JAnE

Harald Becker/Andreas Hoffmann

COLLABORATORS			
	TITLE : JAnE		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Harald Becker/Andreas Hoffmann	February 16, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

JAnE

1.1 Just Another Editor

SCEleton reuser manual with port

(Symbolic Code Expansion language/extended tool nomenclature)
(Dieses Akronym (c) 1992 Andreas Hoffmann)

Kontaktadresse:
Harald Becker
Ferrenbergstrae 41
51465 Bergisch Gladbach
Tel. 02202/34913
Harald Becker @ K (MausNet)

1.2 einfhrung

Einfhrung SCELREF.HYP
Die folgenden Abschnitte beschreiben die Programmiersprache SCEleton, die als Makrosprache in den Texteditor JAnE integriert ist, ziemlich detailliert. Er setzt dabei grundlegende Programmierkenntnisse voraus, die allerdings nicht auf eine bestimmte Sprache bezogen sein mssen. Wenn gelegentlich auf C verwiesen wird, so liegt das daran, da der Autor sich bei einigen Eigenarten von SCEleton von C hat inspirieren lassen. Grundlegende Konzepte, wie Variablen, Unterprogramme, strukturierte Programmierung etc. sollten zumindest in Umrissen bekannt sein. Andernfalls sollten Sie lieber zunchst die Datei SCELINTRO.TXT durcharbeiten, die die Grundlagen von SCEleton anhand von Beispielen beschreibt und dabei diese Themen nher erklrt.

Also ans Werk: Bei SCEleton handelt es sich um eine vollwertige Programmiersprache, die die Methoden der strukturierten Programmierung untersttzt. Sie bietet fr die meisten Anweisungen zwei Syntaxformen an, von denen die eine auf leichte Lesbarkeit ausgelegt ist, whrend die andere sich an der Sprache C orientiert und daher krzer ist. Im Gegensatz zu C ist SCEleton aber eine Sprache ohne Datentypen, d.h. eine Variable kann sowohl eine Zahl als auch eine Zeichenkette als auch eine Liste von Werten enthalten und ihren Datentyp whrend der Laufzeit eines Programms auch ndern. SCEleton ist eine "kleine" Programmiersprache, die gezielt fr den Einsatz als Makrosprache konzipiert ist. Daraus erklren sich auch einige Einschrnkungen, so etwa das Fehlen von Fliekommaoperationen.

Der Interpreter
Die Ausfhrung

1.3 der interpreter

Der Interpreter SCELREF.HYP
SCEleton ist eine interpretierte Sprache, d.h. anders als etwa bei C oder Modula 2 werden die Programme nicht in die Maschinensprache der 680x0-Prozessoren bersetzt, sondern von einem weiteren Programm, dem sog. Interpreter, ausgefhrt. Dieser Interpreter ist ein Bestandteil von JAnE.

Er verarbeitet ein SCEleton-Makropaket in zwei Schritten. Zunchst wird das Programm syntaktisch analysiert (dabei werden smtliche Syntaxfehler aufgedeckt) und das Programm (sofern es fehlerfrei ist) in eine tokenisierte Zwischendarstellung bersetzt. Anschlieend wird dieser Zwischencode ausgefhrt.

Um sich einen weiteren Analyselauf zu sparen, knnen Sie von JAnE aus ein fehlerfreies Paket auch in dieser Zwischendarstellung abspeichern. Normalerweise wird immer eine Quelldatei zu einem Makropaket bersetzt; Sie knnen aber auch unter JAnE mit dem Menubefehl "Hinzuladen" zwei bersetzte (d.h. im Zwischencode vorliegende) Makropakete zu einem verbinden.

Wenn Sie JAnE unter MagiCMac auf einem Power Mac laufen lassen und dann einen Makro ausfhren, arbeitet dort also ein Interpreter auf einem Interpreter (nmlich die SCEleton-Maschine auf der 68K-Emulation der Power-PCs). Aber darber denkt man besser nicht allzu genau nach ;-)

1.4 die ausfhrung

Die Ausfhrung SCELREF.HYP
Im Gegensatz zu einem C-Programm hat ein SCEleton-Paket keinen fest definierten Startpunkt. Stattdessen knnen Sie unter JAnE jeden Makro, der keine Parameter erwartet, durch einen Klick in das Fenster des entsprechenden Makropaketes starten. Eine andere Mglichkeit besteht darin, einen solchen Makro auf eine

Funktionstaste zu legen und durch einen Tastendruck zu starten.

Mit einem solchen Start beginnt eine Makrosequenz, die diesen Makro und alle weiteren, die von ihm aufgerufen werden, umfat. Eine solche Sequenz endet dann, wenn der erste Makro endet, oder wenn ein Laufzeitfehler auftritt, der das Weiterarbeiten unmglich macht. Mit dem Ende der Sequenz werden smtliche Variablen gelscht. Eine Methode, Daten ber das Sequenzende hinaus im RAM zu speichern, beschreibt der Abschnitt ber das SCEleton-Environment im Kapitel ber die Laufzeitfunktionen.

Im Falle eines solchen Laufzeitfehlers erscheint unter JAnE im Fenster SCEleton eine Meldung, die die Fehlerursache und die Zeilennummer der Fehlerstelle angibt. Wenn der entsprechende Quelltext geladen ist, wird er angezeigt und die Fehlerzeile markiert.

1.5 die sprachbeschreibung

Die Sprachbeschreibung SCELREF.HYP
Das Programmformat
Die Datentypen
Die Programmstruktur

1.6 das programmformat

Das Programmformat SCELREF.HYP
In einem SCEleton-Programm drfen alle Zeichen auer dem Nullbyte vorkommen. Sie knnen die Programme im freien Format eingeben, d.h. Leerzeichen, Zeilenwechsel und Seitenvorschbe knnen beliebig zwischen Namen, Operatoren, Kommentaren etc. verteilt werden. Es knnen also insbesondere auch mehrere Befehle in einer Zeile stehen. Sie sollten diese Freiheit nutzen, um Ihre Programme so lesbar wie mglich und nicht so kompakt wie mglich zu gestalten. Die einzelnen lexikalischen Grundbausteine sind im Folgenden beschrieben:

Kommentare
Bezeichner
Notation fr Zahlen
Notation fr Zeichenketten
Sprungmarken
Das Nullsymbol

1.7 kommentare

Kommentare SCELREF.HYP
Ein SCEleton-Programm kann Kommentare enthalten, d.h. Anmerkungen, die vom Interpreter schlicht ignoriert werden. Fr Sie knnen diese Kommentare aber um so wertvoller sein, wenn Sie sich ein Programm nach lngerer Zeit noch einmal vornehmen, zum Beispiel um es zu ndern. Ein Kommentar wird mit einem Lattenkreuz '#' eingeleitet und

reicht immer bis zum Ende der Zeile, in der das Lattenkreuz steht.

```
# Dies ist ein Kommentar
```

1.8 bezeichner

Bezeichner SCELREF.HYP
Bezeichner, d.h. die Namen von Makros oder Variablen, mssen mit einem Buchstaben - mit Ausnahme der Umlaute und des '' - oder dem Unterstrich '_' beginnen, auf den dann weitere Buchstaben, Ziffern oder Unterstriche folgen knnen. Ein Name kann bis zu 24 Zeichen umfassen, die alle unterschieden werden. Gro- und Kleinbuchstaben werden unterschieden, so da 'ABCD' und 'abcd' zwei verschiedene Namen sind.

Gltige Namen sind also z.B. 'abcd', 'a123', '_test', 'dies_ist_ein_Name'. Ein ungltiger Name ist dagegen '8abcd'.

1.9 notation fr zahlen

Notation fr Zahlen SCELREF.HYP
In einem SCEleton-Programm knnen Sie konstante Zahlen in einem der folgenden Zahlensysteme angeben: Dezimal, hexadezimal oder binr.

Eine Dezimalzahl schreiben Sie einfach als Zahl, d.h. als Reihe von Ziffern. Eine Hexzahl wird mit dem Prfix '0x' kenntlich gemacht, eine Binrzahl mit '0b'. Hinter einem solchen Prfix drfen Sie natrlich nur Zeichen benutzen, die Ziffern des jeweiligen Zahlensystems sind, also fr Binrzahlen nur '0' und '1'. Die sechs zustzlichen Ziffern fr Hexzahlen werden, wie allgemein blich, durch die Buchstaben 'a' bis 'f' ausgedrckt.

1.10 notation fr zeichenketten

Notation fr Zeichenketten SCELREF.HYP
Manchmal bentigen Sie in SCEleton-Programmen auch konstante Zeichenketten. Hier bietet SCEleton zwei Formate an. Wie allgemein blich, mssen Sie Zeichenketten dadurch kenntlich machen, da Sie sie zwischen Begrenzerzeichen einschlieen. Im ersten, simpleren Format sind das zwei Apostrophe. Dieses Format untersttzt innerhalb der Zeichenkette keinerlei Spezialitten, die Zeichenkette kann also nur Zeichen enthalten, die Sie direkt eingeben knnen. Um in einer solchen Zeichenkette einen Apostroph unterzubringen, mssen Sie zwei Apostrophe hintereinandersetzen.

Das zweite Format ist komplizierter, bietet aber auch mehr Mglichkeiten. Es ist an die Programmiersprache C angelehnt. Die Begrenzerzeichen sind hier Anfhrungszeichen. Eine solche Zeichenkette kann sogenannte Escapezeichen enthalten, die von einem Rckstrich (Backslash) eingeleitet werden und jeweils ein Zeichen

reprsentieren, das nicht ohne weiteres ber die Tastatur erreicht werden kann. Die folgenden Escapesequenzen sind zulssig:

```
'\n'        erzeugt ein LF (ASCII 10)
'\r'        erzeugt ein CR (ASCII 13)
'\b'        erzeugt einen Backspace (ASCII 8)
'\t'        erzeugt einen Tab (ASCII 9)
'\f'        erzeugt einen Seitenvorschub (ASCII 12)
'\xhh'      erzeugt ein beliebiges ASCII--Zeichen. 'hh' sind zwei
            Hexziffern, die den ASCII-Code angeben.
'\'        erzeugt einen Rckstrich.
'\"        erzeugt ein Anfhrungszeichen innerhalb der
            Zeichenkette.
```

In allen anderen Kombinationen wird der Backslash ignoriert und aus der Zeichenreihe entfernt.

Hierbei ist eines zu beachten: Es gibt eine Art von Zeichenketten, die fast immer Backslashes enthalten, nmlich Dateinamen mit Pfaden. In diesem Format mssen dabei die Backslashes immer doppelt angegeben werden. (Das ist brigens auch ein typischer Fehler von C-Anfngern auf dem Atari.) Da aber Dateinamen wohl nur selten die exotischen Zeichen enthalten, die das zweite Format ermglicht, gengt wohl meist das erste Format.

Beispiel fr das erste Format:

```
'''Space Invaders'' ist ein tolles Spiel'
```

ergibt die Ausgabe:

```
'Space Invaders' ist ein tolles Spiel
```

hnliches im zweiten Format:

```
"\" Space Invaders\" ist ein tolles Spiel"
```

ergibt die Ausgabe:

```
"Space Invaders" ist ein tolles Spiel
```

Ein Dateiname im zweiten Format:

```
"C:\JANE\JANE.APP"
```

ergibt die Ausgabe:

```
C:\JANE\JANE.APP
```

Im ersten Format gengt dafr:

```
'C:\JANE\JANE.APP'
```

1.11 sprungmarken

Sprungmarken SCELREF.HYP
Im Zusammenhang mit dem unbedingten Sprungbefehl ’goto’ (jawohl, den gibt es hier auch!) ist die Verwendung von Sprungmarken (Labels) erforderlich. Eine Marke ist ein Name, gefolgt von einem Punkt. Beispiel:

```
Das_ist_eine_Marke.
```

1.12 das nullsymbol

Das Nullsymbol SCELREF.HYP
Das Nullsymbol ’$$$’ bezeichnet ein Datenelement, das je nach Kontext die Zahl ’0’, eine leere Zeichenkette oder eine leere Liste beschreibt. Zur Verwendung siehe unter "Alleinstehende Ausdrcke".

1.13 die datentypen

Die Datentypen SCELREF.HYP
Die Sprache SCEleton kennt drei grundlegende Datentypen: Zahlen, Zeichenketten und Listen von Werten. Anders als etwa in C brauchen Sie aber nicht fr jede Variable einen Datentyp festzulegen, den sie ab dann fr immer behlt. Statt dessen hat das jeder Wert bzw. jedes Ergebnis einer Berechnung einen impliziten Typ, und sobald Sie ein solches Ergebnis in einer Variablen speichern, nimmt diese neben dem Wert auch den Typ des Ergebnisses an. Sie knnen also in einer Variablen jederzeit jeden dieser drei Typen speichern. Ebenso knnen Sie das Nullsymbol einer Variablen zuweisen, um sie zu lschen.

Der Datentyp Zahl
Der Datentyp Zeichenkette
Der Datentyp Liste

1.14 der datentyp zahl

Der Datentyp Zahl SCELREF.HYP
Der Datentyp "Zahl" wird dargestellt durch eine vier Byte groe, vorzeichenbehaftete Ganzzahl. Der Zahlenbereich reicht also von -2147483648 bis 2147483647. Es handelt sich stets um ganze Zahlen; bei Divisionen, die Nachkommastellen ergeben wrden, werden diese Stellen abgeschnitten. SCEleton untersttzt keine Fliekommaarithmetik und kennt demgem keinen entsprechenden Datentyp.

1.15 der datentyp zeichenkette

Der Datentyp Zeichenkette SCELREF.HYP
Eine Zeichenkette kann bis zu 4096 Zeichen umfassen. Anders als in C darf sie auch das Nullbyte als Zeichen enthalten. Sie knnen in Zeichenketten also beliebige binre Daten verarbeiten. Zeichenketten werden vollstndig dynamisch verwaltet; Sie brauchen die Lnge einer Zeichenkette also nicht vorher festzulegen. Sie knnen an jeder Stelle Teilzeichenreihen einfgen oder lschen.

1.16 der datentyp liste

Der Datentyp Liste SCELREF.HYP
Eine Liste kann als Elemente Zahlen, Zeichenketten oder weitere Listen enthalten. Auch Listen werden dynamisch verwaltet; ihre Lnge (d.h. die Anzahl der Elemente) ist nur durch den freien Speicher begrenzt. Im Gegensatz zu den Datenstrukturen anderer Sprachen (Arrays oder Records) kann sich die Lnge einer Liste whrend der Programmlaufzeit ndern. Sie knnen sogar an beliebiger Stelle neue Elemente einfgen oder Elemente lschen.

Eine Liste kann Zahlen, Zeichenketten und weitere Listen enthalten. Diese letzte Variante ermglicht Ihnen, auch mehrdimensionale Datenstrukturen aufzubauen.

1.17 die programmstruktur

Die Programmstruktur SCELREF.HYP
Ein SCEleton-Quelltext, der zu einem Makropaket bersetzt wird, besteht aus einer Reihe von Deklarationen und Makros. Ein Makro enthlt eine Reihe von Befehlen, die der Reihe nach ausgefhrt werden, sofern diese Reihenfolge nicht durch Kontrollstrukturen verndert wird. Die Ausfhrung beginnt, wenn der Makro von JAnE aus durch einen Menubefehl gestartet oder von einem anderen Makro aus aufgerufen wird. Eine Deklararation kann entweder eine globale Variable definieren, die von allen Makros aus gleichermaen anzusprechen ist, oder es handelt sich um eine Vorwrtsdeklaration eines Makros, dessen Text an anderer Stelle steht.

Makrodefinitionen
Vorwrtsdeklarationen
Blcke
Variable
Einfache Ausdrcke
Befehle

1.18 makrodefinitionen

Makrodefinitionen SCELREF.HYP
Ein Makro besteht aus einem Kopf und einem Anweisungsblock. Der Anweisungsblock enthlt die Befehle, die im einzelnen angeben, wie der Makro seine Aufgabe erfllen soll. Der Kopf legt den Namen des Makros sowie eventuelle bergabeparameter fest. Bei den bergabeparametern handelt es sich um Variablen, die vor dem Aufruf des Makros mit Werten gefllt werden. Auf diese Art kann man dem Makro beim Aufruf Informationen mitgeben, die etwa seine Arbeit steuern. Der Kopf beginnt mit dem Makronamen, fr den die oben erluterten Regeln fr Namen gelten. Dahinter steht in Klammern die Liste der Parameter, durch Kommas getrennt.

Beispiele fr Kpfe:

```
    Makro1()        # Dieser Makro hat keine Parameter
    Makro2(p1)      # dieser einen: p1
    Makro3(p1, p2)  # ...und dieser zwei: p1 und p2
```

Ein Makroname mit einer besonderen Bedeutung ist ’errortrap’. Diesen Namen sollten Sie nicht fr beliebige Makros verwenden; er ist reserviert fr einen Makro, der im Falle eines Laufzeitfehlers vom System aufgerufen wird.

1.19 vorwrtsdeklarationen

Vorwrtsdeklarationen SCELREF.HYP
Eine Vorwrtsdeklaration hat die Form

```
    forward Kopf;
```

"Kopf" steht fr den Kopf eines Makros, wie oben unter "Makrodefinitionen" beschrieben. Eine Vorwrtsdeklaration ist immer dann erforderlich, wenn der Makro aufgerufen werden soll, der nicht vorher im selben Quelltext definiert wurde. Unumgnglich ist dies aber nur im Falle von zyklischer Rekursion. Wenn SCEleton zu einer Vorwrtsdeklaration keine Auflsung findet, dann wird whrend der Compilierung eine Fehlermeldung erzeugt.

1.20 blcke

Blcke SCELREF.HYP
Ein Anweisungsblock ist eine Liste von Befehlen, die zwischen einem Blockanfang und einem Blockende stehen. Der Blockanfang ist das Schlsselwort "begin" oder die offene geschweifte Klammer ’{’. Das Block-ende ist das Schlsselwort "end" oder die geschlossene geschweifte Klammer ’}’ (damit werden die Pascal- und Modula-Freunde hoffentlich genauso glcklich wie die C-Freunde). Es ergibt sich also folgende Grobstruktur fr einen Makro:

```
Kopf
    Blockanfang
```

```
        Befehl
        Befehl
        .
        .
        .
    Blockende
```

1.21 variable

Variable SCELREF.HYP
Die Befehle eines SCEleton-Programmes bearbeiten Daten, die sich in bestimmten Speicherpltzen im Rechner befinden. Diese Speicherpltze heien Variable. Um auf diese Variablen im Programm Bezug nehmen zu knnen, erhalten sie Namen. Fr diese Namen gelten die oben unter "Bezeichner" aufgefhrten Regeln.

Eine Variable enthlt entweder eine Zahl, eine Zeichenkette, oder eine Liste von Werten. Dies sind die drei grundlegenden Datentypen von SCEleton. Anders als etwa in C brauchen Sie aber nicht fr jede Variable einen Datentyp festzulegen, den sie ab dann fr immer behlt. Statt dessen hat das jeder Wert bzw. jedes Ergebnis einer Berechnung einen impliziten Typ, und sobald Sie ein solches Ergebnis in einer Variablen speichern, nimmt diese neben dem Wert auch den Typ des Ergebnisses an. Sie knnen also in einer Variablen jederzeit jeden dieser drei Typen speichern. Ebenso knnen Sie das Nullsymbol einer Variablen zuweisen, um sie zu lschen.

Globale Variable
Lokale Variable

1.22 globale variable

Globale Variable SCELREF.HYP
Globale Variable sind Variable, die von mehreren Makros aus angesprochen und gendert werden knnen. Sie knnen also zum Austausch von Daten zwischen Makros verwendet werden, auch wenn diese Methode nach den Regeln der strukturierten Programmierung nicht immer empfehlenswert ist. Globale Variable mssen deklariert werden. Dazu dient die 'global'-Anweisung, die auerhalb eines Makros stehen mu. Sie hat die Form

```
    global Name [, Name];
```

'Name' steht hier fr gltige Variablennamen. Die eckigen Klammern sind so zu lesen, da hinter 'global' ein oder mehrere Namen stehen drfen. Bei mehreren Namen sind Kommas als Trennzeichen zu verwenden. Die 'global'-Anweisung mu mit einem Semikolon abgeschlossen werden.

1.23 lokale variable

Lokale Variable SCELREF.HYP
Eine lokale Variable ist eine Variable, deren Name nur innerhalb eines Makros bekannt ist. Daraus folgt, da eine solche Variable auerhalb eines Makros nicht angesprochen und insbesondere nicht verndert werden kann. Anders als in C oder Modula brauchen lokale Variable in SCEleton nicht deklariert zu werden, sondern sie werden bei der ersten Nutzung angelegt. Das ist in einer 'kleinen' Programmiersprache einfach praktischer.

