Ein moderner Editor für UNIX -kompatible Betriebssysteme

1. Einleitung

Dies ist eine Beschreibung des 1988 von Jens -Peter Redlich entwickelten Editor fse (full-screen-editor).
Er ist voll- ständig in C implementiert und mit relativ geringem Aufwand auf UNIX-kompatible Betriebssysteme portierbar.

Bei seiner Entwicklung wurden folgende Schwerpunkte gesetzt:

- Ausnutzung des gesamten Bildschirmes für die Textdarstellung (d.h. 24x80 bzw. 25x80 Zeichen)
- Änderungen am Text müssen sofort auf dem Bildschirm sichtbar werden
- die Grundfunktionen müssen leicht erlernbar sein, só daß bereits nach kurzer Einarbeitungszeit alle anfallenden Aufgaben bearbeitet werden können
- parallele Verarbeitung von 2 Dateien in 2 Fenstern
- keine Forderungen an die Struktur der Datei. Das bedeutet, daß Texte mit beliebig langen Zeilen, sowie beliebige Nicht-ASCII Dateien bearbeitet werden können

Weiter war eine relative Hardwareunabhängigkeit bezüglich des verwendeten Bildschirms und der Tastatur gefordert. Derzeit erfolgt die Anpassung beim Programmstart, so daß fse auch auf Rechnern mit mehreren verschiedenen Terminals problemlos arbeitet. Obwohl vorrangig für die Bearbeitung von Quelltexten und Nicht-ASCII-Dateien vorgesehen, läßt sich fse auch sinnvoll für die Textverarbeitung einsetzen. Zusätzlich wird dem Anwender durch eine Vielzahl praktischer Standardaktionen die Arbeit mit dem Editor erleichtert (z.B. automatisches Einrücken, Wiederladen der zuletzt bearbeiteten Datei usw.).

2. Übersicht

fse stellt Funktionen zum Editieren beliebiger Dateien bereit. Elementare Textverarbeitung wird ebenso unterstützt, wie die Verarbeitung von Nicht-ASCII-Dateien oder die Bearbeitung von Quelltexten. Es ist jederzeit möglich, ein zweites Editorfenster für eine weitere Datei zu eröffnen und beide Dateien unabhängig voneinander zu bearbeiten. Umfangreiche Blockkommandos und die Möglichkeit der Definition von Metakeys helfen die Arbeit mit dem Editor zu vereinfachen. Für die Korrektur von Syntaxfehlern in Quelltexten stehen für die Sprachen C und Modula-2 Kopplungen von Compiler und Editor zur Verfügung, so daß dem Anwender beide Programme als eine Einheit erscheinen.

Zusammenstellung wichtiger Kommandos in der Helpfunktion (^QH)

^E up	^R start of page	^QS start of line	^QP goto line
^S left	^C end of page	^QD end of line	^QU exchange locat
^X down	^QR start of file	^A word left	^QB goto blkbegin
^D right	^QC end of file	^F word right	^QK goto blkend

^QE def metakey	^QJ goto last error	^QZ save linenumber	^W change win
^QT help metakey	^QN goto next error	^I tabulator	^Z dump/text
^UI set margin	^B format paragraph	^QI autotab on/off	^N new page
^QF find string	^UL delete file	^QL retake	^QB backup
^QA replace	^UM change directory	^QV view current directory	
^QQ repeat ^QF	^UT change filemode	^V insert/overwrite	ascii/hex

^UA new filename	^T del word	^UB,^UK set blockmark
^US save	^O del start of line	^UH del blockmark
^UQ quit,not save	^QY del end of line	^UC copy block
^UZ compress		^UF copy from windc
^UD save,quit	^Y del line	^UY del block
^UX compr,save	<DEL> del left	^UV move block
^UU compr,save,quit	^G del right	^UR read block
^UP forget old,load new	^P ins control	^UW write block
^UN save,load	^QM bytemask	^UJ mark word
^UE return to system	^QX displaymode	^UG mark line

3. Fenster

Durch ^W kann zur Bearbeitung einer zweiten Datei übergegangen werden. Beide Fenster teilen sich den Bildschirm und arbeiten vollständig unabhängig voneinander. Anschließend kann mit ^W zwischen den Fenstern gewechselt werden.

^UD, ^UQ usw. schließen das Fenster, in dem sich der Kursor gerade befindet (Es ist auch möglich, das zuerst vorhandene vor dem zuletzt eröffneten Fenster zu schließen. Dann übernimmt letzteres die Rolle des ersten Fensters.)

