EDV - ZENTRUM TECHNISCHE UNIVERSITÄT WIEN Hybridrechenanlage	TUNIX
	Operatoranleitung
	3. Auflage, Februar 1988 FB

TUNIX Operatoranleitung

1. TUNIX einschalten

- ON/RESET-Taste am TUNIX drücken (oberer oranger Knopf direkt unter dem Schlüsselschalter).
- Systemkonsole einschalten.
- Bei Eingabe von **<CR>** meldet sich der ICM Monitor mit % , sonst nochmals ON/RESET-Taste am TUNIX drücken und warten, bis sich nach etwa einer halben Minute der Monitor meldet.

2. Bootstrap (auf der Systemkonsole)

- Wenn sich der Monitor mit % gemeldet hat, **B <CR>** zum Starten der UNIX-Bootstrap-Prozedur eintippen.
- Die Frage nach dem Datum und der Uhrzeit wird normalerweise mit **y** zu beantworten sein. Stimmen Datum und/oder Uhrzeit nicht, so ist mit **n** zu antworten und es sind das richtige **Datum** und die richtige **Uhrzeit** in der geforderten Form einzutippen. Bei richtiger Kontrollausgabe schließlich mit **y** antworten. (Tippfehler können mit der BACKSPACE-Taste korrigiert werden, jede Eingabe ist mit einem **<CR>** abzuschließen. Groß- bzw. Kleinschreibung unbedingt beachten).
- Das File System sollte bei jedem Bootstrap überprüft werden, daher die Frage nach dem File System Check immer mit **y** beantworten.
- Die weiteren Schritte des Bootstraps laufen dann automatisch ab. Das System ist bereit, wenn auf der Konsole *Console Login:* erscheint.

3. TUNIX abschalten

- Login.
- Shutdown-Programm starten mit **SHUTDOWN** . Shutdown erfolgt nur dann, wenn sonst kein Benutzer mehr eingeloggt ist.
- Nach Aufforderung vom Rechner die ON/RESET-Taste am TUNIX drücken, dann etwa eine Minute warten und schließlich die OFF-Taste drücken.
- Systemkonsole abschalten.

4. System Backup

- Login.
- Streamer Tape für Backup von /u einlegen. Es sind abwechselnd die beiden Tagesbänder, alle 14 Tage eines der 2-Wochenbänder A,B oder C bzw. das entsprechende 2-Monatsband (Feb.,April,Juni,...) zu verwenden.
- Warten bis die Tapestation nicht mehr spult, dann den Befehl totape eintippen. Backup dauert ca. eine halbe Stunde. Nach erfolgreichem Backup erscheint Meldung, welcher File eine Liste der kopierten Files enthält.
- Logout auf der Konsole mit **<^d>** . Tape ausspannen und (außer bei den Tagesbändern) Backup Datum am Label eintragen. Backup im TUNIX Backup Protokoll vermerken.

5. System Operating

Informationen über die Benutzung des Systems erhält man nur, wenn man sich als eigener Benutzer einloggt und dann bestimmte Befehle eingibt.

5.1 Login

Falls auf der Konsole nicht die Aufforderung *Console Login:* bzw. *login:* aufscheint, **<CR>** oder **<^d>** eintippen. Sodann sind einzugeben:

ownername
password

Das System meldet sich nach der Ausgabe diverser Informationen mit *$* und ist bereit für die Eingabe von Befehlen.

<^d> statt eines Befehls bewirkt ein Logout.

5.2 Überblick über Benutzer, Prozesse, Spoolfiles

allusers liefert eine Liste aller möglichen Benutzer.

who liefert eine Liste aller eingeloggten Benutzer mit Terminaladresse, Ownernamen und Login-Zeit.

whodo listet alle eingeloggten Benutzer mit Terminaladresse, Ownernamen, Login-Zeit sowie die zugehörigen Prozesse mit ihrer Prozeßnummer, z.B.

```
tty1     fritz    15:38
    tty1     3841     0:04 sh
    tty1     3860     0:12 vi
```

lps listet den Printer-Status (Dataproducts-Printer ist *prl*) und alle Printerfiles in der Spool Queue mit Request-Id, Ownername, Filelänge und Zeitpunkt des Druckbefehls.

5.3 Abbrechen eines Tasks

Selbst gestartete Befehle oder Programme (z.B. Listen, Kompilierungen) können jederzeit durch Eingabe von **<DEL>** abgebrochen werden.

Fremde Befehle oder Programme können mit dem Befehl **KILL** abgebrochen werden. Dazu ist allerdings die Kenntnis der entsprechenden Prozeßnummer *pid* nötig, die man etwa durch Exekution des Befehls **whodo** erhält.

KILL pid(s) Der Befehl oder das Programm mit der Prozeßnummer *pid* wird abgebrochen. Es können mehrere *pids* (getrennt durch Leerzeichen) angegeben werden.

5.4 Reaktivieren des Printers

Der Printer ist üblicherweise immer bereit (*enabled*) zum Drucken. Es können jedoch Umstände eintreten (Umstecken, Stromausfall, etc.), die den Drucker desaktivieren. Der mit dem Befehl **lps** angezeigte Printer-Status ist dann *disabled* . Damit die Listen aus der Spool Queue gedruckt werden, muß der Printer neu aktiviert werden:

enable prl Der Printer *prl* wird aktiviert und kann Listen drucken.

Bei einem nachfolgenden **lps** muß der Printer-Status *enabled* sein. Ist dies nicht der Fall, liegt ein Hardware-Problem vor (Drucker nicht angeschlossen, nicht eingeschaltet u.ä.).

5.5 Errichten einer Benützungsberechtigung

adduser errichtet eine neue Benützungsberechtigung:

add user entry facility

enter user name: **user_name<CR>**
enter user group (RZ = 700, others >= 500): **500<CR>**
add user user_name group 500 ? (y/n): **y<CR>**

Nach der Eingabe des Namens und der Gruppennummer wird gefragt, ob die angegebene Benutzungsberechtigung errichtet werden soll. Bei Eingabe von **y** wird sie errichtet, bei **n** (z.B. bei Tippfehler bei der Eingabe) wird das Programm abgebrochen.

EDV - ZENTRUM TECHNISCHE UNIVERSITÄT WIEN Hybridrechenanlage	TUNIX
	Benützungsanleitung
	2. Auflage, Februar 1988 FB

TUNIX Benützungsanleitung

TUNIX Benützungsberechtigungen

Ansuchungsformulare für TUNIX Benützungsberechtigungen sind beim Operator erhältlich und auch dort abzugeben.

Der Benützer kann sich einen Benützernamen (maximal 8 Zeichen) aussuchen. Der Benützername ist nicht durch ein Password geschützt. Wird ein Password gewünscht, so muß nach dem ersten Login der Befehl **passwd** exekutiert werden. Das gewünschte Password muß mindestens 6 Zeichen haben (davon mind. 2 alphabetische Zeichen und mind. 1 numerisches Zeichen oder Sonderzeichen), und ist aus Sicherheitsgründen zweimal einzutippen. Durch weitere Aufrufe des Befehls **passwd** kann das Password laufend geändert werden.