Eine dadurch neu angelegte Variable hat immer implizit den Typ 'Zahl' und den Wert 0, falls ihr nicht sofort etwas anderes zugewiesen wird.

1.24 einfache ausdrcke

Einfache Ausdrcke SCELREF.HYP
Ein einfacher Ausdruck ist eine Formel, deren Wert von SCEleton berechnet wird. In einer solchen Formel werden Datenelemente durch Operatoren verknpft.

Datenelemente
Operatoren
Zugriff auf Variable
Funktionsaufrufe

1.25 datenelemente

Datenelemente SCELREF.HYP
Ein Datenelement kann in eine Konstante oder eine Variable sein. Die Regeln fr Konstante wurden bereits oben besprochen. Eine Variable wird einfach durch Angabe ihres Namens angesprochen.

1.26 operatoren

Operatoren SCELREF.HYP
In SCEleton werden Ausdrcke in der Infix-Schreibweise notiert, d.h. ein zweistelliger Operator (das ist ein Operator, der zwei Gren miteinander verknpft) steht zwischen seinen Operanden (Beispiel: 1+1). Ein einstelliger Operator (also ein Operator, der nur auf eine Gre wirkt), steht vor seinem Operanden.

In SCEleton hat jeder Operator eine Prioritt, die die Reihenfolge der Rechnungen bei der Auswertung eines Ausdruckes bestimmt. Dies stellt u.a. die Einhaltung der bekannten Rechenregel "Punkt- vor Strichrechnung" sicher. Operatoren gleicher Prioritt werden von links nach rechts ausgewertet. Durch den Einsatz von Klammern knnen Sie diese Berechnungsreihenfolge bersteuern. So hat etwa der Ausdruck '(4 + 3) * 2' den Wert 14, der Ausdruck '4 + 3 * 2' dagegen den Wert 10.

SCEleton stellt fr einfache Ausdrcke die folgenden Operatoren zur Verfgung. Sie sind hier nach aufsteigender Prioritt sortiert, d.h. die untersten werden zuerst ausgewertet. Operatoren in einer Gruppe haben die gleiche Prioritt. Alle Operatoren sind zweistellig, bis auf die letze Gruppe, die nur einstellige Operatoren enthlt.

'||'	logisches oder
'&&'	logisches und
'|'	binres oder
'^'	exklusives oder (binr)
'&'	binres und
'=='	gleich
'!='	ungleich
'<'	kleiner als
'<='	kleiner oder gleich
'>'	grer als
'>='	grer oder gleich
'?'	durchsucht eine Zeichenkette nach einem Teilstring bzw. eine Liste nach einem bestimmten Element.
'//'	Fgt zwei Zeichenketten zusammen
'++'	Fgt zwei Listen zusammen
'**'	Erzeugt eine Liste durch Wiederholung eines Elementes
'+'	Addition
'-'	Subtraktion
'*'	Multiplikation
'/'	Division
'%'	Modulo-Operation
'!'	logisches nicht
'~'	binres nicht
'+'	unres plus
'-'	unres minus
'@'	Lnge einer Liste bzw. einer Zeichenreihe

Bemerkungen:

- Die logischen Operatoren 'und', 'oder' und 'nicht' verknpfen Bedingungen, die etwa von den Vergleichsoperatoren geliefert werden. Eine Bedingung hat immer den Wert 'wahr' oder 'falsch'. 'falsch' wird intern durch eine 0 dargestellt, whrend jeder Wert ungleich 0 als 'wahr' interpretiert wird. Ihre binren Gegenstcke verknpfen zwei Ganzzahlen Bit fr Bit.

- Die Operatoren, die hier fr Zeichenketten angegeben werden, knnen auch auf Zahlen angewandt werden. Dann wird eine Zahl als Ziffernfolge aufgefat. Der Ausdruck '12 // 34' etwa hat den Wert '1234'. Umgekehrt kann man mit den resultierenden Zeichenfolgen wieder rechnen, sofern diese als Zahl zu interpretieren sind. Der Ausdruck '(12 // 34) + 1' hat dann den Wert 1235.

- Da SCEleton nur mit Ganzzahlen rechnet, werden bei einer Division evtl. entstehende Nachkommastellen abgeschnitten.

- Beim Operator '?' steht links das Objekt, das durchsucht werden soll, und rechts das Objekt, das gesucht werden soll. Wenn links eine Zeichenkette steht, wird das rechte Objekt darin als Teilzeichenkette gesucht. Steht links eine Liste, wird es als Listenelement gesucht. Das Ergebnis dieser Operation ist eine Zahl, die die Position des gesuchten im durchsuchten Objekt angibt. Die Zhlung beginnt immer bei 0. Das Ergebnis ist -1, wenn nichts gefunden wurde. Beispiel:

```
    dump('Der Test' ? 'Test');      # gibt 4 aus
    dump('Der Test' ? 'XXX');       # gibt -1 aus
```

(In diesem und den folgenden Beispielen werden die Laufzeitfunktionen 'dump', 'print' und 'lprint' verwendet, die unter JAnE Daten in das 'SCEleton'-Fenster ausgeben. Irgendwie mu man ja Ergebnisse sichtbar machen. Die genaue Erklrung dieser Funktionen folgt im Abschnitt ber die Laufzeitumgebung.)

- Der Operator '**' dient dazu, schnell eine initialisierte Liste zu erzeugen, in der alle Elemente identisch sind. Dazu steht links das zu wiederholende Datenobjekt, rechts die Anzahl der Wiederholungen (also die Lnge der Liste). Beispiel:

```
    dump('Test' ** 3);  # gibt [Test Test Test] aus
```

- Der Operator '//' fgt zwei Zeichenkette zu einer zusammen; der Operator '++' fgt zwei Listen zu einer oder zwei Elemente zu einer Liste zusammen oder verlngert eine Liste um ein Element. Beispiel:

```
    dump(12 // 34);                 # gibt 1234 aus
    dump(12 ++ 34);                 # gibt [12 34] aus
```

- Der Operator '@' bestimmt die Anzahl der Zeichen in einer Zeichenreihe oder die Anzahl der Elemente in einer Liste. Eine Zahl wird als Zeichenreihe interpretiert, so da der Wert von '@' die Anzahl der Ziffern ist.

1.27 zugriff auf variable

Zugriff auf Variable SCELREF.HYP

Um in einer Rechnung den Wert einer Variable zu verwenden, geben Sie einfach den Namen dieser Variablen als Operand des entsprechenden Operators an. Bei komplexen Variablen (Zeichenketten oder Listen) ist es gelegentlich erwnscht, auf bestimmte Teile der Variablen zuzugreifen. Dies leisten die Index- und die Bereichsnotation.

Indexnotation
Bereichsnotation

1.28 indexnotation

Indexnotation SCELREF.HYP

Mit der Indexnotation greifen Sie gezielt auf ein Element einer

komplexen Variablen (d.h. auf ein Zeichen einer Zeichenkette bzw. auf ein Element einer Liste) zu. Dazu geben Sie hinter dem Namen der Variable in eckigen Klammern '[]' einen Ausdruck an. Dieser Ausdruck mu als Wert eine Zahl ergeben, die die Nummer (den Index) des gewnschten Elementes angibt. Zeichen in einer Zeichenkette und Listenelemente sind fortlaufend durchnummeriert. Die Nummerierung beginnt bei 0.

Beispiel: Wenn 'a' die Zeichenkette 'abcde' enthlt, ergibt 'a[1]' den Wert 'b'. (Der Einfachheit halber wird hier eine Zahl als Index verwendet; hier kann natrlich auch eine komplizierte Rechnung stehen.)

Wenn der berechnete Index negativ oder grer als die aktuelle Lnge der Variablen (d.h. die Anzahl der Elemente) ist, wird die laufende Sequenz mit einem Laufzeitfehler abgebrochen. Um sich dagegen abzusichern, knnen Sie mit dem Operator '@' die Lnge des Objektes ermitteln und den Index vor dem Zugriff berprfen.

Da eine Liste als Elemente Zeichenketten oder andere Listen enthalten kann, ist es mglich, mehrere Indices anzugeben, um auf ein Element eines Elementes zuzugreifen. Ein solcher Zugriff sieht etwa so aus: 'a[2][3]'.

1.29 bereichsnotation

Bereichsnotation SCELREF.HYP

Mit der Bereichsnotation greifen Sie aus einer komplexen Variablen einen zusammenhngenden Teilbereich heraus, also aus einer Zeichenkette eine krzere Zeichenkette oder aus einer Liste eine krzere Liste. Die Bereichsnotation hnelt der Indexnotation. Hier geben Sie hinter dem Namen in eckigen Klammern zwei Ausdrcke, getrennt durch einen Doppelpunkt ':' an. Der erste Ausdruck liefert den Index des ersten herauszugreifenden Elementes, der zweite Ausdruck liefert die Lnge des gewnschten Teilobjektes.

Beispiel: Wenn 'a' die Zeichenkette 'abcde' enthlt, ergibt 'a[2:2]' den Wert 'cd'. (Der Einfachheit halber wird hier eine Zahl fr Index und Lnge verwendet; hier knnen natrlich auch komplizierte Rechnungen stehen.)

Wenn der berechnete Teilbereich nicht ganz im Innern des Objektes liegt, wird die laufende Sequenz mit einem Laufzeitfehler abgebrochen. Um sich dagegen abzusichern, knnen Sie mit dem Operator '@' die Lnge des Objektes ermitteln und den Index vor dem Zugriff berprfen.

Da eine Liste als Elemente Zeichenketten oder andere Listen enthalten kann, ist es mglich, mehrere Indices anzugeben, um auf ein Element eines Elementes zuzugreifen. Ein Bereich kann nur an Stelle des letzten Index stehen. Ein solcher Zugriff sieht etwa so aus: 'a[2][3:2]'.

1.30 funktionsaufrufe

Funktionsaufrufe SCELREF.HYP
In einem Ausdruck kann zu guter Letzt noch ein Aufruf einer Funktion stehen. 'Funktion' bedeutet dabei entweder eine vordefiniertes Unterprogramm der Laufzeitumgebung oder einen anderen Makro, der einen Wert zurckliefert. Die Syntax eines solchen Aufrufes lautet:

```
name(Ausdruck1, Ausdruck2)
```

'name' ist der Name der Funktion. Die mglichen Namen fr Laufzeitunterprogramme finden Sie im Abschnitt ber die Laufzeitumgebung. Der Name eines Makros wird in seiner Kopfzeile festgelegt. 'Ausdruck1' und 'Ausdruck2' sind zwei Ausdrcke, deren Werte (in diesem Beispiel) als erster und zweiter Parameter bergeben werden. Es knnen auch mehr oder weniger sein. Erwartet die Funktion keine Parameter, so lautet der Aufruf:

```
name()
```

Bei nur einem Parameter lautet die Syntax:

```
name(Ausdruck)
```

Jeder Makro und jedes Laufzeitunterprogramm liefert auch einen Wert zurck (im Zweifel immer 0). Wenn der Aufruf in einem Ausdruck erscheint, wird die Auswertung des Ausdrucks mit diesem Wert fortgesetzt. Es sei beispielsweise folgender Makro definiert, der die Summe seiner Parameter berechnet und zurckgibt:

```
addiere(a, b)
{   return a + b;
}
```

Dann hat der Ausdruck

```
1 + addiere(2, 3)
```

den Wert '6'. Die Auswertung von Funktionsaufrufen hat eine hhere Prioritt als alle Operatoren. Ein Makro kann sich auch direkt oder indirekt selber aufrufen (Rekursion). Ein klassisches Beispiel dafr ist die rekursive Definition der Fakulttsfunktion:

```
# Berechne fak(n)
# fak(n) := n * (n - 1) * (n - 2) * ... * 2 * 1
# fak(0) := 1
# Beispiele: fak(3) = 6, fak(6) = 720
fak(n)
{   if ( n == 0 )
        return 1;
    else
        return n * fak(n - 1);
}
```

Dies ist natrlich kein guter Algorithmus zur Berechnung der Fakultt einer Zahl, es zeigt aber das Prinzip der Rekursion. Im Zusammenhang

mit Fllen indirekter Rekursion (a ruft b, b ruft c, c ruft wieder a)
knnen Vorwrtsdeklarationen unumgnglich werden.

1.31 befehle

Befehle SCELREF.HYP
SCEleton kennt drei Arten von Befehlen: Kontrollbefehle, Zuweisungen
und alleinstehende Ausdrcke. Diese drei Kategorien berschneiden sich
teilweise, so kann z.B. eine Zuweisung sowohl als Befehl als auch als
Ausdruck geschreiben werden. Befehle knnen nur innerhalb von Makros
stehen. Ein Befehl wird immer mit einem Semikolon ';' abgeschlossen.

Zuweisungen
Kontrollbefehle
Alleinstehende Ausdrcke

1.32 zuweisungen

Zuweisungen SCELREF.HYP
Zuweisungen sind Befehle, die den Wert eines Datenelementes ndern.
Ein Datenelement kann dabei eine Variable, ein Listenelement oder ein
Teil einer Zeichenkette sein. Der zugewiesene Wert wird als Ergebnis
eines Ausdrucks bestimmt, d.h. er kann aus einem anderen Datenelement
stammen oder Resultat einer Berechnung sein. Es gibt unterschiedliche
Befehle, je nachdem, ob Sie eine Variable oder einen Teil einer
Zeichenkette oder Liste ndern mchten. In der Kategorie
'Zuweisungen' werden auch die Befehle zum Einfgen und Lschen von
Datenelementen behandelt.

Zuweisung an ein Datenelement: 'set...to'
Zuweisung an einen Teilbereich: 'replace...by'
Einfgen in ein komplexes Objekt: 'at...insert'
Lschen von Daten aus einem Objekt: 'delete'

Eine abgekrzte Schreibweise dieser Befehle wird unter

Zuweisungsoperatoren

beschrieben.

1.33 zuweisung an ein datenelement: 'set...to'

Zuweisung an ein Datenelement: 'set...to' SCELREF.HYP
Die Syntax des 'set'-Befehls lautet:

```
    set Variable to Ausdruck;
```

Dabei ist fr 'Variable' der Name einer Variablen einzusetzen, fr
'Ausdruck' der Ausdruck, dessen Ergebnis der Variablen zugewiesen
wird. Die Variable erhlt den Typ und den Wert des Ergebnisses.

Anstelle einer Variablen kann auch ein Listenelement in Indexnotation stehen. Beachten Sie bitte: Wenn Sie einen Variablennamen einsetzen, der bisher weder als global deklariert noch zuvor im Makro benutzt wurde, wird im Makro eine neue lokale Variable angelegt.

Beispiele:

```
    set a to 3;
```

Dieser Befehl weist 'a' den Typ Zahl und den Wert 3 zu.

```
    set a[4] to 3;
```

Falls 'a' eine Liste von fnf oder mehr Elementen ist, ist wird dem fnften Element von 'a' der Typ Zahl und der Wert 3 zugewiesen. Andernfalls wird die laufende Sequenz mit einem Laufzeitfehler abgebrochen.

1.34 zuweisung an einen teilbereich: 'replace...by'

Zuweisung an einen Teilbereich: 'replace...by' SCELREF.HYP
Mit der 'replace'-Anweisung knnen Sie einen Teilbereich einer komplexen Variablen durch andere Daten ersetzen. Die Syntax lautet:

```
    replace Teilbereich by Ausdruck;
```

Der 'Ausdruck' bestimmt wieder den zuzuweisenden Wert.

Beispiele:

1. Die Befehlsfolge

```
    set a to 'Dies ist ein Beispiel.';
    replace a[0:4] by 'Das';
    dump(a);
```

erzeugt die Ausgabe

```
    Das ist ein Beispiel.
```

2. Die Befehlsfolge

```
    set a to list(1, 2, 3, 4, 5);
    replace a[1:3] by list(7, 8, 9);
    dump(a);
```

erzeugt die Ausgabe

```
    [1 7 8 9 5]
```

(Die erste Zeile dieses Beispiels weist 'a' eine Liste aus den Zahlen 1, 2, 3, 4, 5 zu.)

1.35 einfgen in ein komplexes objekt: ’at...insert’

```
Einfgen in ein komplexes Objekt: ’at...insert’            SCELREF.HYP
```

Mit der ’insert’-Anweisung fgen Sie Daten in eine Liste oder eine Zeichenreihe ein. Der Rest des Objektes wird nach hinten geschoben, um fr die neuen Daten Platz zu schaffen. Die Syntax lautet:

```
    at Element insert Ausdruck;
```

Auch hier bestimmt ’Ausdruck’ den zuzuweisenden Wert. ’Element’ bezeichnet ein Listenelement oder Zeichen einer Zeichenkette in Indexnotation. Dadurch wird das Objekt bestimmt, in das die Daten einzusetzen sind, und die Einfgeposition. Der Wert von Ausdruck (oder sein erstes Element, falls es sich um eine komplexe Gre handelt) erscheint anschlieend an der durch den Index bestimmten Position im fraglichen Objekt.

Beispiele:

1. Die Befehlsfolge

```
    set a to ’Dies ein Beispiel.’;
    at a[4] insert ’ ist’;
    dump(a);
```

erzeugt die Ausgabe

```
    Dies ist ein Beispiel.
```

2. Die Befehlsfolge

```
    set a to list(1, 2, 3, 4, 5);
    at a[1] insert list(7, 8, 9);
    dump(a);
```

erzeugt die Ausgabe

```
    [1 7 8 9 2 3 4 5]
```

1.36 lschen von daten aus einem objekt: ’delete’

```
Lschen von Daten aus einem Objekt: ’delete’               SCELREF.HYP
```

Mit der ’delete’-Anweisung lschen Sie einen Teilbereich aus einem komplexen Objekt. Die Syntax lautet:

```
    delete Bereich;
```

’Bereich’ steht dabei fr die Angabe eines Teilbereichs, der das Objekt, in dem gelscht werden soll, und die zu lschenden Elemente bestimmt.

Beispiele:

1. Die Befehlsfolge

```
    set a to 'Dies ist ein langes Beispiel.';
    delete a[13:7];
    dump(a);
```

erzeugt die Ausgabe

```
    Dies ist ein Beispiel.
```

2. Die Befehlsfolge

```
    set a to list(1, 2, 3, 4, 5);
    delete a[1:1];
    dump(a);
```

erzeugt die Ausgabe

```
    [1 3 4 5]
```

1.37 kontrollbefehle

Kontrollbefehle SCELREF.HYP
Kontrollbefehle steuern den Ablauf eines Makros, meist anhand von Bedingungen, die durch logische Ausdrcke spezifiziert werden. Diese Ausdrcke knnen nur den Wert 'wahr' oder 'falsch' annehmen. Diese Ergebnisse werden intern durch Zahlen reprsentiert. Dabei wird der Wahrheitswert 'falsch' als 0 dargestellt, whrend alle Werte ungleich 0 'wahr' bedeuten.

Die Abfrage: 'if...elseif...else'
Die abweisende Schleife: 'while'
Die nichtabweisende Schleife: 'do...while'
Das Ende einer Schleife: 'break'
bergehen zum nchsten Punkt: 'continue'
Der unbedingte Sprung: 'goto'
Das Ende eines Makros: 'return'

1.38 die abfrage: 'if...elseif...else'

Die Abfrage: 'if...elseif...else' SCELREF.HYP
Die Bedingungsabfrage 'if' whlt, abhngig von einer Bedingung, zwischen zwei oder mehr alternativen Befehlspfaden aus. Die erste Variante lautet:

```
    if ( Ausdruck )
        Befehl1
```

Dabei ist 'Ausdruck' ein Ausdruck, der als Ergebnis eine Zahl liefert. Wenn dieses Ergebnis verschieden von 0 ist, wird die Bedingung als erfllt angesehen, andernfalls nicht. Wenn die Bedingung erfllt ist, wird die Anweisung 'Befehl1' ausgefhrt, ansonsten wird sie bersprungen und die Ausfhrung hinter 'Befehl1' fortgesetzt.

