# TUSTEP: Editormakros jetzt auch auf zentralen Anlagen des ZDV

Editormakros erleichtern die Arbeit im Editor, da sie mehrere Tastenanschläge unter einer einfachen Tastenkombination zusammenfassen. Eine solche Tastenkombination (Editormakro) steht jeweils stellvertretend für einen Text und/oder eine Folge von Sondertasten für Steuerbefehle (markieren, springen, Zeilen löschen, Gelöschtes einfügen, blättern etc.), die bei der Definition des Editormakros beliebig miteineinander kombiniert werden können.

Bisher war die Verwendung von Editormakros im wesentlichen nur auf dem PC unter MS-DOS möglich, da die COMPAREX unter MVS ihre Verwendung nicht zuließ. Nun stehen unter UNIX (*convex*, *textserv*) die Editormakros auch auf den zentralen Anlagen des ZDV zur Verfügung. Aus diesem Anlaß sollen hier typische Anwendungen von Editormakros vorgestellt werden.

Eine vollständige (wenn auch knappe) Beschreibung von Definition und Aufruf der Editormakros sowie die vollständige Liste und Beschreibung der Steuerbefehle findet sich im Nachtrag zu den Seiten 157–165 des TUSTEP–Handbuchs 1989, der in neuester Version vom Juli 1992 bei Frau Krier im ZDV erhältlich ist.

### Definition und Aufruf von Editormakros

Die y-Anweisung zur Definition eines Editormakros kann als Anweisung interaktiv im Editor gegeben werden. Ein auf diese Weise definiertes Editormakro behält seine Gültigkeit bis zum Beenden von TUSTEP. Sinnvoll ist diese Art der Definition zum Testen und für Ad-hoc-Anwendungen, die nur für die aktuelle Arbeit gebraucht werden.

Komplexere Editormakros und solche, die man öfter benutzen will, definiert man mit dem Kommando

```
#ediere, definitionen=*
```

(abgeschlossen mit **eof*).

Schreibt man dieses Kommando in die Datei TUSTEP.INI, dann stehen die Editormakros immer zur Verfügung.

Die Editormakros, die nur aus einem Buchstaben bestehen, sind wesentlich leichter und schneller aufzurufen als die mit längeren Namen. Daher gilt die Empfehlung:

Für Editormakros, die oft gebraucht werden und die einfache Leistungen bringen, wählt man als Namen *einen* Buchstaben. Für komplexe Editormakros nimmt man – wenn keine geeigneten Einzelbuchstaben mehr frei sind – längere Namen.

Die Ausführung der Editormakros beginnt in jedem Fall an der Stelle, an der der Cursor sich beim Aufruf des Editormakros befindet.

## Anwendungen

### Textstücke

Benötigt man beim Schreiben häufig dasselbe lange Wort, so lohnt es sich, dafür – spontan für diesen speziellen Zweck – ein Editormakro zu definieren.

Beispiel

```
y,d='Donaudampfschiffahrtsgesellschaft'
```

Die Zeichenfolge beginnt mit einem freiwählbaren Begrenzungszeichen und endet mit dem gleichen Begrenzungszeichen; das Begrenzungszeichen darf kein Buchstabe sein.

Nachdem die Definition erfolgt ist, wird durch die Tastenkombination *<ALT>+<D>* bzw. *<PF1> <D>* das berühmte Wortungetüm an der Cursorposition auf den Bildschirm geschrieben, wie wenn man das Wort an der Stelle eintippen würde. (Die Einstellung *INSERT* und *REPLACE* ist zu beachten, Wordwrap am Zeilenende wird durchgeführt.)

Sinnvoll ist diese Verwendung von Editormakros auch, um Steuerzeichen in den Text zu schreiben.

### Einzelne Steuerbefehle

Die Definition eines Editormakros mit einem einzelnen Steuerbefehl ist sinnvoll, wenn die für den entsprechenden Steuerbefehl vorgesehenen Sondertasten zu kompliziert sind (z. B. beim Markieren) oder wenn für den entsprechenden Steuerbefehl keine Sondertasten vorgesehen sind (z. B. das Schreiben des Tagesdatums).