Jeder Benützer erhält eine Directory, die unter dem Benützernamen ansprechbar ist. Auf dieser Directory kann er seine Files abspeichern und zusätzliche Subdirectories generieren. Es werden vom EDV-Zentrum tägliche Backups der Benützerdirectories gemacht.

Terminal-Session

Nach dem Einschalten des Terminals sind je nach Terminal-Typ eventuell off-line Initialisierungen durchzuführen. Das Terminal sollte auf Kleinbuchstaben-Ein/Ausgabe eingestellt sein. Der entsprechende TUNET-Server meldet sich nach Eingabe von **<CR>** (Carriage Return) z.B. mit

hybl>

Durch Eingabe von

do tunix<CR>

wird eine Verbindung zum TUNIX hergestellt. Es wird ausgeschrieben, welche Netzverbindung hergestellt wird, z.B.:

Querying Primary Name Server...
Connecting... session 1 -- connected to tunix
session 1 with tunix resumed

Falls keine Verbindung zum TUNIX hergestellt werden kann, wird

Connecting... Remote is busy

ausgeschrieben und man muß warten, bis ein Anschluß frei wird.

Das Betriebssystem UNIX des TUNIX-Rechners meldet sich nach der Eingabe von <CR> mit

login:

Jetzt muß der **Benützername** eingegeben werden, abgeschlossen durch <CR> .

Nach

Password:

muß das **Password** eingegeben werden, ebenfalls abgeschlossen durch <CR> . Die eingegebenen Zeichen sind dabei auf dem Bildschirm nicht sichtbar.

Ist der Benützername oder das Password nicht richtig oder hat man sich beim Eintippen geirrt (auch wenn man falsche Zeichen mit Backspace richtiggestellt hat), dann wird

Login incorrect
login:

ausgegeben und die Eingabe kann wiederholt werden.

Bei korrektem Benützernamen und korrektem Password liefert TUNIX Informationen über die UNIX Betriebssystem-Version und exekutiert den File *.profile* auf der Benützerdirectory. Dieser File enthält gewisse Initialisierungsbefehle (z.B. Festlegung des Terminaltyps für den Screen Editor, Ausgabe von Datum und Uhrzeit) und die Definition nützlicher Befehlsmakros. Der File *.profile* , der in einer Standardversion jedem Benützer auf seiner Directory zur Verfügung gestellt wird, kann vom Benützer jederzeit seinen Wünschen entsprechend modifiziert werden.

Sodann meldet sich TUNIX mit

$

als eingabebereit. Es können nun UNIX-Befehle eingegeben werden.

UNIX-Commands und Makros

Im folgenden sind die gebräuchlichsten UNIX-Commands und Command Makros angeführt. Einführend sind jedoch wichtige Sonderzeichen erklärt, die bei UNIX eine spezielle Bedeutung haben.

Spezielle Sonderzeichen

<BS> (Backspace) erlaubt die Korrektur von Tippfehlern bei der Eingabe von Befehlen.

<DEL> (Delete) wird von UNIX auch während der Exekution eines Befehls erkannt und bewirkt einen sofortigen Abbruch des Befehls.

<^s> (control s)
<^q> (control q) dienen zur Unterbrechung der Ausgabe des TUNIX-Rechners. Bei Eingabe von **<^s>** wird die Ausgabe angehalten, mit **<^q>** fortgesetzt.

Filemanipulation

Files werden mit einem Pfadnamen angesprochen, der im allgemeinen aus dem Namen ein oder mehrerer Directories und dem Filenamen besteht, z.B.

/u/stefan/file01

/ bezeichnet die root-Directory des Filesystems, *u* eine Directory im root-Directory, *stefan* eine Subdirectory von *u* und *file01* einen File im Directory *stefan*. Man spricht in diesem Fall von einem vollen Pfadnamen.

Die Angabe der übergeordneten Directories kann entfallen, wenn eine der Directories im vollen Pfadnamen die aktuelle Benützerdirectory ist (relativer Pfadname):

stefan/file01
file01

Als Besonderheit ist dabei zu erwähnen, daß . die aktuelle Directory und .. die unmittelbar übergeordnete Directory bezeichnet.

Manche Programme verlangen Filenamen mit bestimmten Zeichen (z.B. FORTRAN-Compiler).

Der Befehl **p** schreibt die aktuelle Benützerdirectory aus:

p
/u/stefan

Mit dem Befehl **cd** wird die aktuelle Benützerdirectory gesetzt:

cd setzt die Benützerdirectory auf jene Directory, die standardmäßig beim Login gesetzt ist.

cd dir setzt die Benützerdirectory auf *dir* , wobei *dir* eine volle oder relative Directory-Angabe sein kann.

z.B.
```
p
/u/stefan
cd libc
p
/u/stefan/libc
```

Der Befehl **u** setzt die aktuelle Benützerdirectory auf die der momentanen Directory übergeordnete Directory ("parent directory") und schreibt diese aus:

u (entspricht dem Befehl **cd ..**)

z.B.
```
p
/u/stefan/libc
u
/u/stefan
```

Mit dem Befehl **s** wird eine andere Subdirectory der übergeordneten Directory gesetzt und diese ausgeschrieben:

s dir (entspricht **cd ../dir**)

z.B.
```
p
/u/stefan/libc
s fsub
/u/stefan/fsub
```

Subdirectories der eigenen Benützerdirectory können generiert und gelöscht werden mit den Befehlen **mkdir** und **rmdir** . Eine Directory darf nur gelöscht werden, wenn sie keine Subdirectories oder Files enthält.

mkdir dir generiert die Directory *dir* in der aktuellen Benützerdirectory.

rmdir dir löscht die Directory *dir* .

Alphabetische Listen der Filenamen und der Namen der Subdirectories der aktuellen Benützerdirectory erhält man auf verschiedene Arten (Filenamen, die mit . beginnen, werden nur beim Befehl **l** gelistet):

d liefert eine mehrspaltige Liste der Namen aller Files und Subdirectories.

l liefert eine ausführliche Liste aller Files und Subdirectories (auch solcher beginnend mit .)

la listet alle Files und rekursiv auch alle Files in Subdirectories.

Eine Auswahl von Filenamen bzw. eine Auflistung der Filenamen von Subdirectories erhält man, wenn man bei den erwähnten Befehlen zum Auflisten von File- und Subdirectory-Namen als Parameter ein oder mehrere File- oder Subdirectory-Namen (getrennt durch Leerzeichen) angibt:

```
d name(s)
la name(s)
l name(s)
```

Die Verwendung von * ermöglicht dabei das Markieren von Stellen im Namen, an denen beliebige Zeichen stehen können,

z.B.	**d y***	alle Namen, die mit *y* beginnen
	la *.f *.c	alle Namen, die mit *.f* oder *.c* enden
	l *l*	alle Namen, die *l* enthalten

Der UNIX-Befehl **rm** löscht einen oder mehrere Files, wobei mehrere Filenamen durch Leerzeichen getrennt angegeben werden müssen:

rm file(s)

file ist der Pfadname eines zu löschenden Files. Wieder kann * zum Markieren von Stellen im Filenamen, an denen beliebige Zeichen stehen können, verwendet werden,

z.B. **rm *.o** alle mit *.o* endenden Object Files

Der UNIX-Command **cp** kopiert einen File:

cp file1 file2 kopiert den File *file1* auf den File *file2* .

cp file dir kopiert einen File unter Beibehaltung des Namens auf die Directory *dir*.

z.B. **cp /u/pub/profile.all .** kopiert den File *profile.all* von der Directory */u/pub* auf die aktuelle Directory.