Übrigens wird, wenn fse mit der Option -m aufgerufen wird, für jedes Fenster ein 64 KByte großer Puffer angelegt. Standardgemäß müssen sich beide Dateien einen 64 KByte umfassenden Puffer teilen: Die Dateilänge ist durch die Größe des Puffers begrenzt. Das heißt, daß fse nur Dateien mit einer maximalen Länge von 64 KByte bearbeitet.

4. Bearbeitung von Quelltextdateien

Bearbeitungen von Quelltextdateien stellen mit Sicherheit die häufigste Anwendung des fse dar. Deshalb wird hierfür eine Vielzahl, in ihrer Leistungsfähigkeit stark differenzierter Kommandos bereitgestellt. Die einfachsten unter ihnen dienen der Kursorpositionierung, dem Einfügen und dem Löschen von Text.

Die Positionierung des Kursors erfolgt direkt über die Kursortasten oder die dazu äquivalenten Controltasten (siehe Helpmenü). Hinzu kommen Kommandos zur Positionierung des Kursors auf das nächste Wort links (^A) oder das nächste Wort rechts (^F), den Beginn einer Zeile (^QS), das Ende einer Zeile (^QD), den Beginn eines markierten Blockes (^QB), das Ende eines Blockes (^QK), den Beginn einer Seite (^R), das Ende einer Seite (^C), den Beginn der Datei (^QR) sowie das Ende der Datei (^QC). Weiter existieren Kommandos zur Positionierung des Kursors auf eine Zeile mit einer dezimal anzugebenden Zeilennummer (^QP).

^QZ schreibt die aktuelle Zeilennummer (nach Fenstern getrennt) in einen Puffer.

^QU legt die Nummer der aktuellen Zeile im Puffer ab und bewegt anschließend den Kursor in die Zeile mit der Zeilennummer, die zuvor im Puffer stand. War der Puffer leer, so wird ^QP ausgeführt. Die Befehle ^QZ und ^QU sind besonders wertvoll, wenn bei der Bearbeitung des Textes oft zwischen verschiedenen Textstellen gewechselt werden muß (z.B. zwischen Deklarations- und Anweisungsteil in einem Quelltext). Desweiteren sind sie sinnvoll, wenn durch ^QF oder ^QA Zeichenketten gesucht werden, denn anschließend befindet sich der Kursor an einer anderen Textstelle und die alte Ausgangsposition ist oft nicht sofort wiederauffindbar. Hier hilft ^QZ vor und ^QU nach dem Suchkommando. (Wem die Kommandofolge zu lang erscheint, kann sich dafür Metakeys definieren.)

^QF ist ein weiteres nutzliches Positionierkommando. Es bietet die Möglichkeit bis zu 30 Zeichen lange Zeichenketten im Text zu suchen. Die Ausführung kann durch Optionen beeinflußt werden. Als solche sind b für rückwärts suchen (standardmäßig wird im Text vorwärts gesucht) und u für upcase möglich. (letzteres besagt, daß beim Suchen einer Zeichenfolge zwischen Großbuchstaben und Kleinbuchstaben nicht unterschieden wird.)

^QA (Suchen und Ersetzen) wertet zusätzlich noch folgende Optionen aus: Eine Zahl, die angibt wie oft der Befehl automatisch zu wiederholen ist. Wird ein Stern angegeben, so wird der Befehl so oft ausgeführt, bis das Dateiende erreicht wird. Die Option n bewirkt die Unterdrückung von Anfragen vor jedem Austauschvorgang. Ansonsten ist auf diese Fragen mit y (ja, Zeichenkette soll ausgetauscht werden), n (nein, Zeichenkette soll nicht getauscht werden), c (cancel, für Abbruch des Kommandos) oder einem Stern (für Ersetzen ohne Anfragen bei jedem weiteren Auftreten der Zeichenkette im Text) zu antworten.

^QQ wiederholt das letzte Suchkommando (bzw. Suchen und Ersetzen).

Wichtig ist in diesem Zusammenhang, daß beim Suchen eine Bytemaske (siehe ^QM) über die Zeichen gelegt wird, so daß gegebenenfalls auch die Zeichen gefunden werden, die einen internen Code größer als 127 haben (diese treten z.B bei Texten auf, die durch WordStar bearbeitet wurden; dort dient das höchstwertige Bit der Markierung von Textstellen).

Das Einfügen von Text geschieht in recht einfacher Weise. Das Einfügen einzelner Zeichen wird durch Betätigen der jeweiligen Taste bewirkt. Standardgemäß wird das Zeichen so in den Text eingefügt, daß der Text ab Kursorposition um eine Spalte nach rechts rückt und das neue Zeichen dort erscheint, wo zuvor der Kursor stand.