Beispiel: Markieren
Für das Arbeiten mit markiertem Text sind Editormakros zumindest für die vier häufigsten Anwendungen zu empfehlen.
Beginnen des Markierens:

```
y,a=mrk_ini
```

Speichern des markierten Textes in den Puffer:

```
y,e=mrk_rep
```

Markierten Text löschen und in den Puffer speichern:

```
y,l=mrk_del_rep
```

Einfügen des Pufferinhalts auf den Bildschirm an der Cursorposition:

```
y,i=mrk_ins
```

Diese Definitionen gehören in die Datei TUSTEP.INI, damit diese Editormakros immer zur Verfügung stehen.

Beispiel: Schreiben des Tagesdatums
Folgendes Editormakro – aufzurufen mit *<ALT>+<D>* bzw. *<PF1><D>* – schreibt das aktuelle Tagesdatum in der langen Form (mit ausgeschriebenem Monatsnamen) an der Cursorposition auf den Bildschirm:

```
y,d=date_3
```

Kombination von mehreren Steuerbefehlen

In einem Editormakro können mehrere Steuerbefehle angegeben werden. Sie werden durch Komma voneinander getrennt. Beim Aufruf des Editormakros werden sie in der angegebenen Reihenfolge abgearbeitet.

Beispiel: Positionieren des Cursors an das Ende der obersten Bildschirmzeile:

```
y,t=home, skp_end
```

*HOME* springt an den Anfang des Textes in der ersten Bildschirmzeile, *SKP_END* springt mit dem Cursor ans Ende der Zeile.

Eine Schwierigkeit beim Definieren solcher Sequenzen ist, daß die Ausgangsposition des Cursors beim Aufruf des Editormakros nicht festliegt, was bei der Abarbeitung des Editormakros zu unerwünschten Effekten führen kann. Im vorliegenden Fall bringt das Editormakro nicht den gewünschten Erfolg, wenn der Cursor beim Aufruf bereits am Textanfang in der ersten Bildschirmzeile steht. Denn von dieser Position aus springt *HOME* in die Anweisungszeile. Man muß also an den Beginn des Editormakros noch den Steuerbefehl *CMD_LINE* einfügen. Damit wird der Cursor absolut an den Anfang der Anweisungszeile positioniert. Von hier springt *HOME* sicher nach oben. Die verbesserte Form lautet:

```
y,t=cmd_line, home, skp_end
```

Beispiel: Löschen bis zum vorhergehenden Wortanfang
Diese Funktion (Gegenstück zu *DEL_WORD*) kann mit einem Editormakro durch eine Kombination von Steuerbefehlen erreicht werden:

Man beginnt zu markieren (*MRK_INI*), springt nach links zum nächsten Wortanfang (*SKP_LE*) und löscht den markierten Text (*MRK_DEL_REP*). Das Wort steht nun im Puffer und kann später ggf. wieder in den Text eingefügt werden. In der Praxis hat sich bewährt, das Zeichen, auf dem der Cursor aktuell steht, vom Löschen auszunehmen. Da jedoch beim Markieren das erste Zeichen zum markierten Text gehört, muß im Editormakro zunächst der Cursor nach links bewegt und erst dann das Markieren begonnen werden. Anschließend bewegt man den Cursor wieder um ein Zeichen nach rechts, damit das Springen von der ursprünglichen Stelle erfolgt.

```
y,w=cur_le, mrk_ini, cur_ri, skp_le,
  mrk_del_rep
```

Mit *<ALT>+<W>* bzw. *<PF1><W>* werden nun die Zeichen bis zu dem vorhergehenden Wortanfang exklusiv der Cursorposition gelöscht. Der Cursor bleibt auf dem Zeichen stehen, auf dem er vor Aufruf des Makros stand.