Der UNIX-Command **mv** ermöglicht das Umbenennen von Files:

mv file1 file2 Der File mit dem Namen *file1* erhält den Namen *file2*

Filezugriffsattribute

Ausführliche Filelisten (z.B. mit den Befehlen **l** und **la**) enthalten in der ersten Spalte die Filezugriffsattribute, in der dritten und vierten Spalte den *Owner* (=Benützername) und die *User Group* (z.B. *common, RZ, ...*).

Die Filezugriffsattribute bestehen aus 10 Zeichen mit folgender Bedeutung:

Das erste Zeichen ist *d* wenn die Eintragung eine Directory ist,
\- wenn es sich um einen File handelt.

Die nächsten 9 Zeichen sind zu interpretieren als 3 Gruppen zu je 3 Zeichen. Die erste Gruppe bezeichnet die Zugriffsrechte des *Owners* , die zweite die Zugriffsrechte anderer Benützer mit der gleichen *User Group* und die dritte die Zugriffsrechte aller anderen Benützer. Innerhalb jeder Gruppe gibt das erste Zeichen die Leseerlaubnis an, das zweite die Schreiberlaubnis und das dritte Zeichen die Erlaubnis, den File als Programm zu exekutieren. (Für eine Directory bedeutet das dritte Zeichen die Erlaubnis, die Directory nach einem bestimmten File zu durchsuchen.) Die Berechtigungen sind folgendermaßen angegeben:

r File ist lesbar,
w File ist beschreibbar,
x File ist exekutierbar,
\- entsprechende Erlaubnis nicht erteilt.

Bei jeder File- bzw. Directory-Generierung wird für die Zugriffsberechtigungen ein Standardwert gesetzt, der im File *.profile* in der Benützerdirectory mit dem Befehl **umask** festgelegt wird. Der Wert kann vom Benutzer seinen Wünschen entsprechend geändert werden.

umask ooo setzt die Zugriffsberechtigungen für neue Files oder Subdirectories auf das Komplement der dreistelligen Oktalzahl *ooo* , d.h. gesetzte Bits in *ooo* bedeuten, daß die betreffende Berechtigung nicht erteilt wird.

umask schreibt den aktuellen mit **umask** gesetzten Wert der Zugriffsberechtigungen für neue Files oder Subdirectories aus.

z.B. **umask 022** Keine Schreiberlaubnis für die User Group und alle anderen Benützer.

umask 007 Alle Berechtigungen für die User Group, kein Zugriff für alle anderen Benützer.

Die Zugriffsattribute für bestehende Files und Subdirectories können nachträglich mit dem Befehl **chmod** geändert werden:

chmod mode file(s) ändert die Zugriffsattribute für die angegebenen Files oder Subdirectories entsprechend *mode* .

mode hat die Form

[who] ± permission(s) [± permission(s)]

Dabei ist *who* eine Kombination der Buchstaben **u** (*Owner*), **g** (*Group*) und **o** (*Others*). **a** (*All*) steht für **ugo** und ist Defaultwert, wenn *who* fehlt.
Für **+** wird die Erlaubnis erteilt, für **-** wird sie verweigert.
permission steht für die Buchstaben **r** (Leseerlaubnis), **w** (Schreiberlaubnis) und **x** (Exekutiererlaubnis).

z.B. **chmod o-w file** Keine Schreiberlaubnis auf dem File *file* für alle Benützer, die nicht zur gleichen User Group gehören.

chmod +x file Der File *file* darf von allen Benützern exekutiert werden.

Wichtig:
Files bzw. Subdirectories eines Benützers dürfen nur in der eigenen Benützerdirectory bzw. ihren Subdirectories angelegt werden. Finden sich Benützerfiles in einer anderen Directory als der zugewiesenen Benützerdirectory, so werden diese Files ohne Rücksprache mit dem Benützer gelöscht, außerdem wird die Benützungsberechtigung des betreffenden Benützers gesperrt.

Generieren, Editieren, Übersetzen und Binden eines Programmes

Der UNIX-Befehl **e** startet den Screen Editor **vi** , der das Generieren und Editieren von Programm-, Daten- und Text-Resourcen ermöglicht:

e [efile] erlaubt das Editieren des Files *efile* . Bei wiederholtem Aufruf kann die Angabe von efile entfallen, es wird automatisch der zuletzt angegebene File editiert.

ef [ffile] erlaubt das Editieren des FORTRAN-Files *ffile.f* . Bei wiederholtem Aufruf kann die Angabe von ffile entfallen, es wird automatisch der zuletzt angegebene FORTRAN-File editiert.

ec [cfile] erlaubt das Editieren des C-Files *cfile.c* . Bei wiederholtem Aufruf kann die Angabe von cfile entfallen, es wird automatisch der zuletzt angegebene C-File editiert.

Eine Beschreibung der Befehle des Screen Editors **vi** enthält die TUNIX vi Editor-Beschreibung.

Der UNIX-Befehl **ftn** übersetzt und bindet ein FORTRAN-Programm (FORTRAN 77 - Compiler).

ftn ffile [options]
ftn [ffile]

Das Source-Programm muß *ffile.f* heißen. Das Objekt wird unter *ffile.o* abgespeichert, das exekutierbare Programm erhält den Namen *ffile* . Mögliche Options sind etwa Files mit Unterprogrammen, die in FORTRAN-Source (Filename endend mit *.f*), in C-Source (mit *.c*) oder übersetzt (mit *.o*) vorliegen können.

z.B. **ftn fprog cs.o** Übersetzen des FORTRAN-Programms *fprog.f* und Binden des exekutierbaren Programms *fprog* unter Verwendung von bereits übersetzten Subroutinen des Object-Files *cs.o* .

Die Angabe von **ffile** kann entfallen, es wird dann das zuletzt mit **ef** oder **ftn** angesprochene FORTRAN-Programm übersetzt. In diesem Fall können dann keine Options angegeben werden.

Der UNIX-Befehl **f77** ruft den FORTRAN 77 - Compiler, der Befehl **cc** den C - Compiler auf.

f77 file(s)
cc file(s)

Beide Compiler können die entsprechenden Source Programme übersetzen (Filenamen endend mit *.f* bzw. *.c*) und direkt ein exekutierbares Programm mit dem Namen *a.out* erzeugen. Es können auch bereits übersetzte Programme (Filenamen endend mit *.o*) angegeben werden, in diesem Fall erfolgt dann keine Übersetzung mehr. Werden mehrere Filenamen angegeben, so sind diese mit Leerzeichen zu trennen. Eine detaillierte Beschreibung von **f77** und **cc** findet man im *UNIX User Reference Manual* .