^V schaltet zwischen overwrite -Modus (das heißt, daß von diesem Kommando an der eingegebene Text den bereits in der Zeile vorhandenen überschreibt) und insert -Modus (der neue Text wird in den alten eingefügt) um.

^P einer beliebigen Taste (z.B. einer Controltaste) vorangestellt, verhindert deren Interpretation als Editorkommando. Stattdessen wird das ihr zugeordnete Byte unvermittelt in den laufenden Text eingefügt. Insbesondere werden mit ^P^I 'harte Tabulatoren' in den Text geschrieben. Sie eignen sich hervorragend zum Aufbau von Tabellen. Beim Einrücken nach <ET> oder ^I werden sie an den entsprechenden Stellen aus der vorhergehenden Zeile übernommen, so daß sie nur in der 1. Zeile einer Tabelle eingegeben werden müssen.

Werden nicht direkt angebbare Zeichen (z.B. solche mit einem Code größer 127) benötigt, so kann mit ^Z in den Hexadezimal-Modus gewechselt und dort das gewünschte Zeichen eingegeben werden. Mit ^Z gelangt man wieder in den Ausgangszustand zurück.

^I stellt einen Pseudotabulator dar (meist ist hierfür eine spezielle Taste auf der Tastatur vorhanden). Durch dieses Kommando werden soviele Leerzeichen vor dem Kursor eingefügt, daß er unter dem nächsten Wort der darüberliegenden Zeile zu stehen kommt (geeignet zum Aufbau von Tabellen). Befinden sich in der darüberliegenden Zeile 'harte Tabulatoren', so werden diese anstelle der Leerzeichen eingefügt. Es erfolgt standardmäßig ein Aufruf dieses Kommandos nach jedem <ET>, so daß fortlaufend geschriebener Text (z.B. Quelltext) automatisch eingerückt wird.

^QI unterdrückt oder aktiviert (im Wechsel) diesen Dienst. Es ist unbedingt zu beachten, daß durch das Kommando ^I im Normalfall keine echten ('harten') Tabulatoren eingefügt werden, sondern Leerzeichen.

^QL sei hier noch als ein weiteres Einfügekommando genannt. Es kann immer direkt nach einem Löschkommando ausgeführt werden und holt den gelöschten Text wieder zurück.

Zuletzt seien noch einige Blockkommandos erwähnt, die ebenfalls Text einfügen können.

^UB definiert den Beginn und ^UK das Ende eines Textabschnittes als Block. Um diesen Arbeitsgang für häufig auftretende Anwendungsfälle zu vereinfachen, sind noch die Kommandos

^UG zum Markieren der Zeile und

^UJ zum Markieren des Wortes, auf dem der Kursor gerade steht, vorhanden.

^UH löscht Blockmarken im aktuellen Fenster. Blockmarken können auch neu gesetzt werden, ohne die alten explizit zu löschen (sie werden dann entsprechend verschoben).

^UC kopiert einen markierten Block an die aktuelle Kursorposition.

^UV ebenfalls, jedoch wird anschließend der markierte Block gelöscht.

^UR fügt ganze Dateien in den Text ein (ab aktueller Kursorposition). Dies ist besonders sinnvoll für Textverarbeitung in Verbindung mit Metakey, z.B. für Briefköpfe, Vordrucke, Formulare.

^UF transportiert einen Block aus dem jeweils anderen Fenster in das aktuelle. Dieses ist die einzige Möglichkeit des direkten Datenaustausches zwischen den Fenstern, wenn einmal von der Möglichkeit auf externe Dateien zu schreiben und wieder von ihnen zu lesen, abgesehen wird.

Die dritte Gruppe der simplen Editierkommandos bilden die Löschbefehle. Durch sie werden einzelne Zeichen, Zeichenfolgen oder größere Textabschnitte aus der Datei entfernt (wird direkt anschließend ^QL ausgeführt, so kann der gelöschte Text wieder zurückerhalten werden).

Durch <DEL> wird das Zeichen vor dem Kursor gelöscht; es wirkt auch über Zeilenenden hinweg (die verbleibende Zeile wird an die vorangehende angefügt).

^G löscht das Zeichen am Kursor, wirkt aber nicht über das Zeilenenden hinaus.

^T löscht das Wort, auf dem der Kursor gerade steht. Befindet sich der Kursor auf einem Sonderzeichen, so wird dieses gelöscht. Befindet er sich auf einem Leerzeichen, so werden alle Leerzeichen bis an das nächste Wort heran gelöscht.