Tips zum Definieren eines Editormakros

In einem Editormakro muß Schritt für Schritt gesagt werden, wie sich der Cursor bewegen soll und welche Steuerbefehle ausgeführt werden sollen. Am besten führt man die einzelnen Schritte, die in einem Editormakro zusammengestellt werden sollen, von Hand – d. h. unter Verwendung der Sondertasten und der Schreibmaschinentastatur – aus und notiert jeden Tastenanschlag – und zwar wirklich jeden, d. h. auch jede einzelne Bewegung des Cursor nach oben, unten, links, rechts. Anschließend sucht man die Steuerbefehl-Namen für die einzelnen Tastenanschläge im Handbuch und schreibt sie in der gegebenen Reihenfolge in die Editormakro-Definition.

Beim Austesten des Editormakros hilft dann der Steuerbefehl *WAIT*, den man an geeigneten Stellen in die Definition schreibt.

Kombinieren von Textteilen und Steuerbefehlen

Im Editormakro können zwischen den Steuerbefehlen an beliebiger Stelle Textteile angegeben werden. Sie werden bei der Abarbeitung des Editormakros an der Stelle auf den Bildschirm geschrieben, wo sich der Cursor zu diesem Zeitpunkt bei der Abarbeitung befindet.

Beispiel
Man will in einer Datei Zeilen teilen und jeweils am Beginn der neu entstandenen Zeile eine Absatzmarkierungen ($) einfügen.

```
y,a=split, lf, '$'
```

Das Dollarzeichen wird an den Anfang der neu entstandenen Zeile auf den Bildschirm geschrieben, denn dort befindet sich der Cursor zu diesem Zeitpunkt bei der Abarbeitung des Editormakros.

## Integrieren von Anweisungen in Editormakros

Anweisungen im Editor sind zunächst einmal gewöhnlicher Text, der in der Anweisungszeile eingegeben, abgeschickt und – falls die Anweisung richtig war – ausgeführt wird. Die Anweisungszeile kann man innerhalb von Editormakros erreichen, das Schreiben eines gewöhnlichen Textes ist mit Editormakros möglich, und das Abschicken ist ein Steuerbefehl. Damit können Anweisungen in Editormakros integriert werden.

Beispiel: Zeigen der vorhergehenden Seite
In einer Datei, die im Textmodus numeriert ist, soll der Anfang der Seite vor der aktuellen Seite gezeigt werden.

```
y,z=enter, 'za,', page_nr_dec, '.0',
  enter
```

Der Bildschirm wird abgeschickt, und der Cursor springt an den Anfang der Anweisungszeile. Dort wird der Beginn der gewünschten Anweisung (*za,*) geschrieben. *PAGE_NR* ist ein Steuerbefehl, der die aktuelle Seitennummer auf den Bildschirm schreibt; *PAGE_NR_DEC* (decrement) vermindert die aktuelle Nummer um eins und schreibt diese Zahl auf den Bildschirm. Für eine Zeigeanweisung ist es notwendig, daß hinter der Seitennummer eine Zeilennummer folgt, also wird noch *.0* (Zeilennummer 0) geschrieben. Die Anweisung ist jetzt komplett und wird mit dem Steuerbefehl *ENTER* abgeschickt.

Beispiel: Eingabemaske für strukturierte Daten
Das Erfassen strukturierter Daten wird erleichtert durch ein Editormakro, das eine Maske auf den Bildschirm schreibt.

Erfaßt werden soll eine Bibliographie mit Autor (*&a*), Titel (*&t*) und Jahreszahl (*&j*). Davor gibt es zwei Felder für eine laufende Nummer (*&n*) und das Erfassungsdatum (*&d*). Die laufende Nummer soll beim Eintragen der Seitennummer der Sätze (Numerierung im Textmodus) entsprechen. Bei jedem Datensatz wird eine neue Seitennummer begonnen (geschieht automatisch, wenn der Datensatz mit Anweisung *ee* jeweils hinten an die Datei angehängt wird). Das aktuelle Tagesdatum dient zur Kontrolle, wann der Datensatz erfaßt wurde.

```
y,m=2*enter, 'ee', enter, '&n', page_nr,
  lf, '&d', date_1, lf, '&a', lf, '&t',
  lf, '&j', 2*cur_up, cur_ri
```