Rechnen eines Programmes

Ein exekutables Programm wird durch Angabe seines Namens *pgname* gestartet.

pgname

Listen von Files

Der Befehl **print** listet einen oder mehrere Files auf dem TUNIX-Printer *prl* . Mehrere Filenamen müssen durch Leerzeichen getrennt werden.

print file(s)

Es wird dem Benutzer die automatisch zugeordnete Print Request-Id mitgeteilt.

Mit dem Befehl

lps

kann man sich einen Überblick über die Printer-Queue verschaffen und nachsehen, ob die Liste mit der ausgeschriebenen Print Request-Id schon gedruckt ist.

Auf dem Benützerterminal können ein oder mehrere Files durch

c file(s)

gelistet werden. Mehrere Filenamen müssen durch Leerzeichen getrennt werden.

Informationen über die Command-Makros und UNIX-Commands

Der Befehl **h** listet alle im File *.profile* der Benützerdirectory definierten Command-Makros mit einer kurzen Erklärung auf:

h

Der UNIX-Befehl **man** liefert genaue Informationen über einen Befehl *cmd*, indem er die entsprechende Befehlsbeschreibung aus dem Manual auf dem Bildschirm auflistet:

man cmd

Es wird eine Bildschirmseite der Befehlsbeschreibung aufgelistet und dann die Eingabe eines Befehlszeichen zur Steuerung der weiteren Auflistung erwartet. Eine Liste der möglichen Befehlszeichen erhält man, wenn man **h** eintippt. Der Befehl wird mit **q** beendet.

Alle für den Benutzer wichtigen Befehle sind ausführlich im *UNIX User Reference Manual* beschrieben.

Beenden der Terminal-Session

Die Eingabe von <^d> statt eines Befehls beendet eine Terminal Session am TUNIX. Es wird wieder *login:* für eine neue Session ausgeschrieben.

Das TUNIX-System ist verlassen, die TUNET-Verbindung ist noch aufrecht. Man könnte durch neuerliche Eingabe eines Benützernamens wieder eine Terminal Session beginnen.

Durch Eingabe von <^_> gelangt man wieder in den Terminal Server, der sich mit

hybl>

meldet. Zur Beendigung der Terminal Session muß

dc<CR>

eingegeben werden. Die Rechnerverbindung wird für andere Benützer freigegeben. Der Server schreibt z.B. aus:

Disconnecting ... session 1 -- disconnected from tunix(....)

EDV - ZENTRUM TECHNISCHE UNIVERSITÄT WIEN Hybridrechenanlage	TUNIX	
	Screen Editor **vi**	
	1. Auflage, Februar 1988	FB

TUNIX Screen Editor vi

Der UNIX Screen Editor **vi** ermöglicht das Generieren und Manipulieren von Textfiles. Am Benutzerterminal wird jeweils ein bestimmter Ausschnitt des Textfiles dargestellt. Innerhalb dieses Ausschnitts kann Text hinzugefügt, gelöscht oder geändert werden, wobei die Modifikationen direkt am Bildschirm wiedergegeben werden und somit mitverfolgt werden können. Die Änderungen erfolgen nicht direkt am File, sondern werden erst beim Beenden des **vi** tatsächlich auf dem bearbeiteten Textfile durchgeführt. Man kann aber den geänderten Text auch auf einen anderen File abspeichern oder den Editor abbrechen, ohne die Textänderungen abzuspeichern.

Notation

Es gibt Befehle des Editors **vi** , die nicht am Bildschirm ausgeschrieben werden. Diese Befehle sind in der folgenden Beschreibung in spitzen Klammern < > eingeschlossen. **<CR>** bedeutet beispielsweise Carriage Return.

Der Editor kennt auch Befehle, die Control Character sind (Control-Taste gemeinsam mit einer anderen Taste). Diese werden mit vorangestelltem ^-Zeichen angegeben, z.B. **^d** . Da der Befehl **^d** nicht auf dem Bildschirm erscheint, wird er im folgenden in der Form **<^d>** geschrieben.

Zeilennummern

Da **vi** ein Screen Editor ist, sind die Zeilennummern von untergeordneter Bedeutung. Es gibt aber einige Befehle, bei denen Zeilennummern angegeben werden können. Die Zeilen des Textfiles werden implizit, beginnend mit Zeilennummer 1, 2, 3, usw. durchnumeriert. Die Zeilennummer einer Zeile ist also nicht fix, sondern kann sich durch Einfügen oder Löschen vorangehender Textzeilen laufend ändern.

Terminal-Konfiguration

Bevor der Screen Editor gestartet werden kann, muß dem Betriebssystem der Typ des Terminals, auf dem editiert werden soll, mitgeteilt werden. Dies geschieht üblicherweise durch Befehle im File *.profile* , die bei jedem Login durchgeführt werden. Den eingestellten Terminaltyp erhält man mit dem Befehl

```
echo $TERM
```

Sollte der ausgeschriebene Terminaltyp nicht zutreffen, wenden Sie sich bitte an einen Mitarbeiter der Abt. Hybridrechenanlage, der Ihnen gerne bei der erforderlichen Einstellung behilflich ist.

STARTEN DES SCREEN EDITORS vi

Beim Starten des Screen Editors ist der Name des Files anzugeben, der editiert bzw. neu generiert werden soll:

vi file

Da man üblicherweise im Zuge einer Terminal-Session immer wieder den gleichen File editiert (z.B. bei einer Programmerstellung), gibt es im File *.profile* die folgenden Shell Scripts zur einfachen Verwendung des **vi** :

e [efile]	erlaubt das Editieren des Files *efile* . Bei wiederholtem Aufruf kann die Angabe von efile entfallen, es wird automatisch der zuletzt angegebene File editiert.
ef [ffile]	erlaubt das Editieren des FORTRAN-Files *ffile.f* . Bei wiederholtem Aufruf kann die Angabe von ffile entfallen, es wird automatisch der zuletzt angegebene FORTRAN-File editiert.
ec [cfile]	erlaubt das Editieren des C-Files *cfile.c* . Bei wiederholtem Aufruf kann die Angabe von cfile entfallen, es wird automatisch der zuletzt angegebene C-File editiert.

Der **vi** Editor löscht den Bildschirm und listet den Beginn des angegebenen Files auf. Enthält der File weniger Zeilen als der Bildschirm, so wird auf den restlichen Bildschirmzeilen jeweils ein ~-Zeichen ausgeschrieben. In der letzten Zeile erfolgt eine Ausgabe der Form

"file" xx lines, xxx characters

Wird beim Starten des **vi** ein nicht existierender Filename angegeben, dann ist dieser File vorerst leer, es wird in jeder Zeile des Bildschirms ein ~-Zeichen ausgeschrieben und in der letzten Zeile erscheint die Information

"file" [New file]

Der Cursor ist am oberen linken Bildschirmrand positioniert, und **vi** wartet auf die Eingabe des ersten Befehls.

BEFEHLE DES SCREEN EDITORS vi

Die im folgenden beschriebenen Befehle stellen nicht den gesamten Befehlssatz des **vi** dar. Es werden auch bei einzelnen Befehlen nicht alle möglichen Options angeführt. Die Einschränkungen bedeuten jedoch keine Beeinträchtigung bei der Benutzung des Editors, sie wurden ausschließlich im Interesse einer einfachen, kurzen Beschreibung getroffen.