^O löscht den Text vom Zeilenanfang bis ausschließlich Kursorposition.

^QY löscht den Text von (einschließlich) aktueller Kursorposition bis Zeilenende.

^UY löscht einen markierten Block.

^UW schreibt einen markierten Textabschnitt in ein externes File. Durch Schreiben eines Blockes auf das File /dev/lp kann ohne großen Aufwand ein Teil einer Datei ausgedruckt werden.

5. Textverarbeitung

Auch für elementare Textverarbeitung läßt sich der fse sinnvoll einsetzen.

^UI gestattet das Festlegen eines rechten Randes. Die mögliche Spaltenanzahl ist nicht begrenzt, muß aber größer als 2 sein. Ein Wert kleiner als 3 schaltet den Rand aus. Bei gültiger Randeinstellung erfolgt ein automatischer Wortumbruch beim Erreichen des Zeilenendes. Dabei werden Leerzeichen so in den Text eingefügt, daß ein gerader rechter Rand entsteht. Einrückungen bleiben erhalten.

^B wird verwendet, um größere Textabschnitte neu nach dem rechten Rand auszurichten. Durch dieses Kommando wird der gesamte Absatz, in dem sich der Kursor befindet, neu formatiert. Zur eindeutigen Kennzeichnung eines neuen Absatzes sollte entweder das erste Wort eingerückt werden oder die Zeile mit einem Control-

zeichen beginnen, welches von dem Drucker (oder dem Druckprogramm) ignoriert wird. Oft ist es nötig, bestimmte Controlzeichen in den Text einzufügen, die dann auf dem Bildschirm für Verwirrung sorgen können.

^QX gestattet das Festlegen eines Textdarstellungmodus, in dem Controlzeichen nicht angezeigt werden (mit demselben Kommando kann diese Einstellung auch wieder rückgängig gemacht werden).

Sind bestimmte Textabschnitte häufiger zu schreiben, so hat sich die Benutzung von Metakeys bewährt. Entweder wird der Text direkt eingefügt, oder aber für längere Abschnitte durch einen Metakey das Einfügen einer Datei veranlaßt, die den gewünschten Text beinhaltet.

^N fügt ein ^L (Seitenvorschub) in den Text ein, so daß beim späteren Ausdrucken eine neue Seite begonnen wird. Es ist zu beachten, daß ein Seitenvorschubsteuerzeichen bei der Darstellung auf dem Bildschirm den Übergang zu einer neuen Zeile bewirkt, aber von der Zeilenzählung des fse ignoriert wird.

6. Arbeit im Hexadezimaldump -Modus

Wie schon erwähnt, stellt der Editor fse keine speziellen Anforderungen an die Struktur der zu bearbeitenden Datei. Handelt es sich um Nicht-ASCII-Dateien, so wird die Darstellung des Dateiinhaltes als Text wenig Information bieten.

^Z gestattet es darum, den fse in einen Modus umzuschalten, in dem der Dateiinhalt als Hexadezimaldump dargestellt wird.

Beispiel: (Ausschnitt aus einem Verzeichnis)

```
+-----------------------------------------------------------------------------
ö File: /bin                                                        Line:    1
+-----------------------------------------------------------------------------
ö0000 ö 00 03 2E 00 00 00 00 00 00 00 00 00 00 00 00 00 ö >.................
ö0010 ö 00 02 2E 2E 00 00 00 00 00 00 00 00 00 00 00 00 ö >.................
ö0020 ö 00 40 61 64 62 00 00 00 00 00 00 00 00 00 00 00 ö >.@adb.............
ö0030 ö 00 00 67 65 6C 6F 65 73 63 68 74 00 00 00 00 00 ö >..geloescht1....
ö0040 ö 00 42 61 73 00 00 00 00 00 00 00 00 00 00 00 00 ö >.Bas.............
ö0050 ö 00 43 62 63 00 00 00 00 00 00 00 00 00 00 00 00 ö >.Cbc.............
ö0060 ö 00 00 62 63 6D 70 00 00 00 00 00 00 00 00 00 00 ö >..bcmp...........
ö0070 ö 00 45 62 64 00 00 00 00 00 00 00 00 00 00 00 00 ö >.Ebd.............
ö0080 ö 00 46 63 61 73 00 00 00 00 00 00 00 00 00 00 00 ö >.Fcas............
ö0090 ö 00 47 63 61 74 00 00 00 00 00 00 00 00 00 00 00 ö >.Gcat............
+-----------------------------------------------------------------------------
ö ^QH help ^Z text ^V ascii/hex ^QP position ^W window ^UD save,quit
+-----------------------------------------------------------------------------
```

Durch ^V kann der Kursor zwischen der Hexadezimal- und der ASCII-Darstellung bewegt werden. Ansonsten funktioniert alles wie im Textmodus. Speziell sind alle Kommandos zur Kursorpositionierung wirksam; mit folgenden Ausnahmen:

^QP positioniert nicht auf eine Zeile, sondern auf eine Adresse innerhalb der Datei (hexadezimal anzugeben).