'Befehl1' kann auch ein Block sein, der mehrere Befehle enthlt. Die zweite Syntaxvariante lautet:

```
    if ( Ausdruck )
        Befehl1
    else
        Befehl2
```

Wenn die Bedingung 'Ausdruck' erfllt ist, wird 'Befehl1' ausgefhrt und 'Befehl2' bersprungen. Andernfalls wird 'Befehl2' ausgefhrt und 'Befehl1' bersprungen. Es wird also immer genau einer der beiden Zweige ausgefhrt. Die dritte und umfassendste Syntaxvariante lautet:

```
    if ( Ausdruck1 )
        Befehl1
    elseif ( Ausdruck2 )
        Befehl2
    else
        Befehl3
```

Wenn die Bedingung 'Ausdruck1' erfllt ist, wird 'Befehl1' ausgefhrt und alles weitere bersprungen. Andernfalls wird 'Ausdruck2' getestet. Ist dies erfllt, wird 'Befehl2' ausgefhrt und alles weitere ignoriert, ansonsten kommt 'Befehl3' an die Reihe. In dieser Variante kann mehr als ein 'elseif'-Zweig verwendet werden, so da mehr als zwei Bedingungen getestet werden. Ebenfalls kann der 'else'-Zweig weggelassen werden. In diesem Falle wird der Makro, falls keine der Bedingungen zutrifft, hinter dem letzten 'elseif'-Zweig fortgesetzt.

Bemerkung: Da Wahrheitswerte durch Zahlen dargestellt werden, kann eine Bedingung der Form 'if (i != 0)' durch 'if (i)' abgekrzt werden.

1.39 die abweisende schleife: 'while'

Die abweisende Schleife: 'while' SCELREF.HYP
Die 'while'-Schleife erlaubt die bedingungsgesteuerte Wiederholung von Befehlsfolgen. Die Syntax lautet:

```
    while ( Ausdruck )
        Befehl
```

'Ausdruck' ist wieder ein Ausdruck, der einen Wahrheitswert liefert. 'Befehl' kann auch ein Block sein. Beim Erreichen der 'while'-Anweisung wird zunchst die Bedingung berprft, und, falls sie falsch ist, wird 'Befehl' bersprungen. Falls sie zutrifft, wird 'Befehl' ausgefhrt, dann springt SCEleton zur 'while'-Anweisung zurck und berprft die Bedingung erneut. Dies wird solange wiederholt, bis die Bedingung falsch wird. Die Schleife heit 'abweisend', weil der Befehl bersprungen wird, wenn die Bedingung von vornherein falsch ist.

Bemerkung: SCEleton kennt keine Zhlschleife im Sinne der FOR-Schleife von BASIC. Diese lt sich jedoch mit Hilfe der 'while'-Schleife

leicht nachbilden. Im folgenden Code ist 'i' die Zhlvariable, 'n'
die gewnschte Zahl der Durchlufe:

```
    set i to 1;
    while ( i <= n )
    begin
        # Hier stehen Ihre Befehle
        set i to i + 1;
    end
```

Sie knnen auch ohne weiteres andere nderungen der Zhlvariablen als
'set i to i + 1;' verwenden, so z.B. 'set i to i * 2;'.

1.40 die nichtabweisende schleife: 'do...while'

Die nichtabweisende Schleife: 'do...while' SCELREF.HYP
Die Syntax dieser Schleifenkonstruktion lautet:

```
    do
        Befehl
    while ( Ausdruck )
```

'Ausdruck' ist wieder ein Ausdruck, der einen Wahrheitswert liefert.
'Befehl' kann auch ein Block sein. Im Unterschied zum vorherigen
Konstrukt wird hier der Schleifenkrper (d.h. 'Befehl') mindestens
einmal ausgefhrt (darum auch 'nichtabweisende' Schleife). Nach der
Abarbeitung des Schleifenkrpers wird die Bedingung 'Ausdruck'
getestet und, falls sie wahr ist, die Schleife wiederholt.
Andernfalls wird der Makro hinter 'while' fortgesetzt.

1.41 das ende einer schleife: 'break'

Das Ende einer Schleife: 'break' SCELREF.HYP
Der 'break'-Befehl dient dazu, eine 'while'- oder
'do...while'-Schleife vorzeitig, d.h. vor dem Eintreten der
Abbruchbedingung zu beenden. Die Syntax ist denkbar einfach:

```
    break;
```

Dieser Befehl wird meist mit einer 'if'-Abfrage von einer anderen
Bedingung abhngig gemacht werden, da sonst die Schleife nie bis zum
Ende ausgefhrt werden wird. Das folgende Beispiel versucht, von 1
bis 5 zu zhlen, kommt aber nur bis 3:

```
    i = 0;
    while ( i < 5 )
    {   i = i + 1;
        if ( i > 3 )
            break;
        lprint("%d", i);    # drucken
    }
```

Daher sieht die Ausgabe so aus:

```
1
2
3
```

1.42 bergehen zum nchsten punkt: ’continue’

bergehen zum nchsten Punkt: ’continue’ SCELREF.HYP
Mit der ’continue’-Anweisung springen Sie innerhalb einer Schleife sofort zur nchsten Abfrage der Abbruchbedingung, d.h. wenn diese Anweisung in der Mitte des Schleifenkrpers steht und ausgefhrt wird, wird der Rest des Schleifenkrpers bersprungen. Die Syntax ist wiederum sehr einfach:

```
continue;
```

Betrachten Sie dazu das folgende Stck Code, das von 1 bis 5 zhlt:

```
i = 0;
while ( i < 5 )
{   i = i + 1;
    if ( i == 3 )
        continue;
    lprint("%d", i);    # drucken
}
```

Wenn hier ’i’ den Wert 3 hat, wird ’lprint’ nicht ausgefhrt, sondern sofort wieder ’i < 5’ abgefragt, um festzustellen, ob die Schleife schon beendet werden mu. Daher sieht die Ausgabe so aus:

```
1
2
4
5
```

1.43 der unbedingte sprung: ’goto’

Der unbedingte Sprung: ’goto’ SCELREF.HYP
Der Sprungbefehl ’goto’ ist in SCEleton mchtiger (und - das mu man dazusagen - potentiell gefhrlicher) als in anderen Sprachen. Er hat die Form:

```
goto Ausdruck;
```

Dabei ist ’Ausdruck’ ein Ausdruck, der als Wert eine Zeichenkette ergibt, die identisch mit dem Namen einer im selben Makro definierten Sprungmarke sein mu. Andernfalls wird die aktuelle Sequenz mit einem Laufzeitfehler abgebrochen. Die Marke braucht also nicht fest vorgegeben zu werden, sondern kann im Makro berechnet werden. Das hat auch zur Folge, da eine konstant vorgegebene Marke als Zeichenkettenkonstante, d.h. in Apostrophen oder Anfhrungszeichen

codiert werden mu. Da zuerst der Ausdruck ausgewertet und
anschlieend der Makro nach der passenden Marke durchsucht werden
mu, ist dieser Befehl nicht so schnell, wie man erwarten knnte. Das
ist vielleicht ein weiteres Argument, vorsichtig damit umzugehen. Im
folgenden Beispiel sollte 's' den Wert 1 oder 2 haben:

```
        goto "Label" // s;
    Label1.
        lprint("Label 1");
        return 0;
    Label2.
        lprint("Label 2");
        return 0;
```

1.44 das ende eines makros: 'return'

Das Ende eines Makros: 'return' SCELREF.HYP
Die 'return'-Anweisung beendet einen Makro und legt gleichzeitig
seinen Rckgabewert fest. Wenn diese Anweisung fehlt, endet der Makro
dann, wenn seine letzte Anweisung ausgefhrt wurde. Der Rckgabewert
ist in diesem Falle immer vom Typ Zahl und hat den Wert 0. Die Syntax
der 'return'-Anweisung lautet:

```
    return Ausdruck;
```

'Ausdruck' steht fr einen Ausdruck, dessen Ergebnis von beliebigem
Typ sein kann. Dieses Ergebnis wird vom Makro als Rckgabewert
benutzt. Wird kein bestimmter Rckgabewert verlangt, so knnen Sie
einfach 'return 0;' schreiben (siehe auch das Beispiel zu
'goto').

1.45 alleinstehende ausdrcke

Alleinstehende Ausdrcke SCELREF.HYP
In SCEleton bildet - hnlich wie in C - ein Ausdruck fr sich genommen
schon eine Anweisung. Nach dieser Regel ist

```
  1 + 1;
```

ein syntaktisch korrekter Befehl, der allerdings keinerlei praktischen
Nutzen hat, da das Ergebnis der Rechnung nirgends gespeichert oder
ausgegeben wird. (Allerdings warnt SCEleton auch nicht vor solchen
wirkungslosen Befehlen.) Es gibt aber zwei Flle, in denen solche
Konstruktionen eminent ntzlich sind. Das sind die Prozeduraufrufe
und die Ausdrcke mit Zuweisungsoperatoren.

Prozeduraufrufe
Zuweisungsoperatoren

1.46 prozeduraufrufe

Prozeduraufrufe SCELREF.HYP
Ein Makro oder eine Laufzeitfunktion wird dann als Prozedur behandelt, wenn er aufgerufen wird, ohne seinen Rckgabewert auszuwerten. Ein solcher Aufruf ist natrlich nur dann sinnvoll, wenn der Makro etwas anderes tut, als einen Rckgabewert zu berechnen, wenn er etwa Daten ausgibt oder Ergebnisse in globalen Variablen ablegt. Ein solcher Aufruf (der nichts anderes als ein alleinstehender Ausdruck ist) sieht folgendermaen aus:

```
name(Ausdruck1, Ausdruck2);
```

'Ausdruck1' und 'Ausdruck2' sind hier die Ausdrcke, deren Wert dem Makro als Parameter bergeben wird. Je nachdem, wie der Makro definiert ist, knnen hier natrlich auch keine, weniger oder mehr Parameter stehen. Bei parameterlosen Makros sieht der Aufruf also so aus:

```
name();
```

1.47 zuweisungsoperatoren

Zuweisungsoperatoren SCELREF.HYP
Wir haben oben bei der Beschreibung der Operatoren einige unterschlagen, nmlich die Zuweisungsoperatoren. Dabei handelt es sich um Abkrzungen fr die Zuweisungsanweisungen 'set', 'insert', 'delete' und 'replace'. Die Operatorschreibweise fr diese Vorgnge hat neben der Krze noch andere Vorteile. Die Operatoren sehen aus wie folgt:

```
'='     entspricht      'set'
'|-:'   entspricht      'replace'
'<-:'   entspricht      'insert'
```

Links von diesen Operatoren steht jeweils das zu verndernde Datenelement (bzw. das Element, in dem etwas ersetzt oder eingefgt wird), rechts steht der Ausdruck, der die neue Gre ergibt. Daher hoffen wir, da die Operatoren fr 'insert' und 'replace' einen gewissen mnemonischen Wert haben. Beispiele:

```
a = 3;              # weist a den Wert 3 zu

b = 'Dies ist ein Test';
b[0:4] |-: 'Das'    # ersetzt 'Dies' durch 'Das'

b = 'Ein Test';
b[3] <-: ' kleiner'; # fgt ' kleiner' hinter 'Ein' ein
```

Beim Ersetzen steht also links der zu ersetzende Teilbereich; beim Einfgen die Einfgeposition. Um den Lschbefehl 'delete' durch einen Operator zu beschreiben, erhlt das Nullsymbol '$$$' seinen Sinn. Um einen Teilbereich zu lschen, ersetzen Sie ihn durch das Nullsymbol. Beispiel:

```
b = 'Ein kleiner Test';
```

```
    b[4:8] |-: $$$;         # lscht 'kleiner '
```

Da es sich bei diesen Symbolen eben nicht um Anweisungen, sondern um Operatoren handelt, knnen sie auch hintereinandergesetzt werden. Das gestattet weitere Abkrzungen. Sie mssen nur beachten, da diese Operatoren die niedrigste Prioritt von allen haben und - im Unterschied zu allen anderen - von rechts nach links ausgewertet werden. Dadurch lassen sich etwa mehrere Variablen initialisieren, indem man einfach schreibt:

```
    a = b = c = 1;
```

hnliche Abkrzungen sind auch fr '|-:' und '<-:' mglich. Bei bertriebener Anwendung machen diese "Tricks" Ihre Programme aber leicht unbersichtlich. Seien Sie also vorsichtig.

1.48 die laufzeitumgebung

Die Laufzeitumgebung SCELREF.HYP

Die Laufzeitumgebung startet und steuert die Ausfhrung einer Makrosequenz. Darber hinaus stellt sie eine Reihe von vordefinierten Laufzeitfunktionen zur Verfgung, die auf der Ebene des 68K-Maschinencodes ausgefhrt werden. Diese beinhalten eine Anzahl von ntzlichen Operationen, insbesondere aus dem Bereich der Ein- und Ausgabe und der Zeichenverarbeitung. Im Gegensatz zu SCEleton-Makros knnen einige dieser Funktionen mit beliebig vielen Parametern aufgerufen werden.

Diese vordefinierten Laufzeitfunktionen zerfallen in zwei Klassen: In SCEleton-Kernfunktionen und in JAnE-spezifische Aufrufe. Der Grund fr diese Zweiteilung liegt darin, da das SCEleton-System so konzipiert ist, da es an beliebige Applikationen angebunden werden kann und mit diesen ber eine relativ schmale Schnittstelle kommuniziert. Im Stadium der Planung befindet sich eine Erweiterung, die es berflssig macht, da SCEleton Teil einer Applikation sein mu. Dies setzt aber ein stabiles Multitasking-Betriebssystem voraus, das entsprechende Mglichkeiten der Kommunikation zwischen parallelen Prozessen bietet. Dann wird SCEleton als eigener Hintergrundproze laufen und sich ber eine Art 'Fernaufrufe' die Fhigkeiten der laufenden Anwendungen zunutze machen.

Ist dieses System erst einmal realisiert, so kann ein kompatibles Anwendungsprogramm ein eigenes Paket von Laufzeitfunktionen zur Verfgung stellen. Ein solches Paket wird als 'Wrterbuch' bezeichnet. JAnE enthlt bereits ein umfangreiches Wrterbuch mit Routinen zur Bearbeitung von ASCII-Texten.

SCEleton-Kernfunktionen
JAnE-spezifische Funktionen

1.49 sceleton-kernfunktionen

```
SCEleton-Kernfunktionen                                    SCELREF.HYP
Die SCEleton-Kernfunktionen zerfallen in die folgenden Bereiche:

Die Standardfunktionen
Datum und Uhrzeit
Verzeichnisse und Dateinamen
GEMDOS-Dateien
Das Environment
GEM-Aufrufe
Kommunikation mit anderen Programmen
Laufzeitfehler
```

1.50 die standardfunktionen

```
Die Standardfunktionen                                     SCELREF.HYP
Dieser Komplex beinhaltet Aufrufe zur Datenaufbereitung und -ausgabe
sowie einige andere Funktionen, die zur 'Grundausstattung' gehren.

dump
print
lprint
string
list
match
subst
split
uppercase, lowercase
rand, srand
stop
call
encode
datatype
switchmouse
stripchars
```

1.51 dump

```
dump                                                       SCELREF.HYP
Die Funktion 'dump' gibt eine Variable unformatiert, d.h. auf die
einfachste mgliche Art und Weise aus. Die Ausgabe erfolgt unter JAnE
in das Fenster 'SCEleton'. Zahlen werden als Dezimalzahlen gedruckt,
Zeichenketten einfach ausgegeben, und Listen erscheinen als
Aufzhlungen ihrer Elemente zwischen eckigen Klammern '[]', durch
Leerzeichen getrennt. Das Nullsymbol schlielich wird als '$$$'
gedruckt. Jeder 'dump'-Aufruf gibt eine vollstndige Zeile aus, d.h.
er wird mit einem Zeilenvorschub abgeschlossen. Gerade wegen ihrer
Einfachheit und aufgrund der Tatsache, da man den Datentyp eines
Wertes sofort erkennen kann, ist diese Funktion besonders fr
eingestreute Testausgaben zur Fehlersuche ntzlich. Beispiel:

    dtest()
```

```
    {   a = 3;
        dump(a);
       a = 'Harald Becker';
        dump(a);
        a = list(1, 2, 'Test');
        dump(a);
    }
```

Dieser Makro erzeugt die Ausgabe

```
    3
    Harald Becker
    [1 2 Test]
```

1.52 print

print SCELREF.HYP

Die Funktion 'print' gibt ein oder mehrere Datenelemente aus. Die Ausgabe erfolgt unter JAnE in das Fenster 'SCEleton'. Es gibt zwei Aufrufvarianten. Die erste ist an die 'printf'-Funktionsfamilie der C-Standardbibliothek angelehnt und funktioniert folgendermaen:

```
    print(format, daten1, ...);
```

Dabei ist 'format' eine Zeichenkette, die angibt, wie die Datenelemente dargestellt werden sollen. Die Datenelemente mssen vom Typ Zeichenkette oder Zahl sein. Die Punkte '...' sollen andeuten, da hier beliebig viele Datenelemente stehen knnen. Der Parameter 'format' bildet eine Art Maske, d.h. einen Text, an den die Datenelemente (bzw. ihre Textentsprechung) an bestimmten Stellen eingesetzt werden. Eine solche Stelle wird in 'format' durch eine sogenannte 'Konvertierung' markiert. Dabei mssen soviele Konvertierungen im Formatstring stehen, wie Datenelemente bergeben werden. Eine Konvertierung hat folgende Gestalt:

```
    %[Flags][Breite][Typ]
```

Der Typ wird durch ein Zeichen codiert und mu immer angegeben werden. Dabei gibt es folgende Mglichkeiten:

```
    'd'     Das entsprechende Datenelement ist eine Zahl, die als
            vorzeichenbehaftete Dezimalzahl ausgegeben wird.
    'x'     Das entsprechende Datenelement ist eine Zahl, die als
            vorzeichenlose Hexadezimalzahl ausgegeben wird.
    'b'     Das entsprechende Datenelement ist eine Zahl, die als
            vorzeichenlose Binrzahl ausgegeben wird.
    's'     Das entsprechende Datenelement ist eine Zeichenkette
    'c'     Das entsprechende Datenelement ist eine Zahl, deren untere
            8 Bit als ASCII-Zeichen ausgegeben werden.
```

Die Breite ist entweder eine Zahl oder zwei Zahlen, getrennt durch einen Punkt '.'. Die erste (oder einzige) Zahl gibt die Mindestbreite der Ausgabe in Zeichen an. Wenn das Datenelement weniger Zeichen erzeugt, wird die Ausgabe mit Leerzeichen aufgefllt. Wenn das Datenelement mehr Zeichen erzeugt, werden einfach mehr Zeichen

ausgegeben. Die zweite Zahl gibt fr das 's'-Format die Hchstbreite an, nach deren Erreichen die Zeichenreihe abgeschnitten wird. Bei den 'd', 'x', 'b' und 'c'-Formaten hat sie keine Wirkung. Anstelle beider Zahlen drfen auch Sternchen '*' stehen. Wenn dies der Fall ist, mu die zu verwendende Zahl als Parameter bergeben werden, und zwar direkt vor dem auszugebenden Datenelement. Damit knnen Sie leicht die Werte berechnen lassen, ohne die Formatzeichenreihe manipulieren zu mssen. Die Breitenangabe kann auch ganz fehlen. Dann werden immer genau die ntigen Zeichen ausgegeben.

Die Flags sind eine Anzahl von Zeichen, die die Ausgabe nher beeinflussen. Es gibt folgende Mglichkeiten:

'-'	Erzwingt linksbndige Ausgabe, d.h. wenn die vorgegebene Breite grer ist als die bentigte Zeichenzahl, wird das Feld von rechts mit Leerzeichen aufgefllt.
'+'	Wenn dieses Flag angegeben wird, erscheint bei positiven Dezimalzahlen ('d'-Format) das Vorzeichen '+'. Andernfalls erscheint nur bei negativen Zahlen ein Vorzeichen.

Einige Beispiele: Der Makro

```
ptest()
{   b = 'HaraldBecker';
    lprint("|%5.7s|", b);
    lprint("|%16.7s|", b);
    lprint("|%*.*s|", 16, 7, b);
    b = 12345;
    lprint("|%9d|", b);
    lprint("|%-9d|", b);
    lprint("|%-+9d|", b);   # Es knnen mehrere
                            # Flags verwendet werden
}
```

erzeugt die Ausgabe

```
|HaraldB|
|         HaraldB|
|         HaraldB|
|    12345|
|12345    |
|+12345   |
```

(Hier wird statt 'print' die - was Formatierung und Parameter betrifft - identische Funktion 'lprint' verwendet, um die Ausgabe jedes Befehls in eine eigene Zeile zu stellen. Zwei aufeinanderfolgende 'print'-Aufrufe wrden in dieselbe Zeile ausgeben.)