Der Screen Editor **vi** unterscheidet drei verschiedene Eingabe-Modes:

Command Mode: Nach dem Starten befindet man sich im *Command Mode* . Die eingetippten Befehle selbst werden auf dem Bildschirm nicht ausgegeben, es werden nur der dargestellte Ausschnitt des Files bzw. die Cursor-Position entsprechend verändert.
Aus den beiden anderen Modes kommt man zurück in den *Command Mode* durch die Eingabe eines Escape-Zeichens **<ESC>** oder nach Beendigung eines Befehls, der das Verlassen des *Command Modes* bewirkt hat.

Text Input Mode: In den *Text Input Mode* (auch *Append Mode* genannt) gelangt man durch die Eingabe von Befehlen, die Text zum bearbeiteten File hinzufügen bzw. Text ersetzen.
Durch die Eingabe eines Escape-Zeichens **<ESC>** gelangt man wieder in den *Command Mode* . Antwortet der **vi** auf **<ESC>** mit einem Piepsen, so befindet man sich bereits wieder im *Command Mode* . (Auf den Text des Files haben überzählige Escape-Zeichen keinen Einfluß.)

Last Line Mode: In diesem Mode befindet man sich, wenn man Befehle eingibt, die mit *: / ?* oder *!* beginnen. Das eingetippte Zeichen wird in der letzten Zeile des Bildschirms ausgeschrieben, wo auch die weiteren Zeichen des Befehls eingelesen und ausgeschrieben werden. Nach dem abschließenden **<CR>** wird der Befehl ausgeführt und wieder der *Command Mode* gesetzt.

Positionieren des Cursors innerhalb des dargestellten Ausschnitts

Um den Text editieren zu können, muß man den Cursor an die Stelle (Zeile und/oder Spalte) positionieren, wo man mit dem Modifizieren beginnen möchte.

Zeichen- bzw. zeilenweises Positionieren

<h> oder **<BS>**	Der Cursor wandert ein Zeichen nach links.
<j>	Der Cursor wandert eine Zeile nach unten.
<k>	Der Cursor wandert eine Zeile nach oben.
<l> oder **<space>**	Der Cursor wandert ein Zeichen nach rechts.
<+> oder **<CR>**	Der Cursor wandert an den Beginn der nächsten Zeile.
<->	Der Cursor wandert an den Beginn der vorangegangenen Zeile.

Mehrmaliges Drücken einer Taste bewegt den Cursor mehrere Zeichen bzw. Zeilen in der betreffenden Richtung. Wenn der Cursor nicht mehr weiter bewegt werden kann (z.B. Ende der Zeile), ertönt ein Piepsen.

Jedem der angeführten Befehlszeichen kann eine Zahl n vorangestellt werden, z.B. <7h> oder <13+> . Der Cursor wird dann um n Zeichen oder Zeilen versetzt, also um 7 Zeichen nach links oder um 13 Zeilen nach unten.

Positionieren am Zeilenende oder Zeilenbeginn

<$> Der Cursor wird auf das letzte Zeichen der Zeile positioniert.

<0> Die Ziffer 0 positioniert den Cursor auf das erste Zeichen der Zeile.

<^> Das ^-Zeichen (kein Control Character!) positioniert den Cursor auf das erste Zeichen der Zeile, das kein Blank ist.

Positionieren am Bildschirm

<H> Der Cursor wird auf die erste Zeile des Bildschirms (= erste Zeile des dargestellten Ausschnitts des Textfiles) positioniert ("Home").

<M> Der Cursor wird auf die Textzeile in der Mitte des Bildschirms positioniert.

<L> Der Cursor wird auf die letzte Zeile des Bildschirms positioniert.

Positionieren des Cursors innerhalb des Files

Scrolling

<^f> Es wird der nachfolgende Ausschnitt des Files auf dem Bildschirm dargestellt, wobei die beiden letzten Zeilen des Ausschnitts die ersten Zeilen des neuen Ausschnitts werden. Sind auf dem File nicht mehr genügend Zeilen vorhanden, so wird auf den restlichen Bildschirmzeilen jeweils ein ~-Zeichen ausgeschrieben.

<^b> Es wird der vorangehende Ausschnitt des Files auf dem Bildschirm dargestellt, wobei allerdings keine Zeilen des Ausschnitts im neuen Ausschnitt wieder aufscheinen. Sind auf dem File nicht mehr genügend vorangehende Zeilen vorhanden, so wird der Ausschnitt nur bis zur ersten Zeile des Files nach vorne verschoben.

<^d> Es wird der dargestellte Ausschnitt des Textfiles Zeile für Zeile um eine halbe Bildschirmseite nach hinten verschoben.

<^u> Es wird der dargestellte Ausschnitt des Textfiles Zeile für Zeile um eine halbe Bildschirmseite nach vorne verschoben.

Positionieren einer bestimmten Zeile

Mit dem Go-Befehl **<G>** kann eine bestimmte Zeile positioniert werden. Liegt die Zeile im dargestellten Ausschnitt, so wird der Cursor auf die Zeile positioniert, anderenfalls ein neuer Ausschnitt um die betreffende Zeile herum dargestellt.

<G>	Es wird die letzte Zeile des Files positioniert.
<nG>	Es wird die n-te Zeile des Files positioniert.

Die Nummer einer Zeile erhält man, wenn man den Cursor auf die Zeile positioniert und den Befehl **<^g>** eintippt.

<^g>	In der letzten Zeile des Bildschirms erscheint eine Statuszeile mit dem Filenamen, gegebenenfalls Angabe, ob die Zeile geändert wurde, die Zeilennummer n, die Gesamtanzahl der Zeilen und eine Prozentangabe, welchen Anteil die ersten n Zeilen an der Gesamtzeilenzahl haben.

z.B. *Diese Zeile ist die 35. Zeile des Files.*
Der Cursor befindet sich in dieser Zeile.
↑ **<^g>**
Statuszeile:
"file" [Modified] line 36 of 116 --34%--

Positionieren einer bestimmten Zeichenkette

Mit den Suchbefehlen / und ? kann eine bestimmte Zeichenkette, die auch Blanks enthalten darf, gesucht werden. Dabei wird der Editor **vi** in den *Last Line Mode* versetzt. Die eingetippten Zeichen werden in der letzten Zeile des Bildschirms ausgeschrieben. Nach dem abschließenden **<CR>** wird der Befehl ausgeführt und wieder der *Command Mode* gesetzt.

/text<CR>	Vorwärtsgehend von der momentanen Cursor-Position wird das nächste Vorkommen von *text* gesucht. Der Cursor wird auf das erste Zeichen von *text* positioniert.
?text<CR>	Rückwärtsgehend von der momentanen Cursor-Position wird das nächste Vorkommen von *text* gesucht. Der Cursor wird auf das erste Zeichen von *text* positioniert.
<n>	Der letzte Suchbefehl wird wiederholt.
<N>	Der letzte Suchbefehl wird in der entgegengesetzten Richtung wiederholt.

Die angegebene Zeichenkette wird nur gefunden, wenn sie sich zur Gänze auf einer Zeile befindet (auch wenn *text* aus mehr als einem Wort besteht).

