s/als/noch/
```

Es ist Ihnen ueberlassen, welches der Kommandos Sie verwenden. Alle fuenf Zeilen koennten Sie sich mit dem Kommando

```
/Da/,/Faust/p
```

ausgeben lassen; oder auch durch

```
/Da/,/Da/+4p
```

oder durch eine geeignete andere Kombination von Anweisungen zur Mustersuche. Das erste der beiden Kommandos ist dann besser geeignet, wenn Sie nicht genau wissen, wieviele Zeilen der Puffer enthaelt. (Enthaelt der Puffer nur fuenf Zeilen, wuerde man das Kommando

```
1,$p
```

benutzen; aber nicht, wenn sich Hunderte Zeilen im Puffer befinden.)

Generell gilt: eine Anweisung zur Mustersuche ist **das gleiche wie** eine Zeilennummer und kann ueberall dort verwendet werden, wo man Zeilennummern angeben kann.

**Aufgabe 6:**

Ueben Sie das Suchen von Mustern! Verwenden Sie einen Text, der die gleiche Zeichenkette mehrmals enthaelt und durchsuchen Sie den Text immer mit dem gleichen Muster.

Versuchen Sie, die Mustersuche in **substitute**-, **print**- oder **delete**-Kommandos als Zeilennummer zu verwenden. (Auch fuer **r**-, **w**- und **a**-Kommandos ist das moeglich.)

Suchen Sie ein Muster, wobei Sie anstelle von **/text/** die Form **?text?** verwenden. Bei dieser Form der Angabe werden die Zeilen des Puffers in umgekehrter Reihenfolge durchsucht. Das ist dann sinnvoll, wenn man weiss, dass die gesuchte Zeichenkette weit von der aktuellen Zeile entfernt ist, so dass es effektiver ist, rueckwaerts zu suchen.

(Sollten Sie eines der Zeichen

```
^  .  *  [  *  \  &
```

verwenden und sich ueber das Ergebnis wundern, lesen Sie bitte den Abschnitt ueber 'Zeichen mit Sonderbedeutung'.)

**Ed** bietet Ihnen auch eine abkuerzende Schreibweise fuer die wiederholte Suche nach gleichen Zeichenketten an. Durch Eingabe von

```
/string/
```

findet man z.B. die **naechste** Zeile, in der **string** vorkommt.

Oftmals ist das aber noch nicht die gesuchte Zeile, so dass man weitersuchen muss. Dies fuehrt der Editor aus, indem Sie nur

```
//
```

eingeben. Diese Abkuerzung steht fuer "suche die zuletzt gesuchte Zeichenkette". Die gleiche Abkuerzung kann fuer die erste Zeichenkette im **substitute**-Kommando benutzt werden, wie z.B.

```
/string1/s//string2/
```

, wodurch die naechste Zeile, die **string1** enthaelt, gesucht wird und in dieser Zeile **string1** durch **string2** ersetzt wird. Dadurch muss man viel weniger eingeben. Analog bedeutet

```
??
```

'suche rueckwaerts weiter nach der zuletzt gesuchten Zeichenkette'.

## 14. Ersetzen und Einfuegen - 'c' und 'i'

In diesem Abschnitt wird das **change**-Kommando

```
c
```

behandelt. Mit diesem Kommando koennen eine oder mehrere Zeilen veraendert oder ersetzt werden. Ausserdem wird noch das **insert**-Kommando

```
i
```

zum Einfuegen einer oder mehrerer Zeilen erlaeutert.
Mit der **change** - Kommandobezeichnung

```
c
```

kann eine Anzahl Zeilen durch andere Zeilen, die ueber Bildschirm einzugeben sind, ersetzt werden. Um z.B. die Zeilen **.+1** bis **$** durch andere Zeilen zu ersetzen, muessen Sie

```
.+1,$c
...       hier geben Sie die gewuenschten Zeilen ein ...
.
```

eingeben. Die Zeilen zwischen der Startzeile und der Endezeile (**.+1** bis **$**) werden durch jene Zeilen ersetzt, die Sie zwischen dem c-Kommando und dem Punkt eingeben. Dadurch koennen z.B. Zeilen mit falschen Informationen ersetzt werden.

Wird durch das c-Kommando nur eine Zeile adressiert, so wird genau diese Zeile ersetzt (wobei Sie auch hier beliebig viele Ersetzungszeilen angeben koennen). Denken Sie an den Punkt .

nach der letzten Eingabezeile - er entspricht dem . nach dem **append**-Kommando und muss allein auf einer neuen Zeile stehen. Ist keine Zeilennummer vor **c** angegeben, wird die aktuelle Zeile ersetzt. Nach Abarbeiten des **c**-Kommandos entspricht die aktuelle der zuletzt eingegebenen Zeile.

**insert** wirkt aehnlich wie **append**. Z.B. wird durch