^QF sucht nicht Zeichenketten, sondern Bytefolgen (ebenfalls hexadezimal anzugeben).

Desweiteren kann jederzeit Text überschrieben oder an die Datei angefügt werden. Soll eine Zeichenfolge innerhalb der Datei eingefügt werden, so ist ein Wechsel in den Textmodus gestattet. Dort können diese Aufgaben erledigt und anschließend wieder durch ^Z in den Ausgangszustand zurückgekehrt werden. Es ist jederzeit möglich, den Darstellungsmodus der Datei zu ändern.

Im obigen Beispiel ist ein Verzeichnis dargestellt. Jede Zeile stellt einen Eintrag dar. Die ersten beiden Bytes einer Zeile bilden die Inode-Nummer der Datei. Ist sie 0, so ist der Eintrag ungültig. Die folgenden 14 Bytes stellen den Dateinamen dar. Innerhalb des Editors kann die Datei beliebig bearbeitet werden, jedoch gestattet der fse kein Rückschreiben von Verzeichnissen. (Eine Ausnahme gilt für den Superuser. Es ist dabei zu beachten, daß beim Rückschreiben keine Backupdateien angelegt werden dürfen. Allgemein sollte diese Möglichkeit nur von erfahrenen Systemprogrammierern genutzt werden, da durch unsachgemäße Eingriffe große Schäden am Dateisystem entstehen können.)

7. Abspeichern der Datei und Beenden des Editierens

Der fse legt beim Laden eine Kopie der Datei im Hauptspeicher an und nimmt alle Veränderungen vorerst nur an dieser Kopie vor. Das

heißt, daß sich während der Arbeit mit dem fse das Aussehen der Datei auf dem externen Datenträger nicht verändert.

Durch ^US kann ein Abspeichern der Kopie bewirkt werden. Gewöhnlich wird dabei eine Backupdatei angelegt (alte Datei erhält am Namensende eine Tilde). Letzteres kann jedoch durch die Option -b beim Editoraufruf oder durch das Kommando ^OO zur Laufzeit abgestellt werden.

^UQ verwirft die im Hauptspeicher befindliche Kopie und beläßt die Datei in ihrem ursprünglichen Zustand. Wurden Modifikationen am Text vorgenommen, so wird zur Vergewisserung angefragt: workfile has been modified, save before quit ? Als Antwort sind hierauf n für nein (d.h. nicht Abspeichern, Editierfenster schließen), y für ja (d.h. Abspeichern und anschließend Fenster schließen) und c für Abbruch des Kommandos (d.h. es geschieht nichts und fse arbeitet normal weiter) möglich. Sind alle Fenster geschlossen worden, so beendet fse seine Arbeit.

^UD speichert die im Hauptspeicher befindliche Datei ab und schließt das Fenster. Wurde durch ^W ein zweites Fenster eröffnet, so bleibt dieses erhalten. Treten während des Abspeicherns Fehler auf (z.B. Gerätefehler oder fehlende Zugriffsrechte) so wird der Editor nicht verlassen, sondern es erscheint eine Fehlerausschrift; anschließend wird die Arbeit fortgesetzt.

^UA bietet die Möglichkeit, der im Hauptspeicher befindlichen Dateikopie einen anderen Namen zu geben, unter dem sie beim nächsten Savekommando abgespeichert wird. Dabei wird nicht überprüft, ob der Name gültig ist oder ob Zugriffsrechte auf die im Pfadnamen enthaltenen Verzeichnisse bestehen.

Nach Beenden der Eingabe einer Zeile durch <ET> werden gewöhnlich am unmittelbaren Zeilenende stehende Leerzeichen gelöscht, da sie an dieser Stelle überflüssig sind. Werden sie jedoch nachträglich in eine Zeile eingefügt, so bleiben sie erhalten und machen die Dateien länger als nötig.