Gelangt man bei der Suche an das Ende des Files, so wird der File vom Beginn weg weiter durchsucht (*wrap around*). Bei der Rückwärtssuche wird ebenso verfahren. Bei Erreichen des Filebeginns wird die Suche von der letzten Zeile an rückwärts fortgesetzt.

Generieren von Text

Die Eingabe von neuem Text ist mit dem Append-Befehl **<a>** , dem Insert-Befehl **<i>** und dem Open-Befehl **<o>** möglich. Jeder dieser Befehle versetzt den **vi** in den *Text Input Mode* . Tippfehler bei der Texteingabe können mit **<BS>** (Backspace) korrigiert werden. **<@>** löscht den gesamten bis dahin eingetippten Text. Die Eingabe von **<CR>** im eingefügten Text schließt die laufende Zeile ab und setzt die nachfolgenden Zeichen in eine neue Zeile. So können mit einem Befehlsaufruf gleich mehrere neue Zeilen erzeugt werden.

<a> Nachfolgend eingetippte Zeichen werden rechts bzw. nach der momentanen Cursor-Position eingefügt.

<A> Nachfolgend eingetippte Zeichen werden am Ende der momentanen Zeile (= Zeile, in der sich der Cursor befindet) eingefügt.

z.B. *Einfügen von Text nach dem h von hier (Append).*
↑ **<a>**

Einfügen von Text nach dem h von h⌐→ier (Append).
└ eingefügter Text

(↑ bezeichnet die Cursor-Position.)

<i> Nachfolgend eingetippte Zeichen werden links bzw. vor der momentanen Cursor-Position eingefügt.

<I> Nachfolgend eingetippte Zeichen werden am Beginn der momentanen Zeile eingefügt.

z.B. *Einfügen von Text vor dem h von hier (Insert).*
↑ **<i>**

Einfügen von Text vor dem h von ⌐→hier (Insert).
└ eingefügter Text

<o> Nachfolgend eingetippte Zeichen werden am Beginn einer neuen Zeile unterhalb bzw. nach der momentanen Zeile eingefügt. Der Cursor kann sich in einer beliebigen Spalte der Zeile befinden.

<O> Nachfolgend eingetippte Zeichen werden am Beginn einer neuen Zeile oberhalb bzw. vor der momentanen Zeile eingefügt. Der Cursor kann sich in einer beliebigen Spalte der Zeile befinden.

z.B. *Einfügen einer neuen Zeile unterhalb der momentanen Zeile.*
↑ **<o>** (Cursor auf beliebigem Zeichen)

Einfügen einer neuen Zeile unterhalb der momentanen Zeile.
⌐ - - - - - - - - ┘
└→ eingefügte Zeile

<ESC> Durch die Eingabe eines Escape-Zeichens gelangt man aus dem *Text Input Mode* wieder in den *Command Mode* . Antwortet der **vi** auf **<ESC>** mit einem Piepsen, so befindet man sich bereits wieder im *Command Mode* . (Auf den Text des Files haben überzählige Escape-Zeichen keinen Einfluß.)

Rückgängigmachen des letzten Befehls

Der Undo-Befehl **<u>** ist besonders bei einem irrtümlich gegebenen Befehl von Nutzen. Damit kann man etwa eine gelöschte Zeile wieder in den File zurückholen. Ein zweiter Undo-Befehl hebt das erste Undo wieder auf, d.h. eine gelöschte und dann wieder zurückgeholte Zeile wird wieder gelöscht.

<u> Der letzte Befehl wird rückgängig gemacht.

Löschen von Text

Mit dem Befehl **<x>** können einzelne Zeichen des Textfiles gelöscht werden. Der Delete-Befehl **<d>** erlaubt das Löschen von ganzen Textobjekten bzw. Textzeilen.

<x> Das Zeichen an der Cursor-Position wird gelöscht.

<nx> Ausgehend von der Cursor-Position werden n Zeichen gelöscht.

z.B. *Löschen von sechs Zeichen ab der Cursor-Position*
↑ **<6x>**

Löschen von Zeichen ab der Cursor-Position
↑

<dw> Ausgehend von der Cursor-Position werden alle bzw. die restlichen Zeichen des Wortes und auch das nachfolgende Blank oder Interpunktionszeichen gelöscht.

z.B. *Dieser Befehl löschtauch einen Teil eines Wortes.*
↑ **<dw>**

Dieser Befehl löschteinen Teil eines Wortes.
↑

<ndw> Es werden die nachfolgenden n Worte gelöscht. Der Cursor muß am Beginn des ersten zu löschenden Wortes positioniert sein.

z.B. *So können ein oder mehrere Worte gelöscht werden.*
↑ **<2dw>**

So können mehrere Worte gelöscht werden.
↑

<dd> Die momentane Textzeile wird gelöscht. Der Cursor kann sich in einer beliebigen Spalte der Zeile befinden.

<ndd> Ausgehend von der momentanen Textzeile werden n Zeilen gelöscht.
Wenn gelöschte Zeilen außerhalb des dargestellten Ausschnitts liegen, wird in der letzten Zeile die Nachricht *n lines deleted* ausgeschrieben.
Enthält der File weniger als n Zeilen ab der Cursor-Position, so ertönt ein Piepsen und es werden keine Zeilen gelöscht.

<D> löscht den Rest einer Zeile von der Cursor-Position bis zum Ende der Zeile.

Verändern von Text

Textänderungen können durchgeführt werden mit dem Replace-Befehl **<r>** , dem Substitute-Befehl **<s>** und dem Change-Befehl **<c>** . Jeder dieser Befehle löscht Text und generiert neuen Text in einem.

Replace überschreibt Zeichen:

<r> Das Zeichen an der momentanen Cursor-Position wird durch das unmittelbar anschließend eingetippte Zeichen überschrieben.

z.B. *Geeignet zur Korrectur einzelner Zeichen*
↑ **<r>k**

Geeignet zur Korrektur einzelner Zeichen
↑

<R> Beginnend an der momentanen Cursor-Position werden Zeichen durch unmittelbar anschließend eingetippte Zeichen überschrieben, bis **<ESC>** getippt wird. Bei Erreichen des letzten Zeichens auf einer Zeile wird neuer Text angefügt.

Substitute ersetzt Zeichen durch mehr oder durch weniger Zeichen:

<s> Das Zeichen an der momentanen Cursor-Position wird gelöscht und durch die unmittelbar anschließend eingetippten Zeichen ersetzt, bis **<ESC>** getippt wird.

<ns> n Zeichen (von der Cursor-Position ausgehend) werden gelöscht und durch die unmittelbar anschließend eingetippten Zeichen ersetzt, bis **<ESC>** getippt wird.

<S> Alle Zeichen der Zeile (unabhängig von der Cursor-Position) werden durch die nachfolgend eingetippten Zeichen ersetzt, bis **<ESC>** getippt wird.