```
/string/i
...   einzufuegende Zeilen
.
```

der eingegebene Text **vor** der naechsten Zeile eingefuegt, die **string** enthaelt. Der Text zwischen **i** und **.** wird **vor** der adressierten Zeile **eingefuegt.** Ist keine Zeilennummer angegeben, wird vor der aktuellen Zeile eingefuegt. Nach der Operation ist die zuletzt eingefuegte die aktuelle Zeile.

Aufgabe 7:

**change** entspricht im Prinzip der Kommandofolge **delete**, gefolgt von **insert**. Probieren Sie aus, ob

```
start,end d
i
... text ...
.
```

das gleiche Ergebnis liefert wie

```
start,end c
... text ...
.
```

(wobei fuer **start** und **end** fuer Ihr Beispiel gueltige Adressen einzusetzen sind.) Wird allerdings die Zeile **$** mit geloescht, liefern beide Varianten nicht das gleiche Ergebnis. Ueberpruefen Sie das! Welche Zeile wird zur aktuellen Zeile?

Experimentieren Sie mit **a** und **i**. Sie wirken aehnlich, aber nicht gleich. Sie werden feststellen, dass bei

```
line-number a
... text ...
.
```

**text nach** der adressierten Zeile, durch

```
line-number i
... text ...
.
```

aber **vor** der adressierten Zeile eingefuegt wird. Ueberpruefen Sie, ob, wenn keine Zeilennnummer eingegeben wird, von **a** aus nach und von **i** aus vor der aktuellen Zeile eingefuegt wird.

## 15. Umstellen von Text - das Move-Kommando 'm'

Mit dem **move**-Kommando **m** kann Text umgestellt werden, d.h. eine Gruppe von Zeilen kann im Editpuffer von einer Stelle an eine andere geschrieben werden. Nehmen wir an, Sie wollen die ersten drei Zeilen des Puffers an dessen Ende schreiben. Dafuer koennten Sie die Kommandofolge

```
1,3w temp
$r temp
1,3d
```

benutzen. (Verstehen Sie, warum?) Wesentlich einfacher und effektiver ist aber, in diesem Fall das **m**-Kommando anzuwenden:

```
1,3m$
```

Allgemein dargestellt, heisst das

**start-zeile, ende-zeile m nach-dieser-zeile**

Beachten Sie, dass in diesem Kommando noch eine dritte Zeilennummer angegeben wird (nach der Kommandobezeichnung), und zwar die Nummer der Zeile, nach der die zu transportierenden Zeilen stehen sollen. Die entsprechenden Zeilen koennen natuerlich auch durch Mustersuche ermittelt werden. Angenommen im Puffer steht der Text:

```
Erstes Kapitel
...
Ende des ersten Kapitels
Zweites Kapitel
...
Ende des zweiten Kapitels
```

dann koennten Sie mit folgendem Kommando die beiden Kapitel vertauschen:

```
/Zweites/,/Ende des zweiten/m/Erstes/-1
```

Beachten Sie **-1**: die adressierten Zeilen werden nach der gewuenschten Zeile eingefuegt. Die letzte transportierte Zeile wird zur aktuellen Zeile.

## 16. Globale Operationen - die Kommandos 'g' und 'v'

Will man ein oder mehrere ed-Kommandos auf all jene Zeilen anwenden, die eine bestimmte (angegebene) Zeichenkette enthalten, kann man das **globale** Kommando **g** verwenden. Z.B. kann man sich mit dem Kommando

```
g/nummerieren/p
```

alle Zeilen anzeigen lassen, die 'nummerieren' enthalten. Haeufiger wird es sicher in Faellen angewendet, wie

```
g/nummerieren/s//numerieren/gp
```

, wobei in jeder Zeile, die 'nummerieren' enthaelt, diese Zeichenkette in jedem Fall durch die zweite (rechtschreiblich richtige) Zeichenkette ersetzt wird. **Alle** korrigierten Zeilen werden dabei auf Bildschirm ausgegeben. Vergleichen Sie das mit der Wirkung des Kommandos