Die zweite Syntaxvariante der 'print'-Funktion dient zur Ausgabe von Listen. Sie sieht so aus:

```
print(liste, padding);
```

Dabei ist 'liste' die auszugebende Liste; 'padding' ist eine Zeichenreihe, deren erstes Zeichen als Trennzeichen zwischen den

```
Listenelemeneten benutzt wird. Weitere Zeichen werden ignoriert. Die
Listenelemente werden wie bei 'dump' auf die einfachste Art
ausgegeben. Wenn als Listenelement eine Liste auftaucht, wird diese
wieder im selben Format ausgegeben. Beispiel:

    a = list(1, 2, 3);
    print(a, ';');

Diese Zeilen erzeugen die Ausgabe

    1;2;3
```

1.53 lprint

```
lprint                                                        SCELREF.HYP
Diese Funktion ist identisch mit 'print', bis auf die Tatsache, da
'lprint' immer eine vollstndige Zeile ausgibt, d.h. nach der Ausgabe
einen Zeilenvorschub erzeugt. Diese Aufteilung ist darin begrndet,
da unter JAnE ein Zeilenvorschub nicht durch ein bestimmtes Zeichen
dargestellt wird, sondern durch einen Steuerbefehl erzwungen werden
mu. (Beispiel siehe unter 'print')
```

1.54 string

```
string                                                        SCELREF.HYP
Auch diese Funktion ist identisch mit 'print', allerdings gibt sie gar
nichts aus. Stattdessen kopiert sie die erzeugten Zeichen in eine
Zeichenreihe, die sie als Returnwert zurckgibt. Beispiel:

    a = 123;
    b = 456;
    c = string("%d %d", a, b);
    dump('Start: '// c // '|');

Diese Zeilen erzeugen die Ausgabe:

    Start: 123 456|

Die Zusammenfgung innerhalb der 'dump'-Anweisung soll nur deutlich
machen, da das Ergebnis von 'string' eine Variable wie jede andere
ist.
```

1.55 list

```
list                                                          SCELREF.HYP
Die Funktion 'list' erwartet beliebig viele Argumente und fgt diese
zu einer Liste zusammen, die den Rckgabewert bildet. Beispiele:

    a = list(1, 2);
    dump(a);            # Ausgabe: [1 2]
```

```
    a = list(1, 2, 3);
    dump(a);            # Ausgabe: [1 2 3]

    a = list(1, 2);
    a = list(a, 3);
    dump(a);            # Ausgabe: [[1 2] 3]
```

Das erste Beispiel ist quivalent mit ’1 ++ 2’. Das zweite Beispiel soll zeigen, da auch mehr Elemente mglich sind. Das dritte Beispiel ist nicht quivalent mit ’a ++ 3’. Dieser Ausdruck htte die dreielementige Liste ’[1 2 3]’ erzeugt, whrend der obige ’list’-Aufruf eine zwei-elementige Liste erzeugt, deren erstes Element wieder eine zweielementige Liste ist, also eine etwas komplexere Struktur. Die Regel lautet also: Jeder Parameter von ’list’ wird zu einem Element der Ergebnisliste, unabhngig von seiner Gestalt.

1.56 match

match SCELREF.HYP

’match’ ist eine Textsuchfunktion, eine etwas komplexere Version des Operators ’?’ fr Zeichenketten. ’match’ durchsucht allerdings keine Listen. ’match’ hat zwei Parameter, die beide in Zeichenreihen wandelbar sein mssen. Die erste Zeichenreihe wird durchsucht, die zweite Zeichenreihe gibt den zu suchenden Text an. Dieser wird jedoch - im Gegensatz zum ’?’-Operator - nicht als Konstante interpretiert, sondern als regulrer Ausdruck. Zu den Mglichkeiten eines solchen Ausdrucks vergleiche das Kapitel ber regulre Adrcke in der Dokumentation zu JAnE. Ein regulrer Ausdruck darf maximal 255 Zeichen umfassen.

Die Funktion ’match’ gibt eine Liste zurck, die zwei Zahlen enthlt. Die erste gibt die Startposition der Fundstelle an, die zweite die Lnge. Dies ist erforderlich, da im Gegensatz zur Suche nach Konstanten bei regulren Ausdrcken die Lnge der Fundstelle variabel sein kann. Falls die Suche nicht erfolgreich ist, gibt ’match’ eine leere Liste zurck. Dies kann etwa geprft werden, indem man die Lnge der Liste mit dem ’@’-Operator auf 0 testet. Beispiel:

```
    l = match(’aabbbcccde’, ’b+c+’);
    lprint(’Start: %d, Lnge: %d’, l[0], l[1]);
    l = match(’aabbbcccde’, ’b+c+x’);
    dump(l);
```

Diese Zeilen erzeugen die Ausgabe:

```
    Start: 2, Lnge: 6
    []
```

Die zweite Ausgabezeile zeigt eine leere Liste an.

1.57 subst

subst SCELREF.HYP
Die Steigerung von 'match' ist die Funktion 'subst', die einen Such- und Ersetzvorgang mit regulren Ausdrcken in einer Zeichenreihe ausfhrt. Diese Funktion erwartet vier Parameter:

- Die zu durchsuchende Zeichenreihe
- Den regulren Suchausdruck
- Die Zeichenreihe, durch die jedes Auftreten des Suchausdrucks ersetzt werden soll.
- Einen Wahrheitswert, der angibt, ob nur die erste oder alle Fundstellen ersetzt werden sollen.

Die Funktion hnelt im Kleinen den Mglichkeiten im Such- und Ersetzdialog von JAnE. Es kann allerdings nicht im Sinne von JAnE 'komplex' ersetzt werden, insbesondere wird das Arbeiten mit Registern nicht untersttzt. Der Rckgabewert der Funktion ist die modifizierte Zeichenreihe (oder auch nicht modifiziert, falls die Suche nicht erfolgreich war). Beispiel:

```
l = subst('aabbccdbce', 'b+c+', 'hhh', 0);
dump(l);
l = subst('aabbccdbce', 'b+c+', 'hhh', 1);
dump(l);
l = subst('aabbccdbce', 'b+c+x', 'hhh', 1);
dump(l);
```

Diese Zeilen erzeugen die Ausgabe:

```
aahhhdbce
aahhhdhhhe
aabbccdbde
```

Im ersten Aufruf wurde nur einmal ersetzt, im zweiten zweimal (alle Fundstellen), im dritten gar nicht (nichts gefunden).

1.58 split

split SCELREF.HYP
Die 'split'-Funktion ist in gewisser Weise das Gegenstck zu 'string'; sie dient dazu, eine Zeichenkette nach bestimmten Regeln in Teilstcke zu zerlegen. Diese Teilstcke sollen hier 'Worte' genannt werden. 'split' erwartet zwei Parameter. Der erste ist die zu zerlegende Zeichenkette, der zweite ist eine Zeichenkette, die angibt, durch welche Zeichen die Worte getrennt werden. Dabei werden folgende Unterscheidungen getroffen:

1. Die 'Trennzeichenkette' enthlt genau ein Zeichen
   1.a Dieses ist ein Leerzeichen. In diesem Falle werden die Worte durch ein oder mehrere Leerzeichen und/oder Tabulatoren ('weie Zeichen') getrennt.
   1.b Dieses ist kein Leerzeichen. In diesem Falle werden die Worte

```
       durch genau ein solches Zeichen getrennt. Zwei
       aufeinanderfolgende Trennzeichen schlieen ein leeres Wort ein,
       das von 'split' durch das Nullsymbol reprsentiert wird.
2. Die 'Trennzeichenkette' enthlt mehr als ein Zeichen
   In diesem Fall wird die Trennzeichenkette als regulrer Ausdruck
   interpretiert, und die Worte werden durch Zeichenfolgen getrennt,
   die auf diesen Ausdruck passen.
```

Dieser Algorithmus ist an die Sprache 'awk' angelehnt, dort findet er aber implizit in der Hauptschleife eines Programmes statt. Beispiele:

```
    l = split('Dies ist  ein Test', ' ');
    dump(l);
    l = split('Dies,ist,,ein,Test', ',');
    dump(l);
    l = split('Dies,ist,;ein.,Test', '[,;.]');
    dump(l);
    l = split('Dies,ist,;ein.,Test', '[,;.]+');
    dump(l);
```

Diese Zeilen erzeugen die Ausgabe:

```
    [Dies ist ein Test]
    [Dies ist $$$ ein Test]
    [Dies ist $$$ ein $$$ Test]
    [Dies ist ein Test]
```

Im ersten Beispiel werden Leerzeichen als Trenner benutzt (beachten Sie, da die beiden Leerzeichen in der Mitte der Zeichenkette als ein Trenner angesehen werden). Im zweiten Beispiel sind es die Kommata (hier wird zwischen den beiden Kommata ein leeres Feld gesehen). Im dritten und vierten Beispiel werden mittels des regulren Ausdrucks Kommas, Punkte und/oder Semikolons als Trennzeichen angegeben. Durch die Formulierung des regulren Ausdrucks haben Sie es in der Hand, ob mehrere Trennzeichen leere Felder einschlieen oder als ein zusammenhngender Trenner interpretiert werden. Hieraus erklrt sich der Unterschied zwischen dem dritten und dem vierten Beispiel.

1.59 uppercase, lowercase

uppercase, lowercase SCELREF.HYP

Diese Funktionen wandeln alle Buchstaben einer Zeichenkette in Gro- bzw. Kleinbuchstaben. Die Syntax lautet:

```
    l = uppercase(string);
    l = lowercase(string);
```

Der Rckgabewert 'l' enthlt jeweils die konvertierte Zeichenkette.

1.60 rand, srand

rand, srand SCELREF.HYP
Diese beiden Funktionen werden gemeinsam behandelt, da sie eine Aufgabenstellung lsen. Es handelt sich dabei um einen Pseudo-Zufallszahlengenerator. 'srand' initialisiert den Generator mit einem Startwert, und 'rand' liefert immer den jeweils nchsten Wert in der Zufallsfolge zurck. Dabei knnen Sie fr 'rand' einen Skalierungsfaktor angeben, der die Ergebnisse in einen bestimmten Zahlenbereich transformiert. Wenn dieser Faktor n ist, liegen die Ergebnisse von 'rand' zwischen 0 und n - 1.

Die Rede ist von Pseudo-Zuflligkeit, weil von einem festen Startwert ausgehend immer dieselbe Folge geliefert wird. Das bedeutet, da sich bei Kenntnis des Startwertes Ablufe, die von diesem Generator gesteuert werden, doch reproduzieren lassen, also nicht mehr zufllig sind. Unten wird ein Trick beschrieben, der SCEleton dazu veranlat, von sich aus einen ziemlich guten Startwert zu whlen, der dann aber unbekannt bleibt. Der folgende Beispiel-Makro gibt zehn Zufallszahlen zwischen 0 und 99 aus:

```
random()
{   srand(-1);              # inititalisieren
    i = 0;
    while ( i < 10 )
    {   dump(rand(100));
        i = i + 1;
    }
}
```

(Hier knnen wir natrlich keinen Beispieloutput angeben.) 'srand' erhlt in diesem Beispiel einen negativen Parameter, was dazu fhrt, da der Betrag dieses Parameters ignoriert wird. Stattdessen werden als Startwert die unteren 8 Bits der Systemvariablen '_vbclock' verwendet. Dabei handelt es sich um einen Zhler, der abhngig von der Wiederholfrequenz des angeschlossenen Bildschirms 50 - 70 mal pro Sekunde erhht wird. Damit kann man den Inhalt der unteren Bits dieser Variablen, die sich am schnellsten verndern, als hinreichend zufllig ansehen, wenn 'srand' nur einmal pro Makro aufgerufen wird. Ein positiver Parameter von 'srand' wrde direkt als Startwert eingesetzt.

1.61 stop

stop SCELREF.HYP
Der 'stop'-Aufruf beendet die laufende Sequenz und gibt eine Abbruch-Meldung aus. Die Syntax ist denkbar einfach:

```
stop();
```

1.62 call

call SCELREF.HYP

Diese (und die folgende) Funktion ermglichen einige 'faule Tricks' im Zusammenhang mit dem Aufruf von Makros. 'call' ruft einen Makro indirekt auf. Dazu erwartet 'call' eine Zeichenkette als ersten Parameter, die den Namen des Makros ergeben mu. Falls der Name dem SCEleton-System nicht bekannt ist, wird die Sequenz mit einem Laufzeitfehler abgebrochen. Die weiteren Parameter werden von 'call' an den aufgerufenen Makro weitergereicht. Der Returnwert von 'call' ist der des aufgerufenen Makros. Beispiel:

```
main()
{   name = 'addiere';
    r = call(name, 2, 3);
    dump(r + 1);
}

addiere(a, b)
{   return a + b;
}
```

Der Ausgabewert dieses Beispiels ist 6. Dieses Feature wird natrlich nur dann sinnvoll, wenn man den Namen zur Laufzeit aus festen und variablen Bestandteilen (z.B. Benutzereingaben) zusammensetzt. Auf diese Art liee sich ziemlich einfach eine Menusteuerung oder ein Kommandoprozessor aufbauen.

1.63 encode

encode SCELREF.HYP

Diese Funktion erffnet noch weitergehende Mglichkeiten; sie gestattet es nmlich, dynamisch einen Makro zu erzeugen, indem aus einem laufenden Makro heraus der bersetzer aufgerufen wird. 'encode' erwartet als einzigen Parameter eine Liste aus Zeichenreihen, die den Quelltext des neuen Makros enthalten. Dieser Quelltext mu (selbstverstndlich) auch einen Kopf enthalten, der den Namen des Makros festlegt. Unter diesem Namen kann der Makro nach fehlerfreier bersetzung mittels 'call' aufgerufen werden. Falls der Quelltext fehlerhaft ist, wird die Sequenz mit einem Laufzeitfehler abgebrochen. Der neu erzeugte Makro existiert nur so lange, wie die aktuelle Makrosequenz luft. Beispiel:

```
main()
{   source = list('addiere(a, b)',
                  '{  return a + b;',
                  '}');
    encode(source);
    b = call('addiere', 2, 3);
    dump(b + 1);
}
```

Dies ist eine noch kompliziertere Version des Beispiels zu 'call'. Der Ausgabewert ist hier ebenfalls 6. Die erste Zeile weist 'source' den Quelltext des Makros 'addiere' zu. Jedes Listenelement umfat eine Zeile. Das mu aber nicht so sein, da SCEleton bekanntlich Quelltexte in freier Form akzeptiert. Anschlieend wird dieser Makro mit 'encode' erzeugt und mit 'call' ausgefhrt. Nachdem er einmal erzeugt

ist, kann er auch mehrmals mit 'call' aufgerufen werden, solange die aktuelle Sequenz noch luft.

1.64 datatype

datatype SCELREF.HYP
Mittels 'datatype(x)' knnen Sie den Datentyp eines Ausdruckes 'x' ermitteln. Die Funktion liefert eine Zeichenkette zurck, die den Datentyp des bergebenen Ausdrucks anzeigt. Folgende Zeichenketten und Bedeutungen sind als Rckgabe mglich:

```
    'null'      fr das Nullsymbol '$$$'
    'int'       fr eine Zahl
    'string'    fr eine Zeichenkette
    'list'      fr eine Liste von Werten
```

1.65 switchmouse

switchmouse SCELREF.HYP
Syntax:

```
    switchmouse(onoff);
```

Dient zum Ein- oder Ausschalten des Mauszeigers. 'onoff' ist ein Zahlenwert, der die Funktion bestimmt. Mit '0' wird die Maus aus-, mit jedem anderen Wert eingeschaltet.

1.66 stripchars

stripchars SCELREF.HYP
Diese Funktion entfernt Zeichen aus einer Zeichenkette. Ein Aufruf sieht so aus:

```
    result = stripchars(s, class);
```

Dabei ist 's' die Ausgangszeichenkette, whrend 'class' angibt, welche Zeichen entfernt werden sollen. 'class' mu dazu eine Zeichenklasse im Sinne der regulren Ausdrcke enthalten, d. h. die Menge der zu entfernenden Zeichen, eingefat in eckige Klammern. Das folgende Beispiel entfernt aus 's' alle Ziffern:

```
    result = stripchars(s, '[0-9]');
```

1.67 datum und uhrzeit

Datum und Uhrzeit SCELREF.HYP
SCEleton stellt Ihnen die folgenden Aufrufe zur Verfgung, die Datum
und Uhrzeit aus der Systemuhr Ihres Rechners auslesen und sie in
(fast) beliebigen Formaten lesbar machen knnen:

gettime
formattime

1.68 gettime

gettime SCELREF.HYP
Diese Funktion liest die Systemuhr des Rechners aus und stellt das
Ergebnis in einem Format zur Verfgung, das von 'formattime' (s.u.)
akzeptiert wird. Der Aufruf lautet:

```
t = gettime();
```

1.69 formattime

formattime SCELREF.HYP
Dieser Aufruf wandelt einen Zeitwert, der von 'gettime' geliefert
wurde, in eine lesbare Zeichenkette um. Die Syntax sieht so aus:

```
result = formattime(t, format);
```

Dabei ist 't' der Zeitwert, 'format' ist eine Formatzeichenkette, die
angibt, wie das Ergebnis aussehen soll. hnlich wie das Format bei
'print' besteht 'format' aus Zeichen, die bernommen werden, und
Platzhaltern. Es sind folgende Platzhalter mglich:

'%a'	abgekrzter Wochentag
'%A'	ausgeschriebener Wochentag
'%b'	abgekrzter Monat
'%B'	ausgeschriebener Monat
'%c'	Datum und Uhrzeit
'%d'	Tag im Monat (1-31)
'%H'	Stunde (0-23)
'%I'	Stunde (0-12)
'%j'	Tag im Jahr (1-366)
'%m'	Monat (1-12)
'%M'	Minute (00-59)
'%p'	AM/PM
'%S'	Sekunde (00-59)
'%w'	Wochentag (0-6)
'%W'	Woche im Jahr (0-52)
'%x'	lokale Datumsdarstellung
'%X'	lokale Zeitdarstellung
'%y'	Jahr ohne Jahrhundert (0-99)
'%Y'	Jahr mit Jahrhundert
'%Z'	Name der Zeitzone (MEZ)
'%%'	das Zeichen '%'

```
Ein Beispiel zu 'gettime' und 'formattime':

  t = gettime();
  dump(formattime(t, "%d. %B %Y, %H:%M:%S, Zeitzone: %Z"));
  dump(formattime(t, "%x %X"));

liefert (bzw. lieferte) die Ausgabe:
    19. November 1994, 19:39:00, Zeitzone: MEZ
    19/11/94 19:39:00
```

1.70 verzeichnisse und dateinamen

```
Verzeichnisse und Dateinamen                                SCELREF.HYP
Dieser Komplex enthlt eineige Funktionen zur Arbeit mit Datei- und
Ordnernamen sowie dem aktuellen GEMDOS-Pfad:

cd
cwd
splitname
makename
fullpath
```

1.71 cd

```
cd                                                          SCELREF.HYP
Syntax:

    cd(path);

Diese Funktion setzt das aktuelle Arbeitsverzeichnis auf 'path'. In
diesem Verzeichnis sucht GEMDOS alle Dateien, die nicht mit
vollstndigen Pfaden angegeben wurden. Dieses Verzeichnis wird z.B.
auch in der Dateiauswahlbox angezeigt, wenn 'fselect' mit einer
leeren Verzeichnisangabe aufgerufen wurde. Unter JAnE empfiehlt es
sich, das Arbeitsverzeichnis mit 'cwd' zu speichern und zum Ende
der Makrosequenz wieder zu restaurieren. Beispiel:

fsel()
{   sd = cwd();
    cd('c:\');
    s = fileselect("", "", "Dateiauswahl");
    if ( s == "" )
        lprint("Keine Auswahl");
    else
        lprint("Auswahl: %s.", s);
    cd(sd);
}
```

1.72 cwd

```
cwd                                                          SCELREF.HYP
Syntax:

    s = cwd();

Diese Funktion erwartet keine Parameter und liefert das aktuelle
Arbeitsverzeichnis zurck.
```

1.73 splitname

```
splitname                                                    SCELREF.HYP
Syntax:

    l = splitname(string);

Diese Funktion zerlegt einen Dateinamen in drei Bestandteile: Pfad,
Name und Erweiterung. Diese drei Teile werden in einer Liste mit drei
Elementen zurckgegeben. Fehlt einer der Bestandteile, so enthlt das
entsprechende Listenelement eine leere Zeichenkette. Beispiel: der
Makro

splitnames()
{   # Liste von Beispielnamen aufbauen
    tl = list(  'F:\test\testdata.dat',
                'F:\testdata.dat',
                'testdata.dat',
                'F:testdata.dat',
                'testdata',
                'F:\test\testdata',
                '.dat');

    # Jetzt werden die Beispielnamen der Reihe nach
    # zerlegt und ausgegeben
    i = 0;
    while ( i < @tl )
    {   list = splitname(tl[i]);
        lprint('|%10s|%8s|%3s|', list[0], list[1], list[2]);
        i = i + 1;
    }
}

erzeugt die Ausgabe

|   F:\test|testdata|dat|
|       F:\|testdata|dat|
|          |testdata|dat|
|        F:|testdata|dat|
|          |testdata|   |
|   F:\test|testdata|   |
|          |        |dat|
```

1.74 makename

```
makename                                                    SCELREF.HYP
Syntax:

    s = makename(path, name, ext);

Diese Funktion kehrt die vorherige um, d.h. sie setzt die drei
Bestandteile wieder zu einem gltigen Namen zusammen. Dabei fgt sie
Backslash und Punkt selbstndig ein.

Beispiel: Der Makro

buildname()
{   lprint('%s', makename('F:\test', 'testdata', dat');
}

erzeugt die Ausgabe:

F:\test\testdata.dat
```

1.75 fullpath

```
fullpath                                                    SCELREF.HYP
Syntax:

    s = fullpath(string);

Diese Funktion vervollstndigt einen bergebenen Dateinamen, d.h. sie
fgt eventuell fehlende Ordner sowie einen einen Laufwerksbuchstaben
gem dem aktuellen GEMDOS-Pfad hinzu. Der Rckgabewert ist also
immer ein vollstndig qualifizierter Dateiname.

Beispiel: Der Makro

fullnames()
{   lprint('%s', fullpath('testdata.dat'));
    lprint('%s', fullpath('\testdata.dat'));
    lprint('%s', fullpath('F:testdata.dat'));
    lprint('%s', fullpath('F:\test\testdata.dat'));
}

knnte, je nach den vorhandenen Verzeichnissen auf Ihrem Rechner,
folgende Ausgaben erzeugen:

F:\JANE\testdata.dat
F:\testdata.dat
F:\JANE\testdata.dat
F:\test\testdata.dat
```

1.76 gemdos-dateien

GEMDOS-Dateien SCELREF.HYP
SCEleton bietet eine Reihe von Laufzeitfunktionen an, die den Aufrufen des GEMDOS zur Dateibehandlung entsprechen. Damit ist es mglich, von einem SCEleton-Makro aus Dateien auf Disketten oder Festplatten zu bearbeiten, ohne den Umweg ber einen JAnE-Puffer zu nehmen. Die Schreibweise der Dateinamen orientiert sich an den Standarddefinitionen von ATARI fr GEMDOS-Aufrufe. Es sind jedoch der Einfachheit halber einige Features von GEMDOS ausgespart.