Das Editormakro beginnt mit zweimaligem *ENTER* (Steuerbefehle, die hintereinander wiederholt gebraucht werden, können in dieser abgekürzten Form in die Definition geschrieben werden) zum Beenden der Dateneingabe. Es erscheint *Gib Anweisung >* und der Cursor befindet sich dahinter. Dies ist notwendig, um die soeben ausgefüllte Maske abzuschicken und für die nächste leere Maske, die auf den Bildschirm geschrieben wird, die richtige Numerierung zu bekommen. War man beim Aufruf des Editormakros nicht bei der Dateneingabe, so stört zweimaliges *ENTER* nicht.

Mit Anweisung *ee* beginnt das Eintragen unter einer neuen Seitennummer. Die Feldkennung *&n* wird auf den Bildschirm geschrieben, dahinter wird die aktuelle Seitennummer (*PAGE_NR*) als laufende Nummer geschrieben. Nach dem Sprung in die nächste Zeile (*LF*) wird *&d* auf den Bildschirm geschrieben, dahinter das Tagesdatum (*DATE_1*). In den nächsten Zeilen folgen die weiteren Kennungen. Nachdem die Maske fertig auf dem Bildschirm steht, wird der Cursor hinter die Autorenkennung gestellt (*2*CUR_UP, CUR_RI*), damit man an dieser Stelle mit dem Eingeben der Daten beginnen kann.

## Integrieren von Kommandoaufrufen

Die Anweisung *x*, mit der aus dem Editor Kommandos aufgerufen werden, ist eine Anweisung wie jede andere und kann natürlich auch in Editormakros integriert werden. Im einfachen Fall leistet ein solches Editormakro dasselbe wie eine entsprechend belegte Funktionstaste. Einen Vorteil bringt der Aufruf per Editormakro, wenn vor oder nach der Ausführung des Kommandos im Editor noch einige Steuerbefehle ausgeführt werden sollen. Nach der Ausführung des Kommandos wird das Editormakro fortgesetzt, falls es zuvor nicht vollständig abgearbeitet war.

Beispiel
Die aktuelle Editordatei enthält eine Kommandofolge, die ausgetestet werden soll. Dazu soll die Kommandofolge ausgeführt und anschließend das Ende der Zieldatei *ziel* betrachtet werden.

```
y,x=enter, 'x #tu,,<editor>', enter,
  'd,ziel', enter, 'ze', enter
```

## Erzeugen kontextsensitiver Anweisungen

Ein weiterer Schritt an Komfort sind Editormakros, die Zeichenfolgen in die Anweisung integrieren, auf die der Cursor zu Beginn der Abarbeitung des Editormakros positioniert ist. Das erste Beispiel ist als echte Empfehlung gedacht, da es die tägliche Arbeit erleichtert. Das zweite Beispiel soll die Phantasie für eigene Anwendungen anregen.

### Beispiel: Holen eines Segments über das Inhaltsverzeichnis

Bei der Arbeit mit Segmentdateien kommt es häufig vor, daß man sich zunächst mit der Anweisung *h,datei,?* das Inhaltsverzeichnis der Segmentdatei in die Editor-Datei holt, um zu sehen, welche Segmente es gibt. Meist wird man anschließend eines dieser Segmente in den Editor holen. Dazu muß man den Namen des Segments in der Anweisungszeile eintippen, obwohl der Segmentname bereits auf dem Bildschirm steht. Das folgende Editormakro soll diese unnötige Schreibarbeit ersparen: Man wählt das gewünschte Segment aus, indem man den Cursor in die betreffende Zeile des Inhaltsverzeichnisses positioniert, und startet das Editormakro, das eine Hole-Anweisung für das Segment dieser Zeile erzeugt.

```
y,h=cur_ri, skp_beg, skp_ri,
  mrk_ini, skp_ri, mrk_rep,
  cmd_line, 'h,,', mrk_ins, enter
```