^UZ entfernt überflüssige Leerzeichen am Zeilenende aus der Datei. Dieses Kommando tritt noch in folgenden Modifikationen auf:

^UX : Leerzeichen am Zeilenende streichen und Datei abspeichern;

^UU : wie ^UX, jedoch danach das Fenster schließen.

Bisweilen geschieht es, daß ein anderer Dateiname als beabsichtigt in der Kommandozeile oder beim Neuladen einer Datei angegeben wird. In diesem Fall ist es nicht nötig den fse zu beenden und erneut aufzurufen.

^UP ermöglicht jederzeit das Verwerfen der Kopie und das Laden einer neuen Datei. Dabei werden alle Parameter für das jeweilige Fenster zurückgesetzt (z.B. Randeinstellung etc.). Wurden schon Veränderungen am Text vorgenommen, so vergewissert sich der Editor mit der Frage 'save before quit ?' analog zu ^UQ über die Ernsthaftigkeit dieses Kommandos. Soll die bearbeitete Datei abgespeichert und anschließend eine neue Datei in das Fenster geladen werden, so kann dafür das Kommando ^UN verwendet werden. Auch hier erfolgt ein Rücksetzen aller für das Fenster gültigen Parameter auf ihren Standardwert.

^UE ist ein weiteres Kommando zum Beenden des Editors, ohne die Datei abzuspeichern. Der fse kehrt dabei mit einem Rückkehrcode von 1 zum Betriebssystem zurück. Ein make, das den fse aufgerufen hat wird dadurch abgebrochen. So bietet dieses Kommando auch die Möglichkeit, fsc abzubrechen und damit aus dem Zyklus von

Übersetzen und Editieren auszutreten (s. Abschn. 10).

8. Metakey

Ein <ESC> gefolgt von einem Kleinbuchstaben ist ein Metakey. Für jeden Kleinbuchstaben des Alphabets befindet sich in einem fse-internen Puffer eine Folge von Zeichen, die nach Betätigung des entsprechenden Metakey in den Eingabestrom eingefügt wird (auch Editorkommandos sind zulässig). Ein Überblick über die Belegung ist durch das Kommando ^QT abrufbar.

^QE gestattet es dem Nutzer, eigene Sequenzen als Metakey zu definieren. Diese sind aber lediglich im gerade aktiven fse gültig. Sollen die Definitionen für spätere fse-Aufrufe verfügbar bleiben, so ist der Editor mit der Option -k aufzurufen. Die aktuellen Definitionen werden dann beim Verlassen des fse in dem File .fse abgelegt (dieses File darf jederzeit gelöscht werden).

Bei der Definition sind folgende Besonderheiten zu beachten:

- Zuerst wird ein Kleinbuchstabe zur Kennzeichnung des Metakey eingegeben. Ihm folgt die gewünschte Tastenfolge.

- <DEL>-Taste und Kursortasten liefern nur ihren internen Code, bewirken aber keine Korrektur der Eingabe. Bei Fehlern muß darum die gesamte Definition wiederholt werden. Aus Portabilitätsgründen sollten Aufrufe von Editorfunktionen als Controlsequenz und nicht als Funktionstaste eingegeben werden, denn der Code von Funktionstasten kann von Rechner zu Rechner verschieden sein.

- Beendet wird die Definition mit ^D. Soll ein ^D oder ein ^P in der Sequenz enthalten sein, so ist ihnen ein ^P voranzustellen.

9. Weitere Editorkommandos und Optionen

Durch eine Vielzahl von Optionen können Parameter des fse gezielt voreingestellt werden. Sie werden beim Aufruf des fse als erstes Argument in der Kommandozeile, (als mit einem Minuszeichen beginnende Zeichenfolge) angegeben. Bei Mehrfachkodierungen der gleichen Option wird ihr Wert bei jedem Auftreten negiert. Die Reihenfolge der Optionen ist ohne Bedeutung.

Beispiel: fse -bm100 testfile

Folgende Buchstaben sind zulässig und werden wie folgt ausgewertet:

a: setzt eine Randmarkierung auf die Spaltenposition 66 (siehe Textverarbeitung)
b: unterdrückt das Anlegen von Backupdateien
c: unterdrückt die Ausgabe der Eröffnungsausschrift
e: sucht beim Programmstart im aktuellen Verzeichnis nach einem File 'errmsg' und benutzt die Einträge für die Zuordnung von Fehlerausschriften zu den betreffenden Textstellen (siehe ^QJ und ^QN)
k: bewirkt das Laden und nach Beendigung des fse das Sichern der Metakeytabelle (benutzt File .fse im aktuellen Verzeichnis)
m: legt einen größeren Puffer an (für jedes Fenster 64 Kbyte)
n: fügt bei <ET> CR LF in den Text ein (MS-DOS Zeilenende). Auf Unixsystemen wird gewöhnlich nur LF zur Zeilenendekenn-

zeichnung verwendet.

r: unterdrückt explizit das Laden der zuletzt bearbeiteten Datei (im Falle daß kein Dateiname in der Kommandozeile angegeben war).

y: tauscht logisch die <Y> und die <Z> -Taste auf der Tastatur aus.