Fcreate
Fopen
Fread
Freadline
Fwrite
Fseek
Fclose
Fexist
Fdelete
Frename
Fsfirst
Fsnext

1.77 fcreate

Fcreate SCELREF.HYP
Diese Funktion legt eine neue Datei an oder leert eine bestehende (d.h. stutzt sie auf eine Lnge von 0 Bytes zurck). Die Aufrufsyntax lautet:

```
    handle = Fcreate(name);
```

Dabei ist 'name' ein Ausdruck, der den Dateinamen ergibt, der unverndert an GEMDOS weitergeleitet wird. Der Rckgabewert 'handle' ist eine positive Kennzahl, die bei jeder nachfolgenden Operation auf der Datei angegeben werden mu. Im Falle eines Mierfolgs wird anstelle dieser Kennzahl ein negativer Fehlercode zurckgegeben. In SCEleton knnen im Gegensatz zum GEMDOS-Aufruf keine Attribute fr die neue Datei angegeben werden. Insbesondere sieht SCEleton es nicht als seine Aufgabe an, mit solchen Spezialitten wie Diskettenlabels umzugehen.

1.78 fopen

Fopen SCELREF.HYP
Dieser Aufruf ffnet eine bestehende Datei zum Lesen oder Schreiben. Die Aufrufsyntax lautet:

```
    handle = Fopen(name);
```

Auch 'Fopen' erwartet einen Dateinamen als Parameter und gibt einen positiven Handle oder einen negativen Fehlercode zurck. Insbesondere

fhrt der Versuch, eine noch nicht existierende Datei zu ffnen, zu einem Fehler. In diesem Fall mssen Sie 'Fcreate' anwenden. Ferner sollten Sie 'Fcreate' benutzen, wenn Sie die Datei auschlielich als Ausgabedatei verwenden mchten. Andernfalls kann es - je nach dem Algorithmus Ihres Makros - passieren, da anschlieend Reste der alten Daten zwischen Ihren neuen Daten stehen bleiben. Solchen Fehlern schieben Sie mit 'Fcreate' einen Riegel vor. In SCEleton knnen Sie den GEMDOS-Parameter 'mode' nicht setzen. SCEleton ffnet eine Datei immer zum Lesen und Schreiben.

1.79 fread

Fread SCELREF.HYP
Diese Funktion liest Daten aus einer mit 'Fcreate' oder 'Fopen' geffneten Datei. Die Syntax lautet:

```
    result = Fread(handle, bytes);
```

Dabei ist 'handle' die von 'Fopen' oder 'Fcreate' gelieferte Kennzahl, 'bytes' ist die Anzahl Bytes, die gelesen werden sollen. Das Ergebnis von 'Fread' ist eine zweielementige Liste, deren erstes Element die gelesene Zeichenkette ist. Das zweite Element ist 1, falls das Dateiende schon erreicht wurde, sonst 0. (Sie erinnern sich: in SCEleton darf eine Zeichenreihe alle 256 mglichen Bytes enthalten.) Den gelesenen Text erhlt man also wie folgt:

```
    text = Fread(handle, count)[0];
```

GEMDOS verwaltet fr jede geffnete Datei intern einen sog. Dateizeiger, der immer die nchste Byteposition in der Datei angibt, von der gelesen oder an die geschrieben wird. 'Fread' liest immer ab der aktuellen Position des Dateizeigers und setzt ihn anschlieend hinter das zuletzt gelesene Byte, so da Sie mit mehreren aufeinanderfolgenden 'Fread'-Aufrufen aufeinanderfolgende Bereiche der Datei bekommen.

1.80 freadline

Freadline SCELREF.HYP
Diese Routine liest ebenfalls Daten aus einer offenen Datei; im Gegensatz zu 'Fread' liest sie allerdings eine Zeile, die mit den ASCII-Zeichen CRLF (hex '0D0A') oder LF (hex '0A') abgeschlossen sein mu und maximal 512 Zeichen lang ist. Nach 512 Zeichen wird das Lesen abgebrochen. Der einzige Parameter ist die Dateikennung; das Ergebnis sieht genauso aus wie bei 'Fread'. Die Syntax lautet also:

```
    string = Freadline(handle);
```

bzw.

```
    zeile = Freadline(handle)[0];
```

Die Zeilenendemarkierung wird zwar gelesen, ist aber nicht Teil der Zeile.

1.81 fwrite

Fwrite SCELREF.HYP
Diese Routine ist das Gegenstck zu 'Fread'; sie schreibt Daten in eine geffnete Datei. Hier lautet die Syntax:

```
wbytes = Fwrite(handle, bytes, buffer);
```

'handle' ist wieder die Dateikennung. 'buffer' ist eine Zeichenkette, die die zu schreibenden Bytes enthlt. 'bytes' ist die Anzahl der Zeichen. Diese darf krzer als die Lnge von 'buffer' sein, aber nicht lnger. Sollte dies passieren, wird sie auf die Lnge von 'buffer' zurckgestutzt. Auch wenn 'bytes' 0 oder negativ ist, wird die Lnge von 'buffer' eingesetzt. In 'wbytes' meldet 'Fwrite' die Anzahl von Bytes zurck, die fehlerfrei geschrieben werden konnten. Ist diese 0 oder kleiner, so ist ein TOS-Fehler aufgetreten. Die wahrscheinlichste Ursache dafr ist immer eine volle Platte. 'Fwrite' schreibt an die aktuelle Dateizeigerposition und setzt der Dateizeiger anschlieend weiter.

1.82 fseek

Fseek SCELREF.HYP
Diese Funktion verschiebt den Dateizeiger, so da bestimmte Bereiche innerhalb der Datei (z.B. bestimmte Datenstze) gezielt gelesen oder berschrieben werden knnen. Die Syntax lautet:

```
newptr = Fseek(offset, handle, start);
```

Dabei ist 'handle' - wie gehabt - die Dateikennung. 'offset' gibt an, um wieviele Bytes der Zeiger versetzt werden soll, und 'start' bezeichnet den Ausgangspunkt dieser Verschiebung. Dabei gelten folgende Codes:

```
'0'     =     Dateianfang
'1'     =     aktueller Dateizeiger
'2'     =     Dateiende
```

Der Rckgabewert 'newptr' enthlt die neue Zeigerposition (relativ zum Dateianfang) oder eine negative Fehlernummer. Ein Beispiel:

```
 filelength(handle)             # Bestimmt die Lnge
                                # einer offenen Datei
 {
  aktptr = Fseek(0, handle, 1); # akt. Zeiger sichern
  length = Fseek(0, handle, 2); # Dateiende suchen
  Fseek(aktptr, handle, 0);     # Zeiger auf die
                                # Ausgangsposition
  return length;
```

```
}
```

1.83 fclose

```
Fclose                                                          SCELREF.HYP
Dieser Aufruf schliet die Datei. Die Syntax lautet:

    Fclose(handle);

'handle' ist wieder die Dateikennung. Mit 'Fopen' oder 'Fcreate'
geffnete Dateien mssen grundstzlich nach Gebrauch mit 'Fclose'
geschlossen werden, da TOS nur eine begrenzte Anzahl von offenen
Dateien gleichzeitig verwalten kann.
```

1.84 fexist

```
Fexist                                                          SCELREF.HYP
Diese Funktion prft, ob die Datei unter dem bergebenen Namen
existiert. Der Rckgabewert ist 0, wenn dies nicht der Fall ist,
sonst ungleich 0. Die Syntax lautet:

    yesno = Fexist('C:\AUTOEXEC.BAT');
```

1.85 fdelete

```
Fdelete                                                         SCELREF.HYP
Diese Funktion lscht eine Datei. Sie soll nur fr nicht
geffnete Dateien aufgerufen werden und operiert daher auf dem
Dateinamen, nicht auf einem Handle. Die Syntax lautet:

    Fdelete('C:\ASSIGN.SYS');
```

1.86 frename

```
Frename                                                         SCELREF.HYP
Dieser Aufruf benennt eine Datei um. Auch er arbeitet ber Namen. Der
erste Parameter ist der ursprngliche Name; der zweite ist der Name,
den die Datei erhalten soll. Dieser Aufruf ist auch in der Lage, eine
Datei innerhalb eines Laufwerks zu verschieben. Die Syntax lautet:

    Frename('C:\ASSIGN.SYX', 'C:\ASSIGN.SYS');
```

1.87 fsfirst

Fsfirst SCELREF.HYP
'Fsfirst' und 'Fsnext' (s.u.) lesen das Inhaltsverzeichnis eines Laufwerkes oder eines Ordners. Mit Hilfe einer Suchmaske lassen sich nur bestimmte Dateien suchen. Ein Suchvorgang mu mit einem Aufruf von 'Fsfirst' eingeleitet werden; weitere passende Dateinen werden dann mit 'Fsnext' gesucht. Die Syntax von 'Fsfirst' sieht so aus:

```
l = Fsfirst('C:\GEMSYS\*.FNT');
```

(Dieser Aufruf wrde z.B. eine Suche nach allen GDOS-Zeichensatzdateien im Ordner GEMSYS einleiten.) Der Rckgabewert von 'Fsfirst' ist eine Liste, die Informationen ber die erste passende Datei enthlt. Falls keine passende Datei aufgefunden werden konnte, ist die zurckgegebene Liste leer. Ansonsten gilt:

```
'l[0]'  =  Name der Datei
'l[1]'  =  1, falls es sich um einen Ordner handelt, sonst 0
'l[2]'  =  Gre der Datei in Bytes
'l[3]'  =  Datum/Uhrzeit der Erzeugung bzw. des letzten Zugriffs.
           Datum/Uhrzeit liegen in einem Format vor, das mit
           'formattime' lesbar gemacht werden kann.
'l[4]'  =  Das GEMDOS-Attributbyte der Datei.
'l[5]'  =  Das MiNT-Moduswort der Datei (enthlt u. a. die
           Zugriffsrechte fr Multiuser-Systeme. Genaueres in der
           MiNT-Doku.)
```

1.88 fsnext

Fsnext SCELREF.HYP
Wie oben erlutert, werden mit 'Fsnext' nach einem 'Fsfirst' weitere passende Dateien gesucht. Ein Aufruf sieht so aus:

```
l = Fsnext();
```

Der Rckgabewert in 'l' ist genauso zu interpretieren wie bei 'Fsfirst'.

1.89 das environment

Das Environment SCELREF.HYP
Das SCEleton-Environment, von dem hier die Rede ist, ist nicht mit dem Environment zu verwechseln, das GEMDOS fr eine neues Programm anlegt. Das SCEleton-Environment dient einfach dazu, Daten ber die Lebensdauer einer Makrosequenz hinaus zu speichern.

Es besteht ebenfalls aus Zeichenketten der Form 'Name=Wert', wobei 'Name' die Zeichenkette benennt und 'Wert' ihr Inhalt ist. Als zustzlichen Service wird es beim Start mit einer Kopie des GEMDOS-Environments gefllt, das JAnE mit auf den Weg bekommt. Abgesehen davon sind beide Environments grundverschiedene Dinge. SCEleton kann maximal 64 Environment-Variable verwalten.

Mit dem Environment befassen sich die folgenden Aufrufe:

setenv
getenv
unsetenv

1.90 setenv

setenv SCELREF.HYP
'setenv' richtet eine neue SCEleton-Environmentvariable ein oder
ndert eine bestehende. Die beiden Parameter sind Zeichenketten, die
den Namen und den Wert der neuen Variablen angeben. Wenn der Name
schon existiert, erhlt die Variable den neuen Wert. Ansonsten wird
sie komplett neu angelegt. Der Rckgabewert ist 0, wenn ein Fehler
aufgetreten ist, sonst 1. Allein schon wegen der Begrenzung der Zahl
der Variablen sollten Sie diesen Wert testen. Beispiel:

```
    if ( setenv('VARIABLE', 'Wert') == 0 )
        lprint('Fehler');
```

Der Name sollte kein Gleichheitszeichen und kein Nullbyte, der Wert
kein Nullbyte enthalten

1.91 getenv

getenv SCELREF.HYP
Mit 'getenv' wird der Wert einer bestehenden Environment-Variablen
abgefragt. Der einzige Parameter ist eine Zeichenkette, die den Namen
der Variablen angibt. Zurckgegeben wird eine weitere Zeichenkette,
die den Wert angibt. Falls die gewnschte Variable nicht existiert,
ist der Rckgabewert leer.

1.92 unsetenv

unsetenv SCELREF.HYP
Diese Routine lscht eine Variable aus dem SCEleton-Environment. Der
einzige Parameter ist wiederum der Name. Der Rckgabewert ist 0, wenn
ein Fehler aufgetreten ist, sonst 1.

1.93 gem-aufrufe

GEM-Aufrufe SCELREF.HYP
Fr einfache Aufgaben im Bereich der Benutzerfhrung stellt SCEleton
einige wenige GEM-Routinen zur Verfgung:

message

query
fileselect
shell

1.94 message

message SCELREF.HYP
Die Funktion 'message' stellt eine Alertbox dar, die ausschlielich dazu bestimmt ist, dem Benutzer eine unbersehbare Mitteilung zukommen zu lassen. Dabei handelt es sich nicht um den GEM-Standard, sondern um eine erweiterte Version, die auch fliegen kann und ber Tastendrcke gesteuert wird. Die Box hat immer einen Button mit der Beschriftung 'Ok'. Als Icon enthlt sie immer das Rufzeichen. Der einzige Parameter von 'message' ist eine Zeichenreihe, die den Meldungstext enthlt. Dieser besteht aus einem Boxtitel und maximal zwei Textzeilen. Die einzelnen Bestandteile werden durch das Zeichen '|' unterteilt. Der Titel sollte das SCEleton-System und das verwendete Makropaket identifizieren. Beispiel:

```
message('SCEleton - Test|Erste Zeile|Zweite Zeile');
```

1.95 query

query SCELREF.HYP
'query' stellt ebenfalls eine Alertbox dar, die aber im Gegensatz zu 'message' dem Benutzer eine Frage stellt. Diese Box hat mindestens zwei frei whlbare Buttons und als Icon das Fragezeichen. 'query' erwartet als Parameter zwei Zeichenreihen, die den Meldungstext und die Button-Beschriftungen angeben. Der Meldungstext ist genau wie bei 'message' aufgebaut. Im zweiten Parameter werden die Buttons mit '|' getrennt. 'query' gibt die Nummer das ausgewhlten Buttons zurck, wobei 0 fr den linken, 1 fr den mittleren und 2 fr den rechten Button (so vorhanden) steht. Beispiel:

```
button = query( 'SCEleton - Test|Was soll's denn sein?',
                'Bier|Schnaps|Wein');
```

Anschlieend enthlt 'button' 0 fr Bier, 1 fr Schnaps, oder 2 fr Wein.

1.96 fileselect

fileselect SCELREF.HYP
Mit 'fileselect' kann Ihr Makropaket die GEM-Dateiauswahlbox auf den Bildschirm bringen. Dazu erhlt diese Routine als Parameter drei Zeichenketten. Die erste gibt den gewnschten Pfad nebst Suchmaske an, die zweite einen vorgewhlten Dateinamen (kann auch leer sein) und die dritte einen Text, der im Titel der Auswahlbox erscheinen soll (wird erst ab TOS 1.04 genutzt). Der Rckgabewert ist eine Zeichenkette, die den ausgewhlten Dateinamen samt Pfad enthlt. Wenn

der Anwender 'Abbruch' selektiert, wird eine leere Zeichenkette zurckgegeben. Beispiel:

```
  name = fileselect('C:\BIN\*.PRG', '', 'Programm starten');
```

Wenn der Anwender in diesem Pfad eine Datei namens 'TEST.PRG' auswhlt, enthlt 'name' anschlieend die Zeichenkette C:\BIN\TEST.PRG Klickt er dagegen 'Abbruch' an, so bleibt 'name' leer.

1.97 shell

shell SCELREF.HYP

Der Aufruf 'shell' ermglicht es Ihnen, von SCEleton aus weitere GEM- und TOS-Programme zu starten. Die Syntax lautet:

```
    shell(programm, parameter, parallel, argv)
```

'shell' bietet Mglichkeiten, die ber den entsprechenden Dialog von JAnE hinausgehen. Dazu werden vier Argumente bentigt:

'programm'	Der Dateiname des gewnschten Programmes
'parameter'	Die Parameterzeile fr das Programm
'parallel'	Ein Wahrheitswert, der angibt, ob das Programm parallel gestartet werden soll, oder ob der Makro erst nach Ende des Programms weiterlaufen soll.
'argv'	Ein Wahrheitswert, der angibt, ob zur Parameter- bergabe das ARGV-Verfahren verwendet werden soll.

Hier sind einige systemtechnische Anmerkungen erforderlich: Der Parameter 'parallel' wird nur unter Multitasking-Umgebungen beachtet. In anderen Umgebungen wird immer auf das Programmende gewartet. Das ARGV-Verfahren ermglicht die bergabe langer Parameterzeilen (mehr als 124 Zeichen). Leider funktioniert dieses Verfahren zur Zeit nur unter MultiTOS. In anderen Umgebungen wird also der Parameter 'argv' nicht beachtet.

1.98 kommunikation mit anderen programmen

Kommunikation mit anderen Programmen SCELREF.HYP

SCEleton bietet Ihnen sogar einen Zugriff auf die Mglichkeiten von GEM, an parallel ablaufende Programme Nachrichten zu verschicken. Dies betrifft unter normalem TOS die Accessories, unter MultiTOS oder MagiC parallel laufende Anwendungen. Damit knnen Sie von JAnE aus diese Programme fernsteuern, soweit sie diese Mglichkeit vorsehen. Die betreffenden SCEleton-Calls sind die folgenden:

```
applfind
applsend
applreceive
applparms
```

1.99 applfind

applfind SCELREF.HYP
Um einem anderen Programm Nachrichten zu schicken, mssen Sie zuerst einmal herausfinden, ob es gerade luft. Dies leistet der Befehl 'applfind'. Die Syntax lautet:

```
ids = applfind(name);
```

Dabei ist 'name' der Name, unter dem das gesuchte Programm den AES bekannt ist. Dieser Name ist bis zu acht Zeichen lang. Meist handelt es sich dabei um den Dateinamen des Programmes ohne den Extender (z.B. '.APP'). Um ein Programm anzusprechen, bentigt GEM eine Kennzahl. Diese Kennzahl wird von 'applfind' in 'ids' zurckgeliefert. Da es unter Multitaskingsystemen ohne weiteres mglich ist, da dasselbe Programm mehrmals luft, wird 'ids' grundstzlich eine Liste zurckgegeben, die mehrere Kennungen aufnehmen kann. Luft das fragliche Programm (oder Accessory) zur Zeit nicht, so liefert 'applfind' eine leere Liste zurck.

