

1. Kurze Dokumentation der Hilfsprogramme

Wichtig ist hier die Kommunikation zwischen Programm und Nutzer über Umgebungsvariable, was die Skriptsprache in der gewünschten Weise nicht leistet. Diese Funktionalität wird durch folgende Methode erreicht: Es ist eine Umgebungsvariable 2-Punkt wie folgt zu setzen: `set ..=call aa.cmd`. Wenn im Aufruf eines Programms eine – evt. modifizierte – Option der Form `-:var` auftritt, wird die 1. Zeile der Ausgabe des Programms der Umgebungsvariablen `var` zugewiesen; diese Zuweisung ist einerseits im aktuellen Programm sofort verfügbar und wird andererseits in die Datei `aa.cmd` (Beispiel) geschrieben; Beispiel:

```
set ..=call aa.cmd
# cc.txt --A1 -:bb & %..%
# cc.txt -:;bb & %..%
```

Im ersten Falle erhält die Umgebungsvariable `bb` als Wert die 2. Zeile der Datei `cc.txt`, im zweiten Falle werden alle Zeilen der Datei – durch Semikolon getrennt – der Umgebungsvariablen `var` zugewiesen.

Es ist zu verhindern, dass überlange Zeilen entstehen.

Umwandlung einer Text-Datei in ein LaTeX-Dokument:

```
#txt2ltx inputname [outputname]
```

Es ist auf richtige Zeilenlänge zu achten, da der Text selbst nicht umgewandelt, sondern in verbatim-Blöcke aufgeteilt wird.

Die Ausgabedatei ist – unter Anwendung der in diesem System vorhandenen Möglichkeiten – sowohl als eigenständiges LaTeX-Dokument als auch als input-Datei für andere Dokumente ohne Änderungen verwendbar.

Behandlung von Kennwörtern:

```
#pwd ...
```

```
#####
```

Umwandlung von ansi nach unicode:

```
#dop :ansi_uni eingabe_datei ausgabe_datei
```

Dabei erhalten reg-Dateien die richtige Anfangskennung.

Ausgabe der nächsten Zahl:

```
hexadezimal: #dop :hex parm -:var & %..%
```

Der Parameter `parm` enthält am Anfang eine Zahl, deren nächste auf `var` hexadezimal gespeichert wird.

Ausgabe einer einfachen reellen Operation:

```
#dop : var[+|-|*|/]=1.Operand{+|-|*|/}2.Operand ... & %..%
```

Die Operanden müssen Zahlen oder mit \$ beginnende Variable sein (eine im Aufruf bereits berechnete Variable).

Beispiel:

```
#dop : a=1+5 b=2*$a c=$b/4 b/=$a & %.% & %..%
```

Danach gilt a=6, b=2 und c=3.

Die Script-Sprache bietet nur eine ganzzahlige Arithmetik, die z. B. für LaTeX-Formate ungeeignet ist.

#####

Aufruf:

```
# {input [anfang] [ende] operation [output]}
```

```
input      : -I {inputfile| ""}| -L stringliste1
anfang     : {-|+}A[+[A=]] anfangswort
ende       : {-|+}E[+[A=]] endewort
operation  : {-o stringliste2} | {+o [maske] [wort]} | --o| ++o
output     : -O[+] outputfile
```

Aus einer Datei oder Liste von Wörtern können auf verschiedene Weisen Zeilen/Wörter modifiziert ausgewählt und formatiert ausgegeben werden. Wenn ein Wort der Parameterliste mit @ beginnt, steht ab dem 2. Zeichen der Name einer Datei, deren Inhalt zeilenweise in die Parameterliste aufgenommen wird.

Wenn ein Wort in stringliste mit \$ beginnt, steht danach der Name einer Umgebungsvariablen; ihr Wert wird in die Parameterliste aufgenommen, wobei standardmäßig das Leerzeichen als Trennzeichen zweier Wörter fungiert. Soll ein anderes Zeichen als Trennzeichen erkannt werden, ist dieses dem Namen der Umgebungsvariablen voranzustellen.

Als mögliche Trennzeichen werden erkannt !\$/()=?{[]}*'+<>|;: .

Standardmäßig wird jedes Wort zu einem Parameter. Soll der gesamte Wert der Umgebungsvariablen zu einem Parameter werden, ist ein Anführungsstrich dem Namen voranzusetzen:

```
# -L $"Path      oder   # -L $"Path\
```

Wenn der Name einer Umgebungsvariablen mit ~ beginnt, handelt es sich um eine interne Variable; bei ihr ist das neue-Zeile-Zeichen das Trennzeichen.