Das Editormakro geht davon aus, daß der Cursor in der gewünschten Zeile überall stehen darf. Um den Cursor definiert an den Anfang zu bringen, muß er zunächst nach rechts bewegt werden (falls er schon auf der 1. Position steht). Mit *SKP_RI* überspringt man die Zeichenfolge #= und gelangt zum Anfang des Segmentnamens. Der Segmentname wird mit Hilfe der Markiere-Steuerbefehle im Puffer gemerkt (*MRK_INI, SKP_RI, MRK_REP*). Das Markieren verändert den Bildschirm nicht; Anweisung *h!* wird nicht notwendig. In der Anweisungszeile (*CMD_LINE*) wird *h,,* geschrieben und dahinter der Segmentname eingefügt (*MRK_INS*). Die gewünschte Hole-Anweisung ist fertig und kann abgeschickt werden (*ENTER*).

### Beispiel: Arbeiten mit Bildschirmfenstern

Beim Arbeiten mit geteiltem Bildschirm erleichtern Editormakros auf vielfältige Weise das Wechseln zwischen den Fenstern.

Man arbeitet z. B. im unteren Fenster, braucht aber zu einzelnen Wörtern oder Zeichenfolgen immer wieder Informationen, die sich in der oberen Datei befinden. Das Editormakro sucht das Wort (Zeichenfolge zwischen Blanks), auf das man im unteren Fenster den Cursor positioniert hat, in der Datei des oberen Fensters und kehrt wieder ins untere zurück. Man hat die gewünschte Information oben stehen und kann sie für die Arbeit unten benutzen.

```
y,m=cur_ri, skp_le,
  mrk_ini, skp_ri, mrk_rep,
  cmd_line, 'm,1', enter, 'za,,,/',
  mrk_ins, '/', enter, 'm,2', enter
```

Der Anfang des Editormakros (*CUR_RI, SKP_LE*) positioniert den Cursor von einer beliebigen Stelle des Worts (einschließlich nachfolgenden Blanks) auf den Wortanfang. Das Wort wird gemerkt, (der Bildschirm verändert sich durch das Markieren nicht), das Fenster wird gewechselt, die Sucheanweisung wird erzeugt und ausgeführt; man kehrt ins untere Fenster zurück.

## Definition von Editormakros im Editoraufruf

Die permanente Definition der Editormakros erfolgt normalerweise (in TUSTEP.INI) mit einem Editoraufruf, der ohne eine Dateiangabe nur zum Zwecke der Definitionen gemacht wird. Es ist auch möglich – und das ist bei Editormakros, die in Kommandofolgen integriert sind – besser, die Definition der Editormakros dem aktuellen Editoraufruf mitzugeben. Dieses Vorgehen bietet die Möglichkeit, in die Definition der Editormakros Makrovariablen oder Kennzeichen für TUE–Parameter zu integrieren, die erst zur Laufzeit belegt werden, und so das Editormakro entsprechend der aktuellen Umgebung im Kommando-Makro oder in der Kommandofolge zu definieren.

## Editormakro-Aufruf im Kommando EDIERE

Ein Editormakro kann auch zur Ausführung gebracht werden, indem man es beim Aufruf des Editors im Kommando EDIERE zur Spezifikation MAKRO angibt. Das angegebene Editormakro wird nach dem Start des Editors ausgeführt.

Man kann zur Spezifikation MAKRO entweder ein Editormakro angeben, das bereits vorher – an anderer Stelle – definiert wurde, oder ein Editormakro, das erst mit dem aktuellen Editoraufruf (Spezifikation DEFINITIONEN definiert wird.

Der Vorteil dieses Aufrufs eines Editormakros ist, daß man den Editoraufruf in Kommandofolgen integrieren kann und daß im Editor sofort einige Dinge erledigt werden.