```
1,$s/nummerieren/numerieren/gp
```

, wo nur die letzte veraenderte Zeile ausgegeben wird. Ein weiterer feiner Unterschied besteht darin, dass vom g-Kommando im Gegensatz zum **s**-Kommando, kein ? ausgegeben wird, wenn die Zeichenkette nicht gefunden wird.

Im g-Kommando koennen auch mehrere Kommandos (einschliesslich **a**, **c**, **i**, **r**, **w**, aber nicht **g**) angegeben werden, die fuer die Zeilen, die das angegebene Muster enthalten, auszufuehren sind. Jede Kommandozeile, ausser der letzten, muss mit einem Backslash \ enden:

```
g/xxx/.-1s/abc/def/\
.+2s/ghi/jkl/\
.-2,.p
```

Mit diesem Kommando werden die Zeilen herausgesucht, die **xxx** enthalten; jeweils die Zeile davor und danach wird veraendert. Anschliessend werden immer diese drei Zeilen ausgegeben.

Das **v**-Kommando wirkt analog zum g-Kommando, mit dem Unterschied, dass die angegebenen Kommandos nur fuer die Zeilen abgearbeitet werden, die das globale Muster **nicht** enthalten: z.B. loescht

```
v/ /d
```

alle Zeilen, die kein Leerzeichen enthalten.

## 17. Zeichen mit Sonderbedeutung

Sie werden beobachtet haben, dass bei der Mustersuche bzw. beim **substitute**-Kommando Fehler auftraten, wenn in den Zeichenketten die Zeichen ., *, ^ u.a. verwendet wurden. Der Grund dafuer ist, dass **ed** diese Zeichen als Zeichen mit Sonderbedeutung interpretiert. Z.B. bedeutet ein . **in einem Muster oder in der ersten Zeichenkette eines substitute-Kommandos**, dass an dieser Stelle ein 'beliebiges Zeichen' stehen kann. So wird z.B. durch

```
/x.y/
```

Jene Zeile zur aktuellen 'Zeile, die ein **x**, danach ein **beliebiges Zeichen** und danach ein **y** enthaelt', und **nicht** die Zeile, die nacheinander ein x, einen Punkt und ein y enthaelt. Alle folgenden Zeichen werden von **ed** als Zeichen mit

Sonderbedeutung interpretiert:

```
^   .   *   [   *   \   &
```

**Hinweis:**Backslash, \, hat in **ed** eine Sonderstellung. Aus Sicherheitsgruenden sollte seine Nutzung vermieden werden. Will man z.B. in einem **substitute**-Kommando eines der oben angegebenen Zeichen in seiner urspruenglichen Bedeutung verwenden, kann man dessen Sonderbedeutung zeitweise aufheben, indem man diesem Zeichen einen Backslash voranstellt. Folglich ersetzt das Kommando

```
s/\\\.\*/Backslash Punkt Stern/
```

die Zeichenfolge **\.*** durch **Backslash Punkt Stern.**

Es folgt ein kurzer Ueberblick ueber andere Zeichen mit Sonderbedeutung. Das erste ist der Zirkumflex **^**, der den Anfang einer Zeile kennzeichnet. Folglich wird durch

```
/^string/
```

nur die Zeile gesucht, wo **string** unmittelbar am Zeilenanfang steht. Die Zeile

```
string
```

wuerde gefunden, aber nicht die Zeile

```
ein string
```

Das Gegenstueck dazu ist das Dollar-Zeichen *, das fuer 'das Ende einer Zeile' steht. Mit

```
/string*/
```

wird nur eine Zeile ausgewaehlt, wo **string** am Ende der Zeile steht. Dann wird natuerlich mit

```
/^string*/
```

eine Zeile gesucht, die **nur** die Zeichenfolge **string** enthaelt, und

```
/^.*/
```

bestimmt eine Zeile, die nur ein einzelnes beliebiges Zeichen enthaelt.

Wie schon erwaehnt, steht der . fuer ein 'beliebiges Zeichen'. Das Muster

```
/x.y/
```

passt auf folgende Zeichenkombinationen:

```
x+y
x-y
x y
x.y
xzy    u.a.
```

Der Punkt wird besonders haeufig in Verbindung mit * angewendet, das als Wiederholungszeichen interpretiert wird. **a*** ist also eine Abkuerzung fuer "eine beliebige Anzahl von **a**'s". .* steht fuer "eine beliebige Anzahl beliebiger Zeichen". Letzteres kann wie folgt angewandt werden:

```
s/.*/unsinn/
```

ersetzt die gesamte aktuelle Zeile durch die zweite Zeichenkette, bzw. loescht mit

```
s/.*,//
```

alle Zeichen der Zeile bis zum und einschliesslich des letzten Kommas. (Da .* die laengste moegliche Zeichenkette bestimmt, geht diese bis zum letzten Komma der Zeile.)

[ wird in Verbindung mit ] zur Bildung von 'Zeichenklassen' verwendet. Z.B. wird durch

```
/[0123456789]/
```

eine beliebige **einzelne** Ziffer bestimmt - das Muster ist gefunden, wenn irgendeines der Zeichen in den eckigen Klammern erkannt wird. Abgekuerzt kann man dafuer [0-9] schreiben.

Schliesslich ist das Ampersand '&' ein weiteres abkuerzendes Sonderzeichen. Es kann nur in der 'Ersetzungs-zeichenkette' eines **substitute**-Kommandos verwendet werden und steht fuer "die Zeichenkette, die als letztes Muster gesucht wird". Dadurch kann die Eingabe verkuerzt werden. Angenommen

```
Da steh' ich nun,
```

ist die aktuelle Zeile, die Sie in runde Klammern einschliessen wollen. Sie koennten die ganze Zeile wieder in die 'Ersetzungszeichenkette' schreiben. Das ist aber ueberfluessig. Sie koennten auch

```
s/^/(/
s/$/)/
```

angeben, wobei Sie die Sonderbedeutung von ^ und $ ausnutzen. Am einfachsten ist es, & zu verwenden:

```
s/.*/(&)/
```

Das heisst, 'nimm die ganze Zeile (.*) und ersetze sie durch sich selbst, in runde KLammern eingeschlossen'. Das & kann auch mehrmals in einer Zeile verwendet werden, z.B. erzeugt

```
s/.*/&?  &!!/
```

die Zeile

```
Da steh' ich nun?  Da steh' ich nun!!
```

Es ist natuerlich nicht noetig, die gesamte Zeile als Muster anzugeben. Enthaelt der Puffer z.B. die Zeile

```
und bin so klug
```

koennten Sie das Kommando

```
/klug/s//& wie schon zuvor./
```

eingeben, um die Zeile

```
und bin so klug wie schon zuvor.
```

zu erzeugen.

Durch das letzte **s**-Kommando wird deutlich, wie man die Moeglichkeien des **ed** nutzen kann, um so wenig wie moeglich Text eingeben zu muessen. Mit der Zeichenkette **/klug/** wird die gewuenschte Zeile gesucht, die Abkuerzung '//' findet das gleiche Wort in dieser Zeile und durch die Verwendung von '&' brauchen Sie es auch in der Ersetzungszeichenkette nicht anzugeben.

'&' hat nur in der Ersetzungszeichenkette eines **substitute**-Kommandos eine Sonderbedeutung, nirgends sonst. Durch Voranstellen eines Backslash '\' kann die Sonderbedeutung von '&' aufgehoben werden:

```
s/ampersand/\&/
```

ersetzt in der aktuellen Zeile das Wort 'ampersand' durch das Sonderzeichen '&'.

## Anhang

### Kurzbeschreibung der Kommandos und Zeilennummern

Ein **ed**-Kommando besteht i.a. aus der Kommandobezeichnung, vor der eine oder zwei Zeilennummern angegeben werden koennen. Bei den Kommandos **e**, **r** und **w** kann noch ein Filename nach der Kommandobezeichnung folgen. Pro Zeile ist nur ein Kommando erlaubt. Nur das **p**-Kommando kann nach beliebigen Kommandos auf der gleichen Zeile mit angegeben werden (ausser bei **e**, **r**, **w** und **q**).

**a:** 'append', d.h. zum Pufferinhalt Zeilen hinzufuegen (nach der aktuellen Zeile, wenn nicht eine bestimmte Zeile adressiert wurde). Der nach dem Kommando eingebene Text wird solange in den Puffer geschrieben, bis auf einer neuen Zeile nur ein . eingegeben wird. Die letzte eingefuegte Zeile wird zur aktuellen Zeile.

**c:** 'change', die adressierten Zeilen werden geloescht. Dafuer wird der neue Text eingesetzt, der nach dem Kommando einzugeben ist. Die Eingabe wird wie bei **a** mit einem . abgeschlossen. Ist keine Zeilennummer angegeben, wird die aktuelle Zeile ersetzt. Die letzte veraenderte Zeile wird zur aktuellen Zeile.

**d:** 'delete', Loeschen der adressierten Zeilen. Ist keine Zeilennummer angegeben, wird die aktuelle Zeile geloescht. Die Zeile nach der letzten geloeschten Zeile wird zur aktuellen Zeile, ausser wenn die Zeile '$' mit geloescht wird. Dann ist '$' die aktuelle Zeile.

**e:** 'edit', Editieren eines neuen Files. Der vorherige Pufferinhalt wird geloescht, so dass man vor dem **e**- ein **w**-Kommando eingeben sollte.

**f:** 'file', Ausgabe des Filenamens, mit dem **ed** gerade arbeitet. Wird nach **f** ein Filename angegeben, wird dieser zum aktuellen Filenamen.

**g:** 'global', das Kommando