Ihr Wert wird nicht nach außen ausgegeben. Sie nimmt ganze Zeilen als ein Wort auf. Ihr Wert kann als Parameter wieder in die Parameterliste eingeführt werden.

Wenn ein Parameter mit ~ beginnt, ist der Parameter maskiert.

Das Neue-Zeile-Zeichen wird in einem Parameter wie folgt kodiert:

Der Parameter ist mit ~ zu maskieren; sodann kodiert man wie üblich durch \n , z. B.

-e ~a\nbc\n oder -e "~a\nbc\n"

Sollte stringliste mit einem Sonderzeichen @, \$, +, -

beginnen, so wird es mit einem vorangestellten ^ maskiert, d. h. das betreffende Sonderzeichen ist wirkungslos; in der Eingabezeile muss ^ doppelt kodiert sein, falls der Parameter nicht in Anführungsstrichen steht. Durch ~ wird nur das 2. Zeichen in stringliste maskiert; dadurch darf stringliste mit - oder + beginnen.

-I {inputfile | ""} :

Ist inputfile das Leerwort, wird die Standardeingabe verwendet.

Falls nach inputfile keine Option, sondern ein String kommt, charakterisiert dieser einen Blockanfang; das Blockende wird auf die nächste Leerzeile gesetzt, falls kein weiterer String folgt. Anfang und Ende gehören nicht zum Block.

-L[A] stringliste1 :

Fall A oder leer:

Jedes Wort aus stringliste1 wird als eine Zeile genommen. Die Daten werden am Anfang eingekettet; Standard ist das Ende.

-.[:|;|+|-|=][extract_itex]var[:act] : Ausgabe der ersten aktuellen Zeile auf die Umgebungsvariable var, falls keine Zusatz-Option angegeben ist. Falls die Variable var mit ~ beginnt, handelt es sich um eine interne Variable, deren Wert nicht nach außen ausgegeben wird; sie darf in der Parameterliste auftreten, z. B. in der Form

... -L[/extract_itex]~ss ...

\$: Die Variable wird fortgeschrieben.

: : alle aktuellen Zeilen auf var; Leerzeichen ist Trennzeichen;

; : alle aktuellen Zeilen auf var; Semikolon ist Trennzeichen; bei einer internen Variablen ist das neue-Zeile-Zeichen das Trennzeichen.

+ : Die erste aktuelle Zeile.

- : Die letzte aktuelle Zeile.

= : Anzahl der aktuellen Zeilen.

sonst : die erste aktuelle Zeile wird auf var ausgegeben.

act :

delete : Alle aktuellen Zeilen werden gelöscht.

Delete : Alle Zeilen werden gelöscht.

return : Falls der Variablen var kein Wert zugewiesen wurde,
 wird das Programm beendet.
 exit : Falls der Variablen var kein Wert zugewiesen wurde,
 wird 'exit' ausgegeben und das Programm beendet.

-A[A|=][z] anfangswort :
 Die folgenden Operationen beginnen nach der ersten Zeile mit
 anfangswort; die Zeilen bis einschließlich anfangswort werden
 gelöscht. Im Falle A wird auf Übereinstimmung am Zeilenanfang,
 bei = auf Übereinstimmung mit der Zeile getestet.
 Falls keines dieser Zeichen auftritt, wird anfangswort in einer
 Zeile gesucht.
 Diese Art der Optionspezifikation wird auch in weiteren Optionen
 verwendet. Falls anfangswort nicht kodiert ist, wird die nächste
 Leerzeile gesucht.
 Falls ein Zeichen z codiert ist, wird bis zur letzten, gefundenen
 Zeile vorgesetzt.

-E[A|=][z] endewort :
 Die folgenden Operationen enden vor der ersten Zeile mit endewort;
 die Zeilen ab endewort werden gelöscht. Falls endewort nicht kodiert
 ist, wird die nächste Leerzeile gesucht.
 Falls ein Zeichen z codiert ist, wird bis zur letzten, gefundenen
 Zeile vorgesetzt.

Die durch A und E eingeschlossenen Zeilen bilden die aktuellen Zeilen.
 Falls diese Optionen nicht kodiert sind, sind alle Zeilen aktuell.

-e[A|E|=|D|S|f|l|+] stringliste2 :
 Alle Zeilen mit einem Wort aus stringliste2 werden ausgegeben.
 D : stringliste2 enthält nur relative Pfade von Ordnern;
 es werden alle Zeilen ausgewählt, die einen dieser Pfade
 am Anfang enthält.
 S : die aktuellen Zeilen werden durch die Wortfolge in stringliste2
 sortiert.
 f : die erste, gefundene Zeile.
 l : die letzte, gefundene Zeile.
 + : Wenn die Zeile das erste Wort aus stringsliste2 enthält, wird
 sie ausgewählt, falls sie ein Wort aus dem Rest von stringliste2
 enthält. Wenn die Zeile das erste Wort aus stringliste2 nicht
 enthält, wird sie ausgewählt.