Beispiel: Belegung der Bildschirmfenster mit bestimmten Dateien

Die Arbeit mit Bildschirmfenstern ist besonders praktisch, wenn die verschiedenen Fenster mit den richtigen Dateien belegt sind. Ein Editormakro hilft, schon beim Editoraufruf die Fenster für bestimmte Arbeitsabläufe zu belegen. Es wird eine Kommandofolge in einer Programmdatei namens *fenster* zusammengestellt, die eine Datei in das große Fenster des Editors ruft und die beiden kleinen Fenster mit anderen Dateien belegt. Die drei Dateien werden als Parameter im Kommando TUE angegeben.

Die Programmdatei *fenster* enthält die Kommandofolge:

```
#ed, ?1, def=*, makro=beleg
y,beleg='m,1', enter, 'd?2', enter,
  'm,2', enter, 'd?3', enter,
  'm,0', enter, 'za', enter
*eof
```

Das Komma bei der Dateianweisung wegzulassen hat den Vorteil, daß man beim Aufruf auch die Parameter unvollständig angeben kann. Es wird dann die zum vorhergehenden Parameter angegebene Datei ins entsprechende Fenster übernommen bzw. die im Fenster vorhandene Datei bleibt stehen.

Der Aufruf dieses Programms sieht z. B. folgendermaßen aus:

```
#tu,fenster,,prog'quelle'ziel
```

Die Datei *prog* kommt in das große Fenster und wird von Anfang an gezeigt. Die kleinen Fenster werden oben mit der Datei *quelle* und unten mit der Datei *ziel* belegt.

Beispiel: Hilfstext zu Beginn einer Editorsitzung

Manchmal werden Editoraufrufe in Kommandofolgen integriert, die auch von anderen – mit TUSTEP weniger Vertrauten – benutzt werden: Editoraufruf gleich nach dem Initialisieren, um den Benutzer bestimmte Arbeiten dort ausführen zu lassen; Aufruf von Dateien zur Recherche während eines Kommandomakros; Sichtkontrolle von Zwischenergebnissen während des Abarbeitens einer Kommandofolge (die Kommandofolge läuft nach Beenden des Editors weiter). Es ist in diesen Fällen sinnvoll, jeweils nach Aufruf des Editors einen Hinweis auf den Bildschirm zu schreiben, der erklärt, was zu tun ist, und u. U. noch weitere Hilfsfunktionen erläutert.

Ein solcher Editoraufruf sieht folgendermaßen aus:

```
#ed, datei, def=*, makro=info
y,info=clear, 1f, 1f, 10*cur_ri,
  'Blättern Sie mit <PgUp> und <PgDn>',
  1f, 1f, 10*cur_ri,
  'Weitere Hilfen mit <ALT>+<H>',
  ignore, wait, enter
y,h=2*enter, clear, 1f, 1f, 10*cur_ri,
  'Beenden des Editors mit Anweisung B',
  1f, 1f, 30*cur_ri,
  'Weiter mit Leertaste',
  ignore, cmd_line, wait, 'zu*', enter
*eof
```

Der Editor wird aufgerufen und eine erste Information wird auf den leeren Bildschirm geschrieben. Der Steuerbefehl *IGNORE* bewirkt, daß anschließend der Informationstext nicht als Anweisung oder als Daten interpretiert, sondern ignoriert wird. *WAIT* bedeutet, daß die Abarbeitung des Editormakros unterbrochen wird, bis die Leertaste betätigt wird. Das Editormakro wird abgebrochen, wenn irgendeine andere Taste betätigt wird. Im vorliegenden Fall ist es so, daß sowohl bei der Leertaste als auch bei *ENTER*, wie bei Befolgung der Information und Betätigen von *<PgDn>* jeweils das gleiche erfolgt. Ähnliches gilt für die zusätzliche Hilfe, die man sich mit *<ALT>+<H>* bzw. *<PF1> <H>* geben lassen kann.

Die zuletzt vorgestellten Editormakros sollen nur das Vorgehen für Definition und Aufruf von Editormakros erläutern, die in den Editoraufruf integriert sind. Welche Hilfstexte in einer Anwendung sinnvoll sind und welche Parameter der aufzurufenden Kommandofolge man in das Editormakro integriert, ist von Fall zu Fall verschieden.