Die Angabe einer Zahl in der Optionenzeichenkette bewirkt die sofortige Positionierung des Kursors in die Zeile mit der entsprechenden Zeilennummer.

Um ständig benutzte Optionenfolgen nicht immer in der Kommandozeile angeben zu müssen, besteht die Möglichkeit, eine Umgebungsvariable FSE zu definieren (mit setenv in der C-Shell), die als Wert eine Zeichenkette hat. Soweit vorhanden, wird sie bei jedem Start von fse vor die in der Kommandozeile angegebene Optionenzeichenfolge gesetzt. Es ist möglich, durch Angabe der entsprechenden Option in der Kommandozeile eine durch die Umgebungsvariable FSE voreingestellte Option zu neutralisieren.

Einige Parameter des fse können auch zur Laufzeit noch verändert werden. Hier seien die wichtigsten aufgezählt:

^QX legt die Darstellungsweise von Zeichen fest, die laut ASCII nicht darstellbar sind (z.B. Steuerzeichen). Dabei stehen vier Möglichkeiten zur Auswahl: (1) keine Darstellung, (2) Darstellung als Leerzeichen, (3) Darstellung als Punkt, (4) Darstellung als ^Buchstabe für Controlzeichen und als Punkt sonst.

^QM wählt eine Maske aus, mit der die Bytes einer Datei vor der Ausgabe auf dem Bildschirm oder beim Suchen einer Zeichenkette überdeckt werden. Als Masken werden (1) 01111111 und (2) 11111111 angeboten. Standardmäßig ist die erste Maske aktiv. Sie bewirkt (im Gegensatz zur Maske 2, die keine Wirkung auf die Bytes der Datei hat) die Ausblendung des höchstwertigen Bits eines Zeichens. Dieses Bit wird von einigen Textverarbeitungsprogrammen zur Kennzeichnung markanter Textstellen benutzt. Damit ist aber genaugenommen das Zeichen laut ASCII nicht mehr darstellbar und würde auch beim Suchen von Zeichenketten (^QF bzw. ^QA) sich vom ursprünglichen Zeichen unterscheiden, und die Zeichenkette würde nicht gefunden werden. Die Maske (1) schafft hier Abhilfe. Es sei noch einmal erwähnt, daß durch die Maske die Bytes im Text nicht verändert werden. Vielmehr wird diese Maske nur bei der Interpretation des Dateiinhaltes durch den fse angewendet.

^QO legt fest, ob beim Überschreiben bereits vorhandener Dateien durch den fse Backupdateien angelegt werden sollen (das sind Dateien mit einem Dateinamen bestehend aus dem ursprünglichen Namen, gefolgt von einer Tilde; so wird z.B. aus dem File test.c eine Backupdatei test.cß, die den alten Text beinhaltet und test.c ist die Datei mit dem neuen Text). Es wird dabei der gerade eingestellte Modus angezeigt.

^UL bewirkt das Löschen einer Datei.

^UM wechelt das aktuelle Arbeitsverzeichnis des fse. Dieser Befehl hat keinen Einfluß auf das Arbeitsverzeichnis der Shell, von der aus fse aufgerufen wurde. Dort ist nach Beendigung des fse die ursprüngliche Einstellung gültig.

^UT bietet die Möglichkeit, Zugriffsrechte für Dateien zu ändern. Dieses Kommando ist dann sinnvoll, wenn z.B. durch das Nichtvorhandensein von Schreibrechten das Abspeichern der Datei durch das Betriebssystem unterbunden wird. Dieses Kommando wird jedoch nur dann abgearbeitet, wenn der Ausführende auch Eigentümer der Datei ist. Anderenfalls sollte mit ^UA dem File ein anderer

Name gegeben werden, so daß es in ein Verzeichnis geschrieben wird, für das die Schreiberlaubnis vorhanden ist. (Die neuen Zugriffsrechte werden oktal angegeben (siehe UNIX-Kommando chmod). Durch zweimaliges Betätigen der <ESC>-Taste kann das Kommando abgebrochen werden)

^QH listet eine Zusammenstellung der wichtigsten Editorkommandos mit kurzer Bedeutungsangabe auf. Sind zwei Fenster aktiv, so wird der Text auf zwei kleineren Seiten dargestellt, die umgeblättert werden. Durch <DEL> kann das Kommando abgebrochen werden. Ansonsten ist durch das Drücken einer beliebigen Taste die Help-funktion fortzusetzen bzw. zu beenden.