-getvariable var1 var2 ...
 Die Werte angegebenen Variablen Block werden ausgegeben:
 set var1=...
 set var2=...
 usw.

-insertzzz stringliste2
 Die Zeilen der Datei sind alphabetisch aufsteigend geordnet,

wobei in jeder Zeile das Wort zzz steht und ab diesem die Ordnung beginnt. Die Wörter aus stringliste2 werden eingeordnet.

-putvariable var1[=...] var2[=...] ...
 Die angegebenen Variablen die kodierte Werte.
 Ist kein Wert im Aufruf angegeben, wird der Wert der Umgebungsvariablen genommen.

-O[+[vorspann]] [outputfile]:
 Dateiausgabe; bei der Option + wird die Ausgabe an die Datei angehängt.
 Ist vorspann angegeben, wird dieses Wort jeder Zeile vorangestellt (z. B. Dateipfad), falls es kein Prozentzeichen enthält.
 Andernfalls wird vorspann als Format-Anweisung genommen.
 Wenn outputfile fehlt, werden alle Zeilen in die letzte Eingabedatei ausgegeben, andernfalls nur die aktuellen.

-t x1 y2 x2 y2 ...
 In jeder Zeile z werden alle Vorkommen von x1 durch y1, x2 durch y2 usw. ersetzt. Für jede einzelne Ersetzung wird stets die schon teil-ersetzte Zeile verwendet, wodurch Mehrfach-Ersetzungen möglich sind.
 Null-Zeichen-Ersetzung bedeutet den Zeilenanfang.

-T[w] x1 y2 x2 y2 ...
 wie bei -t; die transformierte Zeile und die aktuelle Zeile bilden, getrennt durch das Zeichen w die neue Zeile.

-v[A|E|=|+] stringliste2 :
 Alle Zeilen ohne ein Wort aus stringliste2.
 Im Falle der +-Option passiert folgendes.
 Wenn die Zeile das erste Wort aus stringliste2 enthält, wird sie gestrichen, falls sie ein Wort aus dem Rest von stringliste2 enthält.
 Wenn die Zeile das erste Wort nicht enthält, wird sie übernommen.

-? Aufforderungstext :
 Selektion aus einer Liste, die aus den aktuellen Zeilen besteht; die ausgewählten bilden neue Zeilen. Nach dem Aufruf erscheinen die Auswahlwörter aufsteigend nummeriert auf dem Bildschirm mit der Aufforderung, eine Auswahl zu treffen. Bei der Auswahl dürfen Zahlen in der Form i i-k mit Wiederholung eingegeben werden. Die entsprechenden Wörter werden ausgegeben; im Falle der Eingabe von < ist das Leerwort der Wert.
 Im Aufforderungstext darf eine Vorauswahl angegeben sein, die im Falle der Eingabe von <enter> ausgegeben wird; die Vorauswahl muss dabei mit (enter= eingeleitet werden und mit) enden; im Falle (enter=all) werden alle Wörter ausgegeben.
 Ist keine Vorauswahl angegeben und wird nur <enter> eingegeben, ist das Leerwort der Wert. Ein Auswahlwort darf maximal 70 Zeichen haben.
 Beispiel für die Anwendung:

```
# -L %files% -? "==> Bitte auswählen: " -:file & %..%
```

Die auf der Umgebungsvariablen files stehenden Dateinamen werden zur Auswahl gestellt; nach dem Aufruf enthält die Umgebungsvariable file den ausgewählten Namen. Im Falle des Aufrufs

```
# -L %files% -? "==> Bitte auswählen: " -::file & %..%
```

werden der Variablen file alle ausgewählten Namen zugewiesen.

Die folgenden --Optionen haben keine weiteren Parameter.

-- : Alle vorhandenen Zeilen werden aktuell.

--Azahl :

Der Operationsanfang wird um zahl Zeilen versetzt;
die übersprungenen Zeilen werden gelöscht.

--Ezahl :

Das Operationsende wird um zahl Zeilen (relativ zum
Operationsanfang) versetzt; die folgenden Zeilen werden gelöscht.

--c[x] :

Jede Zeile wird beim ersten Zeichen x geteilt, beide
Teilzeilen werden umgekehrt mit x verkettet.
Wenn x eine Zahl ist, wird nach dem x-ten Buchstaben geteilt.