1.100 applsend

applsend SCELREF.HYP
Ist ein Programm einmal aufgefunden, knnen Sie ihm mit 'applsend' eine Nachricht bermitteln.

```
applsend(id, mid, format, contents...);
```

Zunchst bergeben Sie in 'id' die Kennung des angesprochenen Programms, die Sie zuvor mit 'applfind' in Erfahrung gebracht haben. 'mid' enthlt eine Kennzahl, die den Typ der Nachricht identifiziert. Welche Nachrichtentypen ein bestimmtes Programm akzeptiert, knnen Sie dessen Dokumentation entnehmen. Danach stellt GEM einen Bereich von zehn Bytes zur Verfgung, der den Rest der Nachricht aufnimmt. Was genau in diesem Bereich steht, bestimmen Sie mit 'format'. Dabei handelt es sich um eine Zeichenkette, in der der Datentyp jedes Elementes der Nachricht durch ein Zeichen verschlsselt wird:

'w'	Steht fr ein Wort, d.h. eine ganze Zahl zwischen -32768 und 32767 (belegt zwei Bytes)
'l'	Steht fr ein Langwort, d.h. eine beliebige ganze Zahl (belegt vier Bytes)
's'	Steht fr eine 0-terminierte Zeichenkette (belegt vier Bytes)

Achten Sie darauf, da die Summe der Elemente zehn Bytes nicht berschreitet. Welche Parameter Sie fr eine bestimmte Nachricht in welcher Reihenfolge verwenden mssen, wird in der Regel dadurch festgelegt, was das Empfngerprogramm erwartet.

1.101 applreceive

applreceive SCELREF.HYP
Mit 'applreceive' kann Ihr Makro eine Antwort eines anderen Programmes empfangen. Die Syntax lautet:

```
liste = applreceive(id, format, timeout);
```

'id' ist wiederum die Kennung des Kommunikationspartners, 'format' beschreibt (wie bei 'applsend') den zehn Bytes groen Nutzbereich der Nachricht. Falls Sie fr 'id' den Wert -1 einsetzen, wird die erste eingehende Nachricht empfangen. 'timeout' ist die Anzahl der Millisekunden, die hchstens auf die Nachricht gewartet werden soll. Der Rckgabewert von 'applreceive' ist eine Liste, deren erstes Element der Typ der bersandten Nachricht ist. Darauf folgt zu jedem Zeichen in 'format' das passende Datenelement.

Achtung: Falls Sie die Elemente der Nachricht in 'format' nicht korrekt beschreiben, kann es zu unvorhersehbaren Situationen kommen.

1.102 applparms

applparms SCELREF.HYP
Wird ein Makro von JAnE als Reaktion auf eine AES-Nachricht gestartet, so werden die mit der Nachricht gesendeten Parameter zunchst zwischengespeichert. Mit 'applparms' kann der aufgerufene Makro dann die Parameter aus diesem Zwischenspeicher auslesen. Der Aufruf lautet:

```
liste = applparms(format);
```

Dabei ist 'format' eine Zeichenkette, die die erwarteten Parameter beschreibt. Der Aufbau ist bei 'applsend' beschrieben. Der Rckgabewert von 'applparms' ist eine Liste, deren erstes Element der Typ der bersandten Nachricht ist. Darauf folgt zu jedem Zeichen in 'format' das passende Datenelement.

Achtung: Falls Sie die Elemente der Nachricht in 'format' nicht korrekt beschreiben, kann es zu unvorhersehbaren Situationen kommen.

1.103 laufzeitfehler

Laufzeitfehler SCELREF.HYP
Im Falle eines Laufzeitfehlers bricht SCEleton normalerweise die laufende Sequenz ab und gibt eine Fehlermeldung aus. Fr einige reparable Fehler knnen Sie dies verhindern, indem Sie einen speziellen Makro angeben, der im Falle eines Fehlers ausgefhrt wird. Dieser Makro kann dann versuchen, den Fehler zu reparieren und anschlieend die aktuelle Sequenz fortzusetzen. Dieser Fehlermakro mu den Namen 'errortrap' tragen; er wird vom SCEleton-Laufzeitsystem aus aufgerufen und erhlt von diesem vier Parameter mit auf den Weg

(die zum Teil erst im Zusammenhang mit knftigen Entwicklungen
bedeutsam werden): Den Namen des Wrterbuches, den Namen der
Laufzeitroutine, in der der Fehler auftrat, die Nummer der
Fehlerzeile und einen numerischen Code, den die Laufzeitroutine nach
eigenem Gutdnken festlegt. Dieser Code wird von allen bisher
existierenden Laufzeitroutinen auf 0 gesetzt. Ein einfacher
Fehlermakro ist der folgende, der nur die bergebenen Daten ausgibt
und anschlieend die Sequenz beendet:

```
errortrap(dict, func, line, code)
{
  lprint("Dict: %s, Funktion: %s, Zeile: %d, Code: %d",
          dict, func, line, code);
  stop();
}
```

1.104 jane-spezifische funktionen

JAnE-spezifische Funktionen SCELREF.HYP
In Ihren SCEleton-Makros knnen Sie (oder kann der Makrorecorder) auf
alle Mglichkeiten von JAnE zurckgreifen, genau so, als wrden Sie
an der Tastatur sitzen. Dies ermglichen die hier im einzelnen
beschriebenen Laufzeitfunktionen.

Viele Aufrufe erwarten als Parameter Zeilen- oder Spaltenangaben.
Beachten Sie bitte, da diese Angaben in SCEleton grundstzlich ab 0
zhlen, unabhngig von der Einstellung in der JAnE-Dialogbox. Damit
wird erreicht, da die Makros ohne Probleme von einer Konfiguration
zur anderen bertragbar sind.

Die meisten dieser Funktionen lsen bei schwerwiegenden Fehlern einen
Laufzeitfehler aus, der mit einem 'errortrap'-Makro abgefangen werden
kann. Wenn kein 'errortrap' definiert wurde, fhrt ein solcher Fehler
zum Abbruch der Makrosequenz. Beispiel fr eine solche
Fehlersituation: Die Funktion 'openfile' (ffnet eine Textdatei zum
Editieren) findet die Datei nicht, oder beim Lesen tritt ein
GEMDOS-Fehler auf.

Ein weiteres wichtiges Thema fr die Benutzung dieser Funktionen ist
das folgende:

Umsetzung von Dialog-Optionen

Die JAnE-spezifischen SCEleton-Funktionen zerfallen in die folgenden
Gruppen:

Direkte Makrofunktionen
Pufferfunktionen

1.105 umsetzung von dialog-optionen

Umsetzung von Dialog-Optionen SCELREF.HYP
Um die teilweise etwas komplexeren Optionen der JAnE-Dialoge in SCEleton-Code umzusetzen, werden Stringausdrcke verwendet, deren Inhalt bei der Ausfhrung des entsprechenden JAnE-Aufrufes interpretiert wird. Diese String-Konstanten kommen in zwei verschiedenen Formaten vor:

Einfache Schlsselwrter
Parametrische Schlsselwrter

1.106 einfache schlsselwrter

Einfache Schlsselwrter SCELREF.HYP
Dieses Format wird verwendet, wenn genau eine Option codiert werden soll. Beispiel: Der Aufruf 'setcase' wandelt alle Buchstaben im markierten Block in Gro- oder Kleinbuchstaben. Er erhlt ein Argument, das die Art der Umwandlung bestimmt. Dieses Argument ist eine Zeichenkette, die den Inhalt 'lower' (fr Kleinbuchstaben) oder 'upper' fr Grobuchstaben) haben darf. Andere Werte fhren zu einer Fehlermeldung.

Also: Die Zeile

```
setcase('upper');
```

wandelt den aktuellen Block in Grobuchstaben.

1.107 parametrische schlsselwrter

Parametrische Schlsselwrter SCELREF.HYP
Dieses Format ermglicht es, in einer Zeichenkette mehrere Optionen anzugeben. Beispiel: Der Aufruf 'findreplace' fhrt eine Such- und Ersetzoperation aus und bietet alle Mglichkeiten, die auch von der entsprechenden Dialogbox aus erreichbar sind. Er erwartet drei Parameter: Den Suchtext, den Ersatztext und eine Zeichenkette, die die gewhlten Optionen enthlt. Diese besteht aus Eintrgen der Form 'Option=Wert', die von Kommata getrennt werden. Die Suchrichtung wird z.B. von der Option 'direction' gesteuert, die die Werte 'backward' und 'forward' annehmen kann. Soll das Suchmuster als Regulrer Ausdruck interpretiert werden, weisen Sie der Option 'regexp' den Wert 'yes' zu. Der entsprechende Optionsstring sieht also so aus:

```
'direction=forward,regexp=yes'
```

'findreplace' kennt noch weitere Optionen, die, da sie hier nicht explizit gesetzt werden, Defaultwerte erhalten. Die genauen Angaben finden Sie unten. Hier soll nur das Prinzip erlutert werden.

1.108 direkte makrofunktionen

Direkte Makrofunktionen SCELREF.HYP

Die erste Gruppe der JAnE-Funktionen stellt Ihren SCEleton-Makros im Wesentlichen die von der Benutzeroberflche von JAnE aus erreichbaren Funktionen zur Verfgung. Sie werden auch vom Makrorecorder verwendet.

ckey
cnumber
putstring
getline
textlen
openwindow
release
setcursor
inputline
curline
curcol
linelen
setoption
getoption
helpsystem
metasystem
openfile
savefile
saveas
saveinf
loadinf
infpath
mark
unmark
setstyle
plainstyle
getstyle
findreplace
xfindreplace
findnext
replacenext
openbinary
savebinary
setcase
setlock
runprogram
stdformat
stdprint
gdosformat
gdosprint
loadprinter
reformat
xreformat
setprofile
getprofile
fileformat
loadprofile
saveprofile
settablist
fontselect
setfont

```
setfkey
killfkey
getfkey
setmessage
getmessage
myself
setundo
redraw
helpquery
setmark
gotomark
windowsort
markline
markparagraph
markword
markbraces
terminate
```

1.109 ckey

```
ckey                                                        SCELREF.HYP
    ckey(taste);

Die Funktion 'ckey' simuliert einen Befehlstastendruck. Dabei ist
'taste' eine Zeichenkette mit einer Tastenbezeichnung, die
folgendermaen gebildet wird: Eine Zeichentaste (Buchstabe, Ziffer
oder Sonderzeichen) wird durch das entsprechende Zeichen
reprsentiert. Davor steht ein Zeichen, das eine Umschalttaste
kodiert:

    '#' fr 'Alternate'
    '^' fr 'Control'
    '!' fr 'Shift'

(Die blicherweise in Mens verwendeten Zeichen sind ber die normale
Tastatur nicht so leicht zu erreichen, aber wir hoffen, da die
hnlichkeit gro genug ist.) Die Sondertasten werden wie folgt
angegeben:

    'RGT'         Pfeil rechts
    'LFT'         Pfeil links
    'UP'          Pfeil nach oben
    'DN'          Pfeil nach unten
    'HOME'        Home
    'INS'         Insert
    'DEL'         Delete
    'RET'         Return
    'BS'          Backspace
    'UNDO'        Undo
    'TAB'         Tab
    'HELP'        Help
    'ESC'         Esc

Beispiele:
```

```
    ckey('^RET');   # drcke 'Control-Return'
    ckey('A');      # drcke ein 'A'
    ckey('#A');     # drcke 'Alternate-A'
```

Achtung: Welche Funktion durch welche Taste erreicht wird, ist
natrlich von der gerade geladenen Tastaturbelegung abhngig. Um
dieses Problem zu umgehen, knnen Sie alternativ die Funktion
'cnumber' verwenden.

bergeben Sie 'ckey' einen String, zu dem keine Taste existiert, so
wird das erste Zeichen des bergebenen Strings ausgegeben.

1.110 cnumber

cnumber SCELREF.HYP

```
    cnumber(number);
```

'cnumber' tut im Prinzip das Gleiche wie 'ckey'. Der Unterschied
besteht darin, da 'cnumber' die auszufhrende Aktion nicht ber eine
Beschreibung der zu drckenden Tastenkombination auswhlt, sondern
ber eine interne Funktionsnummer, die innerhalb von JAnE direkt mit
den entsprechenden Routinen verbunden ist. Dadurch werden Makros, die
'cnumber' verwenden, unabhngig von der aktuellen Tastaturbelegung.

'number' steht fr die interne Funktionsnummer der bentigten Funktion
steht. Da die Liste der zur Verfgung stehenden Funktionen recht lang
ist, finden Sie diese im Anhang dieses Handbuches, um bei Bedarf
schnell nachschlagen zu knnen.

1.111 putstring

putstring SCELREF.HYP

```
    putstring(string);
```

Fgt im aktuellen Text an der Cursorposition die Zeichenkette 'string'
ein. Kommandos werden dabei nicht interpretiert.

1.112 getline

getline SCELREF.HYP

```
    getline(number)
```

Liefert aus dem aktuellen Text die Zeile Nummer 'number' als
Zeichenkette zurck.

1.113 textlen

```
textlen                                                    SCELREF.HYP
    l = textlen();

Liefert die Anzahl der Zeilen im aktuellen Text zurck. Dieser Aufruf
erwartet keine Parameter.
```

1.114 openwindow

```
openwindow                                                 SCELREF.HYP
    openwindow(name, short)

ffnet das Fenster zum Textpuffer 'name'. Wenn 'short' ungleich 0 ist,
wird 'name' als Kurzname (wie im Menu "Texte"), sonst als
vollstndiger Name (wie im Fenstertitel) aufgefat. Wenn 'name' als
Leerstring bergeben wird, wird "SCEleton" eingesetzt.
```

1.115 release

```
release                                                    SCELREF.HYP
    release();

Dieser Befehl bentigt keine Parameter. Er ermglicht es, ein mittels
'openwindow' geffnetes Fenster aus dem laufenden Makro heraus zu
schlieen. Das hrt sich auf den ersten Blick zwar vielleicht etwas
merkwrdig an, hat aber durchaus seinen Sinn. Standardmig wird
jedes Fenster, das mit 'openwindow' geffnet wird, als von
SCEleton benutzt markiert. Das bewirkt, da es nicht geschlossen
werden kann, solange der Makro noch luft. Damit stellt SCEleton
sicher, da Fenster, die der Makro noch bentigt, auch tatschlich
offen sind.

Diese Markierung sperrt auch das Schlieen des Fensters durch die
SCEleton-Befehle 'ckey' und 'cnumber' (mit Funktionsnummer 59). Um
diese Befehle innerhalb eines Makros dennoch auf ein mit 'openwindow'
geffnetes Fenster anwenden zu knnen, entfernt der Befehl 'release'
das Sperrflag fr das oberste Fenster.
```

1.116 setcursor

```
setcursor                                                  SCELREF.HYP
    setcursor(line, col);

Setzt den Textcursor auf Zeile 'line', Spalte 'col'.
```

1.117 inputline

```
inputline                                                    SCELREF.HYP
    s = inputline();

Wartet, bis der Anwender eine mit "Return" abgeschlossene Zeile in das
Fenster "SCEleton" eingegeben hat und liefert diese zurck.
Whrenddessen kann er bis zu einem gewissen Mae in anderen Puffern
editieren.
```

1.118 curline

```
curline                                                      SCELREF.HYP
    l = curline();

Liefert die Nummer der Cursorzeile im aktuellen Text zurck.
```

1.119 curcol

```
curcol                                                       SCELREF.HYP
    c = curcol();

Liefert die Nummer der Cursorspalte im aktuellen Text zurck.
```

1.120 linelen

```
linelen                                                      SCELREF.HYP
    l = linelen(line);

Liefert die Lnge der Zeile Nummer 'line' im aktuellen Text zurck.
```

1.121 setoption

```
setoption                                                    SCELREF.HYP
    setoption(option, wert);

Diese Funktion trifft die Einstellungen, die Sie auch von Hand in der
Dialogbox "Einstellungen" machen knnen. 'option' und 'wert' sind
Zeichenketten, die Schlsselwrter enthalten. Es sind folgende Codes
mglich:

    Name                Mgliche Werte     Dialogentsprechung
    ------------------------------------------------------------------
    'undo'              'on/off'           Widerrufen mglich
    'autobackup'        'on/off'           Automatischer Datei-
                                           Backup
    'gemclipboard'      'on/off'           GEM-Clipboard benutzen
    'stable'            'on/off'           Cursor spaltenstabil
    'cursorblink'       'on/off'           Cursor blinkend darstellen
```

```
    'countfrom0'        'on/off'           Zeilennummern ab 0
    'exitsave'          'savequery'        Sichern mit Rckfrage
                        'savetexts'        Alles sichern
                        'savedesktop'      Arbeitsumgebung sichern
    'retcode'           'exitdialog'       Returncode-Abfrage
                        '0-6'              Fester Returncode

Beispiele:

    setoption('undo', 'on');
    i = 2;
    setoption('retcode', i + 1);
    s = 'on';
    setoption('stable', s);
```

1.122 getoption

```
getoption                                                    SCELREF.HYP
    s = getoption(option);

Diese Funktion ist das direkte Gegenstck zu 'setoption'. Sie
gestattet es, den aktuellen Wert einer Einstellung zu erfragen.
'getoption' erwartet als einzigen Parameter den Namen einer Option,
analog zum ersten Paramter von 'setoption'. Der Rckgabewert ist eine
Zahl, die gem der folgenden bersicht zu interpretieren ist:

    Option                Mgliche Werte       Bedeutung
    ------------------------------------------------------------------
    'undo'                'off'                aus
                          'on'                 ein
    'autobackup'          'off'                aus
                          'on'                 ein
    'gemclipboard'        'off'                aus
                          'on'                 ein
    'stable'              'off'                aus
                          'on'                 ein
    'cursorblink'         'off'                aus
                          'on'                 ein
    'countfrom0'          'off'                aus
                          'on'                 ein
    'exitsave'            'savequery'          Sichern mit Rckfrage
                          'savetexts'          Alles sichern
                          'savedesktop'        Arbeitsumgebung sichern
    'retcode'             'exitdialog'         Returncode-Abfrage
                          '0-6'                Fester Returncode

Beispiel: Der folgende Makro schreibt eine Liste der Optionen in das
SCEleton-Fenster:

listoptions()
{ lprint("Widerrufen:     %s", getoption("undo"));
  lprint("Datei-Backup:   %s", getoption("autobackup"));
  lprint("GEM-Clipboard:  %s", getoption("gemclipboard"));
  lprint("Spaltenstabil:  %s", getoption("stable"));
  lprint("Nummern ab 0:   %s", getoption("countfrom0"));
```

```
  lprint("Sichern Ende:   %s", getoption("exitsave"));
  lprint("Returncode:     %s", getoption("retcode"));
  lprint("Cursorblinken:  %s", getoption("cursorblink"));
}
```

1.123 helpsystem

```
helpsystem                                                    SCELREF.HYP
    helpsystem(name);
```

Diese Funktion stellt den Namen des aktuellen Hilfesystems ein - hnlich wie in der Box "Einstellungen". 'name' ist eine Zeichenkette von maximal 8 Zeichen, die den Namen des Accessories angibt.

1.124 metasystem

```
metasystem                                                    SCELREF.HYP
    metasystem(name);
```

Wie 'helpsystem', nur da der Metadateibetrachter eingestellt wird.

1.125 openfile

```
openfile                                                      SCELREF.HYP
    openfile(name);
```

Eine Datei wird zum Editieren in einem neuen Puffer geffnet - entspricht "Datei/ffnen". Die Zeichenkette 'name' enthlt den Dateinamen. Im Falle eines Dateifehlers erzeugt die SCEleton-Laufzeitumgebung einen einen Laufzeitfehler, der mit einem 'errortrap'-Makro abgefangen werden kann.

1.126 savefile

```
savefile                                                      SCELREF.HYP
    savefile();
```

Die aktuelle Datei wird gesichert - entspricht "Datei/sichern". Im Falle eines Dateifehlers erzeugt die SCEleton-Laufzeitumgebung einen einen Laufzeitfehler, der mit einem 'errortrap'-Makro abgefangen werden kann.