```
g/muster/kommandos
```

fuehrt die angegebenen Kommandos fuer alle die Zeilen aus, die 'muster' enthalten.

**i:** 'insert', Einfuegen der folgenden Zeilen vor der adressierten Zeile (bzw. vor der aktuellen Zeile), bis auf einer neuen Zeile ein . eingegeben wird. Die letzte eingefuegte Zeile wird zur aktuellen Zeile.

**m:** 'move', die adressierten Zeilen werden nach der Zeile eingefuegt, deren Adresse nach dem Kommandonamen steht. An ihrer alten Position werden diese Zeilen geloescht. Die letzte umgestellte Zeile wird zur aktuellen Zeile.

**p:** 'print', Ausgabe der adressierten Zeilen. Ist keine Adresse angegeben, wird die aktuelle Zeile angezeigt. Die Eingabe einer einzelnen Zeilennummer auf einer Zeile wirkt wie die Eingabe des Kommandos **zeilennummer p**. Eine leere Eingabe bewirkt die Ausgabe der naechsten Zeile (.+1).

**q:** 'quit', Beenden der **ed**-Arbeit. Loesch den gesamten Pufferinhalt, wenn es zweimal nacheinander ohne vorausgegangenes **w**-Kommando eingegeben wird.

**r:** 'read', Einlesen eines Files in den Puffer ( Ist keine Zeilennummer angegeben, wird an das Ende des Puffers eingeschrieben).

**s:** 'substitute', mit dem Kommando

```
s/zeichenkette1/zeichenkette2/
```

In der adressierten Zeile wird 'zeichenkette1' durch 'zeichenkette2' ersetzt. Ist keine Zeilennummer angegeben, wird die Substitution in der aktuellen Zeile ausgefuehrt. Die letzte veraenderte Zeile wird zur aktuellen Zeile, d.h. die aktuelle Zeile wird nicht veraendert, wenn keine Substitution ausgefuehrt wird. Mit **s** wird die 'zeichenkette1' in jeder adressierten Zeile nur einmal substituiert, und zwar dann, wenn sie das erste Mal erkannt wird. Soll 'zeichenkette1' in jedem Fall ersetzt werden, muss nach dem letzten Schraegstrich ein **g** angegeben werden.

**v:** Durch das Kommando

```
v/muster/kommandos
```

werden alle angegebenen Kommandos nur fuer die Zeilen abgearbeitet, die 'muster' **nicht** enthalten.

**w:** 'write', der Pufferinhalt wird in ein File geschrieben. Die aktuelle Zeile wird nicht veraendert.

**.=:** gibt die Zeilennummer der aktuellen Zeile us. ('=' allein gibt die Zeilennummer von $ aus).

**!:** Das Kommando

```
!kommandozeile
```

bewirkt, dass das in dieser Zeile angegebene Kommando als MUTOS-Kommando interpretiert und abgearbeitet wird.

**/muster/:** Kontextsuche. Die naechste Zeile die das 'muster' enthaelt wird gesucht und ausgegeben. Diese Zeile wird zur aktuellen Zeile. Gesucht wird ab Zeile '.+1' bis '$' und weiter von '1' bis '.', wenn das noetig ist.

**?muster?:** Kontextsuche in umgekehrter Richtung. Gesucht wird ab Zeile '.-1' rueckwaerts bis '1' und, wenn noetig, weiter von '*' bis '.'.

Kv 463/86 WV/6/1-10 3110a