Der Substitute-Befehl markiert das jeweils letzte Zeichen, das durch den Befehl ersetzt wird, mittels *$* . Die zu ersetzenden Zeichen werden nicht sofort vom Bildschirm gelöscht, sondern erst wenn man sie übertippt oder den Befehl mit **<ESC>** abschließt.

z.B. *Ersetzen mehrerer Zeichen einer Zeile*
↑ **<2s>**

Ersetze$ mehrerer Zeichen einer Zeile
↑ **ung<ESC>**

Ersetzung mehrerer Zeichen einer Zeile
↑

Change ersetzt Textobjekte, wie etwa Worte:

<cw>	Ausgehend von der Cursor-Position werden alle bzw. die restlichen Zeichen eines Wortes gelöscht und durch die unmittelbar anschließend eingetippten Zeichen ersetzt, bis **<ESC>** getippt wird.
<ncw>	Es werden die nachfolgenden n Worte ersetzt. Der Cursor muß am Beginn des ersten zu ersetzenden Wortes positioniert sein.
<C>	Alle restlichen Zeichen der Zeile (von der Cursor-Position an) werden durch die nachfolgend eingetippten Zeichen ersetzt, bis **<ESC>** getippt wird.

Der Change-Befehl markiert ebenfalls das jeweils letzte Zeichen, das durch den Befehl ersetzt wird, mittels *$* . Die zu ersetzenden Zeichen werden nicht sofort vom Bildschirm gelöscht, sondern erst wenn man sie übertippt oder den Befehl mit **<ESC>** abschließt.

z.B. *Ersetzen einzelner Worte einer Zeile*
↑ **<cw>**

Ersetzen einzelne$ Worte einer Zeile
↑ **es oder mehrerer<ESC>**

Ersetzen eines oder mehrerer Worte einer Zeile
↑

Globale Textänderungen

Soll ein bestimmter Text, der mehrmals im File vorkommt, durch einen neuen Text ersetzt werden, so kann diese Textersetzung auf einfache Weise global im ganzen File durchgeführt werden. Im Screen Editor **vi** können nämlich bestimmte Line Editor-Befehle verwendet werden, die jeweils mit : beginnen und im *Last Line Mode* eingegeben werden. Der Cursor geht nach der Eingabe von : automatisch in die letzte Zeile des Bildschirms, wo der Befehl eingetippt und mit **<CR>** abgeschlossen wird.

Für eine globale Textersetzung ist die Kombination zweier Line Editor-Befehle notwendig, die im folgenden zuerst einzeln und dann in Kombination beschrieben werden.

:g/text<CR> Es wird das erste Vorkommen von *text* in einer Zeile gesucht.

:s/text/newtext/<CR>
Ersetzung des ersten Vorkommens von *text* in der Zeile durch *newtext* .

:s/text/newtext/g<CR>
Durch Anhängen von *g* an den vorigen Befehl wird jedes Vorkommen von *text* in der Zeile durch *newtext* ersetzt.

:g/text/s//newtext/g<CR>
Dieser Befehl ersetzt *text* durch *newtext* im gesamten File. Es wird *text* gesucht und durch *newtext* ersetzt, danach das nächste Vorkommen von *text* gesucht und so weiter. Beim Ersetzungsbefehl *s* braucht *text* nicht angeführt zu werden, es ist noch vom Suchbefehl *g* bekannt.

Verschieben von Text

Text kann dadurch innerhalb des Textfiles verschoben werden, daß er zuerst an der ursprünglichen Stelle gelöscht und dann an der neuen Stelle eingefügt wird. Der jeweils letzte gelöschte Text (z.B. Zeichen, Worte oder ganze Textzeilen) wird in einem temporären Puffer gespeichert. Wenn unmittelbar nach dem Löschen des Textes der Cursor an die Stelle positioniert wird, an die der Text verschoben werden soll, dann kann anschließend durch den Put-Befehl der Pufferinhalt dort eingefügt werden.

Text kann auch, ohne daß er gelöscht wird, in den Puffer geschrieben werden und mittels des Put-Befehls eingefügt werden. Mehr darüber im nächsten Abschnitt über das Kopieren von Text.

<p> Der Text im temporären Puffer wird nach der Cursor-Position eingefügt.

z.B. *Einfügen des gelöschten letzten Textes*
↑ **<dw>**

Einfügen des gelöschten Textes
↑ **<p>**

Einfügen des letzten gelöschten Textes
↑

<np> Der Text im temporären Puffer wird nach der Cursor-Position n-mal eingefügt.

<xp> Vertauschen zweier Zeichen. Das Zeichen an der Cursor-Position wird gelöscht und nach dem folgenden Zeichen wieder eingefügt.

Kopieren von Text

Text kann kopiert werden, indem er zuerst mit dem Yank-Befehl in den temporären Puffer kopiert wird. Danach muß der Cursor an die Stelle positioniert werden, wohin der Text kopiert werden soll. Anschließend kann dann durch den Put-Befehl der Pufferinhalt dort eingefügt werden (siehe vorigen Abschnitt über das Verschieben von Text). Mit dem Yank-Befehl können (genau wie mit dem Delete-Befehl) Worte oder ganze Textzeilen in den Puffer geschrieben werden.

<yw>	Ausgehend von der Cursor-Position werden alle bzw. die restlichen Zeichen des Wortes und auch das nachfolgende Blank oder Interpunktionszeichen in den Puffer geschrieben.
<nyw>	Es werden die nachfolgenden n Worte in den Puffer geschrieben. Der Cursor muß am Beginn des ersten Wortes positioniert sein.
<yy>	Die momentane Textzeile wird in den Puffer geschrieben. Der Cursor kann sich in einer beliebigen Spalte der Zeile befinden.
<nyy>	Ausgehend von der momentanen Textzeile werden n Zeilen in den Puffer geschrieben. Wenn angesprochene Zeilen außerhalb des dargestellten Ausschnitts liegen, wird in der letzten Zeile die Nachricht *n lines yanked* ausgeschrieben. Enthält der File weniger als n Zeilen ab der Cursor-Position, so ertönt ein Piepsen und es werden keine Zeilen in den Puffer geschrieben.

Einfügen eines Textfiles

Der Inhalt eines bestehenden Textfiles kann in den bearbeiteten Text eingefügt werden.

:r file<CR>	Es werden die Textzeilen des Files *file* nach der Zeile, in der sich der Cursor befindet, eingefügt.

Abspeichern von Textzeilen

Eine einzelne Textzeile oder ein Bereich von Textzeilen können auf einen neuen Textfile gespeichert werden.

:.w file<CR>	Die momentane Textzeile wird auf einen neuen Textfile *file* gespeichert.
:x,yw file<CR>	Die Textzeilen mit den Zeilennummern x bis y werden auf einen neuen Textfile *file* gespeichert.

Spezielle Befehle

Im folgenden werden einige spezielle Befehle beschrieben, die bei der Verwendung des Screen Editors **vi** nützlich sind.