1.127 saveas

```
saveas                                                    SCELREF.HYP
    saveas(name);

Sichert den aktuellen Text unter dem Namen 'name' - entspricht
"Datei/Speichern als". Die Zeichenkette 'name' enthlt den
Dateinamen. Im Falle eines Dateifehlers erzeugt die
SCEleton-Laufzeitumgebung einen einen Laufzeitfehler, der mit einem
'errortrap'-Makro abgefangen werden kann.
```

1.128 saveinf

```
saveinf                                                   SCELREF.HYP
    saveinf(file);

Sichert die aktuelle Arbeitsumgebung in einer Datei. 'file' ist eine
Zeichenkette, die den Dateinamen angibt. Im Falle eines Dateifehlers
erzeugt die SCEleton-Laufzeitumgebung einen einen Laufzeitfehler, der
mit einem 'errortrap'-Makro abgefangen werden kann.
```

1.129 loadinf

```
loadinf                                                   SCELREF.HYP
    loadinf(file);

Ldt eine neue Arbeitsumgebung aus einer Datei. 'file' ist eine
Zeichenkette, die den Dateinamen angibt. Im Falle eines Dateifehlers
erzeugt die SCEleton-Laufzeitumgebung einen einen Laufzeitfehler, der
mit einem 'errortrap'-Makro abgefangen werden kann.

Achtung: Zum Laden der neuen Umgebung werden zunchst
smtliche derzeit im Speicher befindlichen Texte beseitigt. (Das ist
auch zu beobachten, wenn dieselbe Operation vom Menu aus gestartet
wird.) Dabei werden die Flags, die 'openwindow' setzt (siehe dazu
auch 'release'), implizit zurckgesetzt. D.h. es kann nicht davon
ausgegangen werden, da sich nach der Benutzung von 'loadinf' ein
bestimmter Text noch im Speicher befindet. Auch das Markieren der
Fehlerzeile bei Laufzeitfehlern wird durch diesen Befehl unmglich
gemacht.
```

1.130 infpath

```
infpath                                                   SCELREF.HYP
    infpath(path);

Mit 'infpath' wird der aktuelle Pfad fr die INF-Datei gesetzt oder
gelesen. bergibt man in der Zeichenkette 'path' einen vollstndigen
Pfad (mit dem Dateinamen), so wird in der spezifizierten Datei am
Programmende die aktuelle Arbeitsumgebung gespeichert, falls die
```

entsprechende Option in JAnE aktiv ist. Wird in 'path' ein Leerstring bergeben, so gibt 'infpath' nur die aktuelle Einstellung zurck.

1.131 mark

mark SCELREF.HYP

```
mark(l1, c1, l2, c2)
```

'l1', 'c1', 'l2', 'c2' sind Zahlen. Der Bereich zwischen Zeile 'l1', Spalte 'c1' und Zeile 'l2', Spalte 'c2' im aktuellen Text wird als Block markiert.

1.132 unmark

unmark SCELREF.HYP

```
unmark();
```

Die aktuelle Blockmarkierung im aktuellen Text wird aufgehoben. Der Cursor wird dabei auf den vormaligen Blockanfang gesetzt.

1.133 setstyle

setstyle SCELREF.HYP

```
setstyle(style);
```

Stattet den selektierten Block mit den angegebenen Attributen aus oder bestimmt die Attribute fr die nchste Zeicheneingabe, falls kein Block selektiert ist. (Diese Funktion arbeitet also wie "Stil/Farbe" unter "Bearbeiten".)

Der Parameter 'style' ist eine Zeichenkette, in der die Attribute ber parametrische Schlsselwrter wie folgt kodiert werden:

Name	Werte	Bedeutung
'underlined'	'nil'	Status nicht verndern.
	'off'	Unterstreichung beenden
	'on'	Unterstreichung beginnen
'slanted'	dito	Kursivschrift
'bold'	dito	Fettschrift
'light'	dito	Helle Schrift
'color'	'nil'	Farbe nicht verndern
	'black'	Farben
	'red'	
	'green'	
	'blue'	
	'cyan'	
	'yellow'	
	'magenta'	

Beispiel:

```
    setstyle('underlined=on,color=blue');
```

Fehlende Attribute werden unverndert gelassen, als ob sie mit 'nil' gesetzt worden wren.

1.134 plainstyle

```
plainstyle                                                    SCELREF.HYP
    plainstyle();
```

Schaltet fr den selektierten Block (oder fr die nchsten Eingaben, falls kein Block selektiert ist), alle Attribute aus, abgeschaltet werden, d. h. es wird schwarze, unverzierte Schrift gewhlt.

1.135 getstyle

```
getstyle                                                      SCELREF.HYP
    style = getstyle();
```

Fragt die Attribute des selektierten Blockes oder der nchsten Texteingabe ab. Der Rckgabewert 'style' ist eine Zeichenkette, in der die Attribute genauso wie im Eingabewert fr 'setstyle' kodiert sind.

1.136 findreplace

```
findreplace                                                   SCELREF.HYP
    rv = findreplace(string, replace, optionen);
```

Lst einen Such- und/oder Ersetzvorgang aus. Die Parameter (Zeichenketten) entsprechen den Feldern der Dialogbox "Suchen & Ersetzen..." wie folgt:

```
    'string'        Das Suchmuster
    'replace'       Der Ersetztext
    'optionen'      Dieser String kann eine Reihe parametrischer
                    Schlsselwrter enthalten, die die entsprechenden
                    Optionen auswhlen.

    Option        Werte        Bedeutung
    ------------------------------------------------------------
    'regexp'      'no'         Suchtext ist einfache Zeichenkette
                  'yes'        Suchtext ist regulrer Ausdruck
    'complex'     'no'         Einfaches Ersetzen
                  'yes'        Formatiertes Ersetzen
    'word'        'no'         Es wird ein Wortbestandteil gesucht
                  'yes'        Es wird ein ganzes Wort gesucht
    'case'        'ignore'     Ignoriere Gro/Kleinschreibung
```

```
                'mind'       Beachte Gro/Kleinschreibung
    'direction' 'backward'   Rckwrts suchen
                'forward'    Vorwrts suchen
    'range'     'all'        Ganzen Text durchsuchen
                'cursor'     Ab Cursor suchen
                'selection'  Selektierten Block absuchen
    'replace'   'no'         Nur suchen, nicht ersetzen
                'once'       Einmal ersetzen
                'all'        Alles ersetzen
                'query'      Ersetzen mit Rckfrage
```

Der bei jeder Option als erster aufgefhrte Wert ist gleichzeitig der Defaultwert, der beim Fehlen der Option angenommen wird. Das wird auch weiter unten bei allen Aufrufen, die ber parametrische Schlsselwrter gesteuert werden, so gehandhabt. Ein leerer Optionsstring ist also gleichbedeutend mit:

```
    regexp=no,complex=no,case=ignore,direction=backward,
    range=all,replace=once
```

Beispiel:

```
    rv = findreplace('int', 'short', 'direction=forward,replace=all');
```

Ein Rckgabewert von 0 gibt an, da das Suchmuster nicht gefunden wurde; jeder andere Wert bedeutet, da die Suche erfolgreich war.

1.137 xfindreplace

```
xfindreplace                                                SCELREF.HYP
    rv = xfindreplace(string, replace, optionen, srcstyle, rplstyle);
```

Dies ist eine Erweiterung von 'findreplace', die mit der Einfhrung der Textattribute in JAnE 1.50 eingebaut wurde. Die Parameter sind Zeichenketten folgenden Inhalts:

```
    'string'        Das Suchmuster
    'replace'       Der Ersetztext
    'optionen'      Mgliche Optionen siehe unter 'findreplace'
    'srcstyle'      Die Textattribute, nach denen gesucht werden soll
                    Kodierung der Attribute siehe unter 'setstyle'
    'rplstyle'      Die Textattribute, mit denen ersetzt werden soll.
```

Ein Rckgabewert von 0 gibt an, da das Suchmuster nicht gefunden wurde; jeder andere Wert bedeutet, da die Suche erfolgreich war.

1.138 findnext

```
findnext                                                    SCELREF.HYP
    rv = findnext();
```

Setzt eine Suchoperation fort. Der Befehl entspricht exakt dem

```
Menubefehl "Weiter suchen". Im Gegensatz zum Aufruf ber
'ckey(taste)' oder 'cnumber(45)' liefert 'findnext' allerdings eine
Information ber den Erfolg seiner Bemhungen zurck. Folgende
Rckgabewerte und Bedeutungen sind mglich:

    > 0      noch eine passende Stelle gefunden
   == 0      keine passende Stelle mehr gefunden
    < 0      Es ist ein Fehler aufgetreten
```

1.139 replacenext

```
replacenext                                                  SCELREF.HYP
    replacenext();

Sucht das nchste Auftreten des Suchmusters und ersetzt es durch des
Ersetztext. Folgende Rckgabewerte sind mglich:

    > 0      noch eine passende Stelle gefunden
   == 0      keine passende Stelle mehr gefunden
    < 0      Es ist ein Fehler aufgetreten
```

1.140 openbinary

```
openbinary                                                   SCELREF.HYP
    openbinary(name, eol1, eol2, linelen);

ffnet eine Binrdatei Editieren in einem neuen Puffer. Dabei
bedeuten:

    'name'     Eine Zeichenkette, die den Dateinamen angibt.
    'eol1'     Die Zeilenendekennung im Text. (Zeichenkette)
    'eol2'     Die alternative Zeilenendekennung (vgl.
               entsprechende Dialogbox).
    'linelen'  Die Zeilenlnge. Wird nur beachtet, wenn 'eol1' leer
               ist, d.h. keine Zeilendendekennung in der Datei
               existiert. Der Wert von 'eol1' bestimmt also die
               Stellung der Radiobuttons in der Dialogbox.
```

1.141 savebinary

```
savebinary                                                   SCELREF.HYP
    savebinary(name, eol, setparms);

Sichert den aktuellen Text als Binrdatei. 'name' ist eine
Zeichenkette, die den Dateinamen enthlt. Ferner ist 'eol' die
Zeilenendekennung, die nach jeder Zeile eingefgt wird (darf durchaus
auch leer sein, das entspricht dann dem Radiobutton "Ohne
Zeilenenden"). Wenn 'setparms' ungleich 0 ist, wird dies als
Parameter fr den Text fest eingestellt und bei allen weiteren
Sicherungsvorgngen automatisch beachtet (entspricht der Checkbox
```

"Formatangaben ins Textprofil bernehmen").

1.142 setcase

```
setcase                                                    SCELREF.HYP
    setcase(casemode);
```

Wandelt den selektierten Block in Gro- oder Kleinschrift. 'casemode' ist ein einfaches Schlsselwort, das die Umwandlung bestimmt:

```
    'upper'      Groschrift
    'lower'      Kleinschrift
    'caps'       Kapitlchen
```

1.143 setlock

```
setlock                                                    SCELREF.HYP
    setlock(lockmode);
```

Sperrt den aktuellen Text gegen nderungen oder gibt ihn frei. 'lockmode' ist ein einfaches Schlsselwort, das die Stellung des Schlosses bestimmt:

```
    'lock'        gesperrt, nderungen sind nicht mglich
    'unlock'      freigegeben, nderungen sind mglich
```

1.144 runprogram

```
runprogram                                                 SCELREF.HYP
    runprogram(name, cmdline, options);
```

Startet ein anderes Programm, analog zum Menbefehl "Programm starten...". Alle drei Parameter sind Zeichenketten. 'name' ist der Name der Programmdatei, 'cmdline' ist die Parameterzeile. 'options' enthlt parametrische Schlsselworte, die das Verhalten steuern:

Option	Werte	Bedeutung
'toskeywait'	'yes/no'	Wartet nach TOS-Programmen (nicht) auf Taste
'multiwait'	'yes/no'	Wartet unter Multitasking-Systemen (nicht) auf Programmende
'fromdesktop'	'yes/no'	Programmstart wird (nicht) an den Desktop delegiert

Obwohl JAnE immer nur eine der beiden Optionen anzeigt (die erste ist unter Multi-, die zweite unter Singletaskingsystemen nicht sinnvoll), stellt dieser Aufruf beide Optionen zur Verfgung, damit die erzeugten Makros sich von einem Single- auf ein Multitaskingsystem und umgekehrt bertragen lassen.

1.145 stdformat

```
stdformat                                                    SCELREF.HYP
    stdformat(top, bottom, left, right, head, foot, options);

Legt ein Druckformat fr zuknftige 'stdprint'-Aufrufe fest, die den
aktuellen Text (oder einen Block daraus) mit direkter
Druckersteuerung ausdrucken. Es ist also immer erforderlich, zuerst
'stdformat' und dann 'stdprint' aufzurufen. Einmal 'stdformat'
schafft die Grundlage fr beliebig viele 'stdprint'-Aufrufe mit
demselben Format. Das Format setzt sich zusammen wie folgt:

    'top'           Oberer Rand als Anzahl von Zeilen
    'bottom'        Unterer Rand als Anzahl von Zeilen
    'left'          Linker Rand als Anzahl von Zeichen
    'right'         Rechter Rand als Anzahl von Zeichen
    'head'          Die Kopfzeile als Zeichenkette
    'foot'          Die Fuzeile (dito). Beide knnen auch leer sein.
    'options'       Ein String mit parametrischen Schlsselwrtern:

    Name           Werte             Bedeutung
    ------------------------------------------------------
    'spacing'      'one'             Einzeiliger Druck
                   'half'            Anderthalbzeiliger Druck
                   'two'             Zweizeiliger Druck
    'style'        'pica'            Schriftbreiten Pica
                   'elite'           Schriftbreite Elite
                   'compressed'      Schriftbreite Compressed
    'nlq'          'no'              Druck im Draft-Modus
                   'yes'             Druck im NLQ-Modus
    'number'       'no'              Druck ohne Zeilennummern
                   'yes'             Druck mit Zeilennummern

Die analogen Aufrufe fr den GDOS-Druck sind 'gdosformat' und
'gdosprint'.
```

1.146 stdprint

```
stdprint                                                     SCELREF.HYP
    stdprint(device, options);

Druckt den aktuellen Text mit dem zuvor mit 'stdformat' (siehe
oben) eingestellten Format ber den direkten Druckertreiber.

    'device'        Name des Ausgabegertes ("PRN:", "CON:"
                    oder ein Dateiname).
    'options'       Parametrische Schlsselwrter, die die weiteren
                    Optionen der Dialogbox "Drucken..." steuern.

    Name               Werte      Bedeutung
    ------------------------------------------------------------
    'background'       'no'       Druck im Vordergrund.
                       'yes'      Druck im Hintergrund.
    'formfeed'         'no'       Letzte Seite nicht vorschieben.
```

	'yes'	Letzte Seite vorschieben.
'selection'	'no'	Ganzen Text drucken.
	'yes'	Nur selektierten Block drucken.

Der Makrorecorder erzeugt aus einem Aufruf der Dialogbox beide
Laufzeitaufrufe nacheinander.

1.147 gdosformat

gdosformat SCELREF.HYP

```
gdosformat(device, top, bot, lft, rgt, head, foot, options, meta);
```

Der GDOS-Druck funktioniert analog zum Standarddruck: Zuerst 'gdosformat'
aufrufen, um das GDOS-Format festzulegen, dann 'gdosprint', um den Druck
zu starten. Ein 'gdosformat' reicht auch fr beliebig viele
'gdosprint'-Aufrufe. Die Parameter funktionieren wie folgt:

'device'	Die Gertenummer (wie in ASSIGN.SYS), auf die gedruckt werden soll.
'top'	Oberer Rand im mm.
'bot'	Unterer Rand in mm.
'lft'	Linker Rand in mm.
'rgt'	Rechter Rand in mm.
'head'	Die Kopfzeile als Zeichenkette.
'foot'	Die Fuzeile (dito). Beide knnen auch leer sein.
'options'	Ein String mit parametrischen Schlsselwrtern (s.u.)
'meta'	Der Name der Metadatei als Zeichenkette, wird nur beachtet, falls 'device' die Nummer eines Metadateitreibers ist, also zwischen 31 und 40 liegt.

'options' kann folgende Optionen als parametrische
Schlsselwrter enthalten:

Name	Werte	Bedeutung
'spacing'	'one'	Einzeiliger Druck
	'half'	Anderthalbzeiliger Druck
	'two'	Zweizeiliger Druck
'number'	'no'	Druck ohne Zeilennummern
	'yes'	Druck mit Zeilennummern

1.148 gdosprint

gdosprint SCELREF.HYP

```
gdosprint(font, points, just, options);
```

'gdosprint' startet nach vorangegangenem 'gdosformat' den Druck
ber GDOS. Die Parameter lauten wie folgt:

'font'	Der Index des zu verwendenden Zeichensatzes.

```
    'point'    Die Gre der Zeichen in Punkt (1/72 Zoll).
    'just'     Dieser Parameter wird nicht mehr beachtet; er ist nur
               zur Wahrung der Rckwrtskompatibilitt vorhanden.
    'options'  Ein parametrisches Schlsselwort, das die folgende
               Option codiert:

    Name             Werte      Bedeutung
    ------------------------------------------------------------
    'selection'      'no'       Ganzen Text drucken.
                     'yes'      Nur selektierten Block drucken.

(Hier kann es nur eines geben.)
```

1.149 loadprinter

```
loadprinter                                                SCELREF.HYP
    loadprinter(name);

Ldt einen Druckertreiber aus einer JAnE-Konfigurationsdatei. 'name'
ist eine Zeichenkette, die den Dateinamen enthlt.
```

1.150 reformat

```
reformat                                                   SCELREF.HYP
    reformat(mode, width);

Formatiert den aktuellen Block auf 'width' Spalten Breite. 'mode' ist
ein einfaches Schlsselwort, das die Art der Formatierung angibt:

    'left'         linksbndig
    'center'       zentriert
    'right'        rechtsbndig
    'justify'      Blocksatz
```

1.151 xreformat

```
xreformat                                                  SCELREF.HYP
    xreformat(mode, width, indent);

Formatiert den aktuellen Block auf 'width' Spalten Breite mit
eventueller Einrckung. Diese Funktion ist eine Erweiterung von
'reformat', die die neuen Mglichkeiten in JAnE 1.50 zur Verfgung
stellt. 'indent' gibt die Einrcktiefe in Spalten an. 'mode' ist eine
Zeichenkette mit parametrischen Schlsselwrtern, die bestimmt, was
gemacht werden soll:

    Name           Werte       Bedeutung
    ------------------------------------------------
    'mode'         'left'      Wie bei 'reformat'.
                   'center'
```

```
                    'right'
                    'justify'
    'indent'        'none'      Nichts einrcken.
                    'paras'     Absatzkpfe einrcken
                    'all'       Alle Zeilen einrcken

Beispiel:

    xreformat('mode=justify,indent=paras', 70, 4);
```

1.152 setprofile

```
setprofile                                                  SCELREF.HYP
    setprofile(option, wert);

'setprofile' stellt eine der Optionen des Textprofils zum aktuellen
Text ein. 'option' und 'wert' sind einfache Schlsselwrter, die die
Option und ihren Wert angeben. Es gibt folgende Mglichkeiten:

    Name              Mgliche Werte    Bedeutung
    ------------------------------------------------------------
    'freetabs'        'on'              Freie Tabulatoren
                      'off'             Fester Tabulatorabstand
    'tabsize'         '2-20'            Tabulatorweite bei festem
                                        Abstand
    'expandtabs'      'on'              "Echte" Tabulatoren
                      'off'             Tabs als Leerzeichen
    'inputmode'       'normal'          Normale Texteingabe
                      'indent'          Automatisch Einrcken
                      'wrap'            Automatischer Umbruch
                      'wrapindent'      Einrcken und Umbruch
    'paratext'        'on'              Speichern als Flietext
                      'off'             Speichern als Zeilentext
    'wrapwidth'       (Zahl)            Umbruchspalte
    'infoline'        'memory'          Infozeile: Speicherplatz
                      'cursor'          Infozeile: Curosrposition
                      'tabs'            Infozeile: Tabulatoren
    'overwrite'       'on'              berschreibmodus
                      'off'             Einfgemodus
    'styles'          'none'            Ohne Attribute bearbeiten
                      'include'         Attribute im Text speichern
                      'separate'        Attribute separat speichern
                      'auto'            Syntaktische Hervorhebungen
```

1.153 getprofile

```
getprofile                                                  SCELREF.HYP
    getprofile(option);

'getprofile' fragt die Optionen im Textprofil zum aktuellen Text ab.
'option' ist dabei ein einfaches Schlsselwort, das die fragliche
Option angibt. Der Rckgabewert ist genau das Schlsselwort, mit dem
```

die 'option' unter 'setprofile' eingestellt wurde.