^QT stellt eine Liste der aktuellen Metakey-Definitionen zusammen.

Beim Beenden des fse wird das im aktuellen Arbeitsverzeichnis befindliche File .fse aktualisiert oder angelegt. Dieses File darf jederzeit gelöscht werden. Es beinhaltet Informationen über den Namen und die Position des Kursors der zuletzt bearbeiteten Datei. Wird fse ohne Spezifikation eines Filenamens in der Kommandozeile aufgerufen, so wird versucht, die zuletzt bearbeitete Datei wieder zu laden und den Kursor in die Zeile zu setzen, in der er vor dem letzten Verlassen des fse stand. Die Zeilennummer kann durch explizite Angabe in der Kommandozeile überschrieben werden. Wurde fse mit der Option -k aufgerufen, so wird zusätzlich aus .fse die Tabelle der Metakeydefinitionen geladen und nach Beendigung des fse wieder in diesem File abgelegt.

10. Die Zusatzkomponente fsc

Jedem Programmierer ist wohl der oft erhebliche Zeitaufwand für die Korrektur von Syntaxfehlern in Quelltexten bekannt. Und wer hat noch nicht mit den zeitraubenden und ständig wiederkehrenden, oft zeilenfüllenden Kommandofolgen für Aufruf von Compiler und Editor Bekanntschaft gemacht; vom umständlichen Zuordnen der Fehlerausschriften zu den betreffenden Textstellen ganz zu schweigen. Dies sollte endlich der Vergangenheit angehören.
fse stellt deshalb einen besonderen Dienst zur Fehlerkorrektur in Quelltexten bereit, der von beliebigen Programmen genutzt werden kann.
Derzeit existiert Software zur Kopplung von C- , Sprachübersetzer, sowie eine Kopplung mit einem an der Humboldt- Universität Berlin entwickelten Modula-2-System. Die Kopplung mit dem C-Compiler soll hier als Beispiel vorgestellt werden.
fsc ist ein selbständiges Programm, das die oben erwähnte Schnittstelle des fse nutzt und die Zusammenarbeit und den Datenaustausch zwischen Compiler und Editor organisiert.
Nach dem Start wird eine erste Übersetzung des Quelltextes vorgenommen (die Kommandozeile des fsc wird dabei vollständig dem Compiler übergeben). Verläuft die Übersetzung fehlerfrei, dann endet fsc. Treten Syntaxfehler auf, so wird sofort der Editor fse gestartet. Dort enthält die unterste Bildschirmzeile eine kurze Fehlermeldung und der Kursor befindet sich in der betreffenden Zeile. Zusätzlich zum normalen Befehlssatz des fse kann nun mit ^QJ zur vorhergegangenen und mit ^QN zur nächsten fehlerhaften Zeile gegangen werden.

Die zugehörige Fehlermeldung befindet sich immer dann in der untersten Bildschirmzeile, wenn der Kursor in einer der fehlerhaften Zeilen positioniert ist. Eventuelle Verschiebungen der Textstellen durch Einfügen oder Löschen von Text werden durch den fse berücksichtigt. Nach Beendigung des Editiervorganges wird erneut der Compiler gestartet. Dieser Arbeitsablauf wird

wiederholt, bis die Übersetzung fehlerfrei verläuft. Ist dies nicht erwünscht, so ist fse mit dem Kommando ^UE zu verlassen oder fsc mit <DEL> abzubrechen.

Können Fehler nicht behoben werden, so endet fsc mit dem Rückkehrcode 1, so daß ein make ordnungsgemäß abbricht. In Verbindung mit dem Hilfsprogramm make läßt sich der gesamte Arbeitsablauf weitestgehend automatisieren und kann hier sehr empfohlen werden.

Beispiel für ein makefile:

```
CC=fsc
beispiel:    beispiel1.o beispiel2.o beispiel3.o
beispiel1.o: beispiel1.c beispiel.h
```

Übrigens speichert fse nach dem Kommando ^UD die Datei nur dann ab, wenn der Text wirklich modifiziert wurde, so daß durch make für eine Datei, die zwar geladen, aber nicht verändert wurde, keine erneute Übersetzung gestartet wird.