:sh<CR>	Aus dem Screen Editor wird vorübergehend in das Betriebssystem umgestiegen. Man kann dort Systembefehle durchführen, Programme rechnen und sogar einen anderen File mit **vi** editieren.
<^d>	Das Betriebssystem wird verlassen und man kehrt wieder auf die gleiche Zeile des Textfiles im Screen Editor zurück.
<.>	Die Eingabe eines Punktes im *Command Mode* wiederholt die letzte Textänderung (d.h. Ersetzung, Einfügung, Generierung oder Löschung). Dieser Befehl ist praktisch für Korrekturen in Verbindung mit dem Suchbefehl (Positionieren einer bestimmten Zeichenkette).
<J>	Die nachfolgende Textzeile wird an die momentane Textzeile angehängt, wobei dazwischen automatisch ein Leerzeichen eingefügt wird.
<\>	Prefix für die Texteingabe von Zeichen, die im *Text Input Mode* sonst eine spezielle Bedeutung haben. @ beispielsweise ist im Text als \@ einzugeben.
<^l>	Der Bildschirm wird gelöscht und der aktuelle Ausschnitt des Textfiles neu dargestellt. Dies empfiehlt sich dann, wenn der Text am Bildschirm aus irgendeinem Grund (teilweise) zerstört ist.

BEENDEN DES SCREEN EDITORS vi

Vor dem Ausstieg aus dem Screen Editor **vi** muß festgelegt werden, was mit dem bearbeiteten Text geschehen soll. Je nachdem, ob die Änderungen tatsächlich auf dem bearbeiteten Textfile durchgeführt werden sollen oder nicht bzw. ob der geänderte Text auf einen anderen File abgespeichert werden soll, gibt es verschiedene Befehle, den Screen Editor zu beenden.

<ZZ>	Die Textänderungen werden auf dem bearbeiteten Textfile durchgeführt und der Screen Editor beendet.
:q!<CR>	Der Screen Editor wird ohne Abspeicherung der Textänderungen beendet, der bearbeitete Textfile bleibt unverändert.
:w file<CR> **:q<CR>**	Der geänderte Text wird auf einen neuen Textfile *file* gespeichert und der Screen Editor beendet. Der File *file* darf noch nicht existieren.
:w! file<CR> **:q<CR>**	Der geänderte Text wird auf den Textfile *file* gespeichert und der Screen Editor beendet, wobei ein bestehender File *file* überschrieben wird.

BEFEHLSÜBERSICHT

Positionieren des Cursors

Zeichen- bzw. zeilenweises Positionieren

<h> oder <BS>	Cursor nach links
<j>	Cursor nach unten
<k>	Cursor nach oben
<l> oder <space>	Cursor nach rechts
<+> oder <CR>	Cursor an den Beginn der nächsten Zeile
<->	Cursor an den Beginn der vorangegangenen Zeile

Positionieren am Zeilenende oder Zeilenbeginn

<$>	Cursor auf das letzte Zeichen der Zeile
<0>	Cursor auf das erste Zeichen der Zeile
<^>	Cursor auf das erste Zeichen der Zeile ungleich Blank

Positionieren am Bildschirm

<H>	Cursor auf die erste Zeile des Bildschirms
<M>	Cursor auf die Zeile in der Mitte des Bildschirms
<L>	Cursor auf die letzte Zeile des Bildschirms

Scrolling

<^f>	Verschiebung um ganze Bildschirmseite nach hinten
<^b>	Verschiebung um ganze Bildschirmseite nach vorne
<^d>	Verschiebung um halbe Bildschirmseite nach hinten
<^u>	Verschiebung um halbe Bildschirmseite nach vorne

Positionieren einer bestimmten Zeile

<G>	Positionierung der letzten Zeile des Files
<nG>	Positionierung der n-ten Zeile des Files
<^g>	Ausschreiben der Zeilennummer

Positionieren einer bestimmten Zeichenkette

/text<CR>	Suche nach *text* in Vorwärtsrichtung
?text<CR>	Suche nach *text* in Rückwärtsrichtung
<n>	Wiederholung des letzten Suchbefehls
<N>	Wiederholung des letzten Suchbefehls in entgegengesetzter Richtung

Generieren von Text

<a>	Einfügen von Text nach dem Cursor
<A>	Einfügen von Text am Ende der Zeile
<i>	Einfügen von Text vor dem Cursor
<I>	Einfügen von Text am Beginn der Zeile
<o>	Einfügen einer neuen Zeile unterhalb des Cursors
<O>	Einfügen einer neuen Zeile oberhalb des Cursors
<ESC>	Beenden der Zeicheneingabe im *Text Input Mode*

Rückgängigmachen des letzten Befehls

<u>	Letzter Befehl wird rückgängig gemacht.

Löschen von Text

<x>	Löschen des Zeichen an der Cursor-Position
<nx>	Löschen von n Zeichen ab der Cursor-Position
<dw>	Löschen des Wortes ab der Cursor-Position
<ndw>	Löschen von n Worten ab der Cursor-Position
<dd>	Löschen der momentanen Zeile
<ndd>	Löschen von n Zeilen ab der momentanen Zeile
<D>	Löschen der restlichen Zeile ab der Cursor-Position

Verändern von Text

<r>	Überschreiben des Zeichens an der Cursor-Position
<R>	Überschreiben mehrerer Zeichen ab der Cursor-Position
<s>	Ersetzung des Zeichens an der Cursor-Position
<ns>	Ersetzung von n Zeichen ab der Cursor-Position
<S>	Ersetzung der gesamten momentanen Zeile
<cw>	Ersetzung des Wortes ab der Cursor-Position
<ncw>	Ersetzung von n Worten ab der Cursor-Position
<C>	Ersetzung der restlichen Zeile ab der Cursor-Position
:s/text/ntext/<CR>	Ersetzung des ersten *text* in der Zeile durch *ntext*
:s/text/ntext/g<CR>	Ersetzung von *text* durch *ntext* in der ganzen Zeile
:g/text/s//ntext/g<CR>	Ersetzung von *text* durch *ntext* im gesamten File

Verschieben von Text

<p>	Einfügen des temporären Puffers nach dem Cursor
<np>	n-maliges Einfügen des Puffer nach dem Cursor
<xp>	Vertauschen des Zeichens mit dem folgenden Zeichen

Kopieren von Text

<yw>	Speichern des Wortes ab der Cursor-Position im Puffer
<nyw>	Speichern von n Worten ab der Cursor-Position
<yy>	Speichern der momentanen Zeile im Puffer
<nyy>	Speichern von n Zeilen ab der momentanen Zeile

Einfügen eines Textfiles

:r file<CR>	Einfügen des Textfiles *file* nach der momentanen Zeile

Abspeichern von Textzeilen

:.w file<CR>	Speichern der momentanen Zeile auf neuen File *file*
:x,yw file<CR>	Speichern der Zeilen x bis y auf neuen File *file*

Spezielle Befehle

:sh<CR>	Vorübergehendes Umsteigen ins Betriebssystem
<^d>	Rückkehr in den Screen Editor
<.>	Wiederholung der letzten Textänderung
<J>	Zusammenhängen der Zeile mit der nachfolgenden Zeile
<\>	Prefix für die Eingabe von @
<^l>	Löschen und Neuschreiben des Bildschirms

Beenden des Screen Editors vi

<ZZ>	Abspeichern des (geänderten) Textes und Ende des vi
:q!<CR>	Beenden ohne Abspeicherung des (geänderten) Textes
:w file<CR>	Abspeichern des Textes auf neuen Textfile *file*
:w! file<CR>	Abspeichern des Textes auf (bestehenden) File *file*
:q<CR>	Beenden des **vi** nach Abspeicherung des Textes