1.154 fileformat

```
fileformat                                                    SCELREF.HYP
    fileformat(type, eol);
```

Dieser Befehl legt das Dateiformat fest. 'type' ist eine Zeichenkette, die den Dateityp als einfaches Schlsselwort angibt. Es gibt die folgenden Mglichkeiten:

Wert	Bedeutung
binary	Binrdatei
tos	TOS-Textdatei (Zeilende CR/LF, hex 0d0a)
unix	UN*X-Textdatei (Zeilenende nur LF, hex 0a)
mac	Mac-Textdatei (Zeilenende nur CR, hex 0d)

Fr den Typ 'binary' gibt die Zeichenkette 'eol' das zu verwendende Zeilenende direkt an. Ist 'eol' leer, so werden keine Zeilenenden geschrieben.

1.155 loadprofile

```
loadprofile                                                   SCELREF.HYP
    loadprofile(datei);
```

Diese Funktion ldt ein neues Profil fr den Text im obersten Fenster. 'datei' ist eine Zeichenkette, die den Dateinamen angibt. Die Funktion entspricht also dem Menpunkt "Profil laden".

1.156 saveprofile

```
saveprofile                                                   SCELREF.HYP
    saveprofile(datei);
```

Diese Funktion sichert das Profil des aktuellen Textes in eine Datei. 'datei' ist eine Zeichenkette, die den Dateinamen angibt. Die Funktion arbeitet also wie der Menpunkt "Profil sichern".

1.157 settablist

```
settablist                                                    SCELREF.HYP
    settablist(liste);
```

Setzt die freien Tabulatorpositionen im Textprofil des aktuellen Textes. 'liste' ist eine Liste (im SCEleton-Sinne) von Ganzzahlen,

die die Tabpositionen enthlt. Beispiel:

```
    settablist(list(1, 4, 7, 10));
```

(Beachten Sie den 'list'-Aufruf, der die Argumente in die
erforderliche Gestalt bringt.)

1.158 fontselect

fontselect SCELREF.HYP

```
    fontselect(info);
```

Dieser Befehl ruft die Zeichensatzauswahl auf, damit der Anwender
einen Bildschirmzeichensatz auswhlen kann. Er funktioniert nicht fr
Druckerzeichenstze. 'info' ist eine Infozeile, die im Titel der Box
erscheint. Der Rckgabewert ist eine Ganzzahl, deren obere 16 Bit den
Zeichensatzindex (eine eindeutige Kennnummer, die vom Hersteller des
Zeichensatzes vergeben wird) und in den unteren 16 Bit die gewhlte
Punktgre enthlt. Beispiel:

```
    result = fontselect("Zeichensatzauswahl");
    zsatz = result / 65536;
    punkt = result & 0xffff;
```

Danach steht der Index in 'zsatz' und die Punktgre in 'punkt'.

1.159 setfont

setfont SCELREF.HYP

```
    setfont(zsatz, punkt);
```

Diese Funktion ist das Gegenstck zu 'fontselect'; sie stellt einen
Zeichensatz fr das oberste Fenster ein. 'zsatz' mu beim Aufruf den
Zeichensatzindex, 'punkt' die Punktgre enthalten. Das Beispiel zu
'fontselect' vervollstndigt sich also so:

```
    result = fontselect("Zeichensatzauswahl");
    zsatz = result / 65536;
    punkt = result & 0xffff;
    setfont(zsatz, punkt);
```

1.160 newpackage

newpackage SCELREF.HYP

```
    newpackage(name, options);
```

'newpackage' erzeugt eine neues Makropaket aus dem (hoffentlich
fehlerfreien) Quelltext im obersten Fenster. 'name' ist eine
Zeichenkette, die den Dateiname angibt, den das neue Paket erhalten
soll. 'options' ist eine Zeichenkette, die die folgenden

parametrischen Schlsselworte enthalten kann:

Name	Werte	Bedeutung
'save'	'none'	Nichts sichern
	'source'	Den Quelltext sichern
	'package'	Das neue Paket sichern
	'both'	Beides sichern
'old'	'replace'	Im Falle einer Kollision das alte Paket ersetzen
	'link'	Im Falle einer Kollision beide Pakete verbinden.

1.161 setfkey

```
setfkey                                                    SCELREF.HYP
    setfkey(key, name, package);
```

'setfkey' belegt eine Funktionstaste mit einem Makro. Hierzu sind drei Parameter (Zeichenketten) erforderlich: 'key' spezifiziert die zu belegende Taste. Es sind zwei Typen von Strings mglich:

```
    '!#X'       belegt Shift-Alternate-Kombinationen
    'sFnn'      belegt die eigentlichen Funktionstasten
```

In der ersten Form ist das '!#' ein unvernderlicher Bestandteil, das 'X' steht fr einen Buchstaben (auer den Umlauten), eine Ziffer oder eines der Zeichen '~', '#' oder '+'. In der zweiten Form ist nur das 'F' unvernderlich. Das 'nn' steht fr die Nummer der zu belegenden Funktionstaste (wobei fhrende Nullen weggelassen werden knnen), das 's' steht fr einen der folgenden Shift-Indikatoren:

```
    '!'      Shift
    '^'      Control
    '#'      Alternate
```

Wenn eine unmodifizierte Funktionstaste belegt werden soll, entfllt der Shift-Indikator.

'name' gibt den Namen des Makros an und 'package' den Dateinamen des Paketes.

1.162 killfkey

```
killfkey                                                   SCELREF.HYP
    killfkey(key);
```

'killfkey' ist das Gegenstck zu 'setfkey': Die Belegung einer Funktionstaste wird gelscht. 'key' ist eine Zeichenkette, die die Taste angibt, deren Belegung gelscht werden soll. Die Spezifizierung der Taste erfolgt wie bei der Funktion 'setfkey'.

1.163 getfkey

```
getfkey                                                    SCELREF.HYP
    fkey = getfkey(key);
```

Mit dieser Funktion kann ein Makropaket prfen, ob eine Funktionstaste schon belegt ist, und wenn ja, womit. 'key' ist eine Zeichenkette, die die Taste angibt (Aufbau wie fr 'setfkey'). Der Rckgabewert ist in beiden Fllen eine Zeichenkette, die den Makronamen UND den Dateinamen des Makropaketes enthlt, getrennt durch einen Semikolon.

1.164 setmessage

```
setmessage                                                 SCELREF.HYP
    setmessage(messid, name, package);
```

'setmessage' belegt eine AES-Nachricht mit einem Makro. Hierzu sind drei Parameter (Zeichenketten) erforderlich: 'messid' spezifiziert die zu belegende Nachrichtennummer. 'name' gibt den Namen des Makros an und 'package' den Dateinamen des Paketes.

1.165 killmessage

```
killmessage                                                SCELREF.HYP
    killmessage(messid);
```

'killmessage' ist das Gegenstck zu 'setmessage': Die Belegung einer AES-Nachricht wird gelscht. 'messid' ist die Nachrichtennummer, deren Belegung gelscht werden soll.

1.166 getmessage

```
getmessage                                                 SCELREF.HYP
    mess = getmessage(messid);
```

Mit dieser Funktion kann ein Makropaket prfen, ob eine AES-Nachricht schon belegt ist, und wenn ja, womit. 'messid' ist die AES-Nachrichtennummer. Der Rckgabewert ist in beiden Fllen eine Zeichenkette, die den Makronamen UND den Dateinamen des Makropaketes enthlt, getrennt durch einen Semikolon.

1.167 myself

```
myself                                                     SCELREF.HYP
    filename = myself();
```

Mit dieser Funktion erfragt ein Makropaket seinen eigenen vollstndig qualifizierten Dateinamen.

1.168 setundo

setundo SCELREF.HYP

```
    setundo(switch);
```

Dieser Aufruf schaltet die Mglichkeit des Widerrufens ("Undo") fr den laufenden Makro ein oder aus. Der Wert von 'switch' gibt die Stellung des Schalters an: 0 bedeutet "Aus", 1 bedeutet "Ein". Beim Einschalten wird gleichzeitig der Zustand des obersten Fensters zum knftigen Wiederherstellen gespeichert. 'setundo' liefert die vorherige Stellung des Schalters als Rckgabewert. 'setundo(-1)' fragt den Schalter nur ab und bewirkt keine nderung.

Achtung: Beim Start eines Makros wird das Widerrufen grundstzlich abgeschaltet, so da Sie es - falls sinnvoll - mit dem Befehl 'setundo(1)' explizit wieder einschalten mssen. Dadurch wird sichergestellt, da der Start eines Makros in jedem Falle als Wechsel des Operationstyps gilt. Der Makrorecorder enthlt die Option "Makro aktiviert Undo", die bewirkt, da zu Beginn des Makros das Statement 'setundo(1);' eingefgt wird.

Der Befehl 'setoption('undo', 'on');' beeinflut nur den globalen UNDO-Schalter, nicht aber den Schalter, der fr das Widerrufen von nderungen innerhalb eines Makros zustndig ist.

1.169 redraw

redraw SCELREF.HYP

```
    redraw();
```

Dieser Befehl bewirkt ein vollstndiges Neuzeichnen aller Fenster, die JAnE geffnet hat. Dies kann z.B. erforderlich sein, nachdem ein Text mit den Pufferfunktionen bearbeitet wurde, um die nderungen auch am Bildschirm sichtbar werden zu lassen.

1.170 helpquery

helpquery SCELREF.HYP

```
    helpquery(word);
```

'helpquery' bergibt eine Zeichenkette an das eingestellte Hilfesystem. Diese Zeichenkette wird im bergabeparameter 'word' spezifiziert.

1.171 setmark

setmark SCELREF.HYP

```
    setmark(index, describe);
```

'setmark' setzt eine Sprungmarke im obersten Fenster an die Cursorposition. 'index' gibt die Nummer der Marke an, die zwischen 1 und 6 liegen kann. 'describe' ist eine Zeichenkette, die kurze Informationen ber die Marke enthalten kann. Dies entspricht den Mglichkeiten, die die Dialogbox unter "Suchen/Marke setzen..." bietet.

1.172 gotomark

gotomark SCELREF.HYP

```
gotomark(index);
```

Dieser Befehl setzt den Cursor an die Marke mit der Nummer 'index'. Befindet sich die Marke nicht im obersten Fenster, so wird SCEleton das entsprechende Fenster ffnen bzw. nach vorne holen.

1.173 windowsort

windowsort SCELREF.HYP

```
windowsort(mode);
```

Dieser Befehl ordnet die offenen Textfenster von JAnE auf dem Bildschirm neu an. Er entspricht damit dem Menpunkt "Texte/Fenster anordnen...". Der Parameter 'mode' sollte ein einfaches Schlsselwort enthalten, das die Art der Anordnung bestimmt:

'overlap'	berlappend
'horizontal'	nebeneinander
'vertical'	untereinander
'tiled'	gekachelt

1.174 markline

markline SCELREF.HYP

```
markline();
```

Dieser Befehl selektiert im obersten Textfenster die Zeile, auf der sich der Cursor gerade befindet. Er entspricht also einem Mausklick mit gedrckter 'Alternate'-Taste.

Wird ein Block mit 'markword' oder 'markline' selektiert, so erfolgt eine Blockerweiterung mit Shift-Mausklick anschlieend ebenfalls wort- bzw. zeilenweise.

1.175 markparagraph

markparagraph SCELREF.HYP

```
    markparagraph();
```

Dieser Befehl selektiert im obersten Textfenster den Ansatz, auf dem sich der Cursor gerade befindet. Er entspricht also einem Doppelklick mit gedrckter 'Alternate'-Taste oder dem Menbefehl "Absatz auswhlen".

1.176 markword

markword SCELREF.HYP

```
    markword();
```

Dieser Befehl selektiert im obersten Textfenster das Wort, auf dem sich der Cursor gerade befindet. Er entspricht also einem Doppelklick auf dieses Wort.

Wird ein Block mit 'markword' oder 'markline' selektiert, so erfolgt eine Blockerweiterung mit Shift-Mausklick anschlieend ebenfalls wort- bzw. zeilenweise.

1.177 markbraces

markbraces SCELREF.HYP

```
    markbraces();
```

Dieser Befehl selektiert im obersten Textfenster einen geklammerten Bereich, wenn sich der Cursor gerade auf dem Klammerbegrenzer befindet. Er entspricht also einem Doppelklick auf diesen Klammerbegrenzer.

1.178 terminate

terminate SCELREF.HYP

```
    terminate();
```

Dieser Befehl beendet JAnE, genauso, als wrden Sie den Menpunkt "Datei/Programmende" auswhlen.

1.179 pufferfunktionen

Pufferfunktionen SCELREF.HYP

Whrend die direkten Makrofunktionen stets das oberste Fenster bearbeiten knnen, da sie die Arbeit mit der Benutzerschnittstelle simulieren, gestatten Ihnen die Pufferfunktionen, jeden beliebigen Puffer anzusprechen. Er braucht dazu noch nicht einmal in einem offenen Fenster zu liegen. Jeder Puffer wird dabei durch eine

Kennzahl, ein sogenanntes "handle" identifiziert. Dieses Konzept
hnelt der Dateiverwaltung unter GEMDOS.

Zustzlich zu den Puffern, in denen Textdateien bearbeitet werden,
existieren noch drei sogenannte temporre Puffer, die als
Zwischenspeicher fr grere Textstcke bei komplexeren Vorgngen
dienen knnen. Sie haben stets die Handles -1 bis -3.

Es gibt die folgenden Pufferfunktionen

buffind
buflines
bufgetline
bufputline
bufinsertline
bufdeleteline
bufselstart
bufselend
bufselstartcol
bufselendcol
bufunmark
bufquery
bufname
tempcopy
tempcut
temppaste

1.180 buffind

buffind SCELREF.HYP

```
buffind(name, short);
```

Diese Funktion liefert ein Handle fr den Puffer mit Namen 'name'
(Zeichenkette). Wenn 'short' ungleich 0 ist, wird 'name' als Kurzname
(wie im Menu 'Texte'), sonst als vollstndiger Name (wie im
Fenstertitel) aufgefat. Wenn 'name' als Leerstring bergeben wird,
wird ein Handle fr das oberste Fenster zurckgeliefert. Ein Handle
ist immer grer oder gleich -3. Im Fehlerfall (Namen nicht gefunden
bzw. kein Fenster offen) ist das Ergebnis -10, also kleiner als -3.

1.181 buflines

buflines SCELREF.HYP

```
lines = buflines(handle);
```

Dieser Aufruf liefert die Anzahl Zeilen im Puffer, der durch 'handle'
identifiziert wird.

1.182 bufgetline

```
bufgetline                                                  SCELREF.HYP
    bufgetline(handle, line);

Liefert die Zeile Nummer 'line' aus dem Puffer mit Handle 'handle' als
Zeichenkette zurck.
```

1.183 bufputline

```
bufputline                                                  SCELREF.HYP
    bufputline(handle, line, string);

Ersetzt im Puffer mit Handle 'handle' den Inhalt der Zeile Nummer
'line' durch die Zeichenkette 'string'.
```

1.184 bufinsertline

```
bufinsertline                                               SCELREF.HYP
    bufinsertline(handle, line, string);

Fgt im Puffer mit Handle 'handle' vor der Zeile 'line' eine neue
Zeile mit der Zeichenkette 'string' als Inhalt ein. Damit erhlt die
neue Zeile die Nummer 'line'; die entsprechende alte Zeile wird
aufgeschoben. Ist 'line' gleich der Anzahl der Zeilen im Puffer, so
wird eine neue letzte Zeile angehngt. (Beachten Sie bitte: Es wird
ab 0 gezhlt.)
```

1.185 bufdeleteline

```
bufdeleteline                                               SCELREF.HYP
    bufdeleteline(handle, line);

Lscht im Puffer 'handle' die Zeile mit der Nummer 'line'.
```

1.186 bufselstart

```
bufselstart                                                 SCELREF.HYP
    line = bufselstart(handle);

Diese Funktion ermittelt im Puffer mit dem Handle 'handle' die Zeile,
in der der Blockanfang steht. Ist kein Block selektiert, so gibt die
Funktion den Wert -1 zurck. Die restlichen Koordinaten des Blocks
ermitteln Sie mit

bufselend
bufselstartcol
bufselendcol
```

1.187 bufselend

```
bufselend                                                    SCELREF.HYP
    line = bufselend(handle);

Diese Funktion ermittelt im Puffer mit dem Handle 'handle' die Zeile,
in der das Blockende steht. Ist kein Block selektiert, so gibt die
Funktion den Wert -1 zurck. Die restlichen Koordinaten des Blocks
ermitteln Sie mit

bufselstart
bufselstartcol
bufselendcol
```

1.188 bufselstartcol

```
bufselstartcol                                               SCELREF.HYP
    col = bufselstartcol(handle);

Diese Funktion ermittelt im Puffer mit dem Handle 'handle' die Spalte,
in der der Blockanfang steht. Ist kein Block selektiert, so gibt die
Funktion den Wert -1 zurck. Die restlichen Koordinaten des Blocks
ermitteln Sie mit

bufselstart
bufselend
bufselendcol
```

1.189 bufselendcol

```
bufselendcol                                                 SCELREF.HYP
    col = bufselendcol(handle);

Diese Funktion ermittelt im Puffer mit dem Handle 'handle' die Spalte,
in der das Blockende steht. Ist kein Block selektiert, so gibt die
Funktion den Wert -1 zurck. Die restlichen Koordinaten des Blocks
ermitteln Sie mit

bufselstart
bufselend
bufselstartcol
```

1.190 bufunmark

```
bufunmark                                                    SCELREF.HYP
    bufunmark(handle);

Entfernt im Puffer mit Handle 'h' die Blockmarkierung.
```

1.191 bufquery

```
bufquery                                                    SCELREF.HYP
    handle = bufquery();
```

Diese Routine ffnet das Popup mit der Liste der geladenen Texte und liefert den Handle des vom Anwender selektierten Puffers zurck. Wenn der Anwender die Auswahl abbricht (Klick auerhalb des Popups oder Taste "ESC", gibt die Routine den Wert -10 zurck.

1.192 bufname

```
bufname                                                     SCELREF.HYP
    name = bufname(handle, short);
```

Liefert den Dateinamen zu dem Puffer mit Handle 'handle' zurck. Dies ist z.B. ntzlich, um den Namen einem Compiler o.. zu bergeben. Wenn 'short' ungleich 0 ist, wird der kurze Name (der im "Texte"-Popup erscheint) ermittelt, sonst der vollstndige Name (der im Fenstertitel erscheint). Dieser Aufruf ist natrlich nicht auf die temporren Puffer anwendbar, d.h. 'handle' darf nicht kleiner als 0 sein.

Die zurckgegebenen Zeichenkette 'name' enthlt - genau wie die Fenstertitel - neben dem Namen noch weitere Informationen. Es gilt die Belegung:

Stelle	Inhalt
0	'~', wenn der Text gegen nderungen gesperrt ist, ' ' sonst
1	'*', wenn der Text seit dem letzten Sichern verndert wurde ' ' sonst.
2 - n	Der eigentliche Puffername.

1.193 tempcopy

```
tempcopy                                                    SCELREF.HYP
    tempcopy(handle, append);
```

Dieser Befehl kopiert den selektierten Block im aktuellen Fenster in den temporren Puffer mit dem Handle 'handle'. 'handle' kann daher nur Werte zwischen -1 und -3 annehmen. Ist 'append' verschieden von 0, so wird der kopierte Text an den temporren Puffer angehngt. Andernfalls wird der temporre Puffer vor der Operation gelscht.

1.194 tempcut

tempcut SCELREF.HYP

```
tempcut(handle, append);
```

Dieser Befehl verschiebt den selektierten Block im aktuellen Fenster in den temporren Puffer mit dem Handle 'handle'. 'handle' kann daher nur Werte zwischen -1 und -3 annehmen. Ist 'append' verschieden von 0, so wird der kopierte Text an den temporren Puffer angehngt. Andernfalls wird der temporre Puffer vor der Operation gelscht.

1.195 temppaste

temppaste SCELREF.HYP

```
temppaste(handle, copy);
```

'temppaste' fgt den Inhalt des temporren Puffers mit Handle handle' im obersten Fenster an der Cursorposition ein. Auch hier darf 'handle' nur zwischen -1 und -3 liegen. Ist 'copy' ungleich 0, so behlt der temporre Pudffer seinen Inhalt, d.h. es wird eine Kopie dieses Puffers eingefgt. Andernfalls ist der temporre Puffer anschlieend leer.