--C[x] :

Jede Zeile wird beim letzten Zeichen x geteilt, beide
Teilzeilen werden umgekehrt mit x verkettet.
Wenn x eine Zahl ist, wird nach dem x-ten Buchstaben geteilt.

--x[[a][b]] :

Alle aktuellen Zeilen werden gelöscht.
Ist ein Zeichen a kodiert, wird die Vorzeile auch gelöscht;
ist ein Zeichen b kodiert, wird die Nachzeile auch gelöscht.

--X : Alle Zeilen werden gelöscht.

--lines[z] :

Zu einer Zeile zusammenfassen.

z=';': Die vorhandenen Wörter sind durch das Semikolon getrennt;
alle Wörter werden in eine Zeile genommen und sind dort
durch das Semikolon getrennt.

z='' : Alle Wörter sind durch Leerzeichen oder Semikolon oder
Tabulatorzeichen getrennt; sie werden in eine Zeile
genommen und sind dort durch das Leerzeichen getrennt.

z='.': Die aktuelle Datei besteht aus Blöcken, die durch
Leerzeilen getrennt sind; aus den Zeilen jedes Blockes
wird eine Zeile gemacht; wenn der Block aus n
Zeilen bestand, wird der gebildeten Zeile n. vorangestellt.

sonst: Die aktuelle Datei besteht aus Blöcken, die jeweils mit dem
auf z stehenden String beginnen. Die Zeilen jeden Blockes

werden mit dem neue-Zeile-Zeichen zu einer gemacht.
Nach dieser Operation macht es Sinn, Parameter mit
Neue-Zeile-Zeichen einzugeben.

--rows[tr] :

Die Wörter werden zeilenweise aufgeteilt; Trennzeichen sind
das Semicolon und das Leerzeichen, falls auf tr keine kodiert sind.

--end : Der Aufruf wird beendet.

--Sx[<zahl[.]|>zahl[.]|'u'|'z'] :

Sortieren der Zeilen ohne Beachtung von Groß- und Kleinschreibung;

x: Sortierungsvergleich bis zum Zeichen x;

jede Zeile muss das Zeichen x enthalten. Enthält die 1. aktuelle
Zeile nicht das Zeichen x, wird diese Option ignoriert.

u: Übereinstimmende Zeilen werden nur einmal ausgegeben.

<|>: Danach muss eine Zahl zahl folgen; es werden nur solche Zeilen
ausgegeben, von denen es mindestens zahl Übereinstimmungen gibt
(Fall >) bzw weniger als zahl (Fall <).

Die Anzahl der Übereinstimmungen wird, durch Doppelpunkt
getrennt, an jede Zeile angehängt, falls '.' codiert ist

z: zufällige Anordnung der Zeilen.

Beispiel für die Anwendung:

Aus einem Indexregister in dem Buch buch.tex sollen alle

Einträge mit weniger als 13 Verweisen ausgegeben werden.

Vor dem Aufruf von makeindex verwende man den Aufruf

```
# buch.idx "--S|<13"
```

--t : Umwandlung in eine Tabelle; jede Zeile wird ein Eintrag.

Ein Eintrag darf höchstens 79 Zeichen haben.

Falls unmittelbar nach einer Option mit -- keine Option folgt, wird "-I"
eingefügt.

Nach den folgenden +Optionen stehen im Aufruf meist die Wörter maske und
wort. Als Standard wird maske bzw. wort an einer beliebigen Stelle
innerhalb der Zeile gesucht.

+A[A|=][z] anfangswort :

Die folgenden Operationen beginnen nach der ersten Zeile mit
anfangswort; die Zeilen bis einschließlich anfangswort bleiben
erhalten. Im Falle A wird auf Übereinstimmung am Zeilenanfang,
bei = auf Übereinstimmung mit der Zeile getestet. Falls keines
dieser Zeichen auftritt, wird anfangswort in einer Zeile gesucht.
Falls anfangswort nicht kodiert ist, wird die nächste Leerzeile
gesucht.

Falls ein Zeichen z codiert ist, wird bis zur letzten, gefundenen
Zeile vorgesetzt.

+E[A|=][z] endewort :

Die folgenden Operationen enden vor der ersten Zeile mit endewort; die Zeilen ab endewort bleiben erhalten. Falls endewort nicht kodiert ist, wird die nächste Leerzeile gesucht.

Falls ein Zeichen z codiert ist, wird bis zur letzten, gefundenen Zeile vorgesetzt.

+delete maske wort|"" :

Der mit maske beginnende und mit wort endende Block wird gelöscht.

+f[A|E|.][E|.][maske [wort]] :

In jeder aktuellen Zeile wird der String zwischen maske und wort ausgegeben; ist maske das Leerwort, beginnt der String am Zeilenanfang; ist wort das Leerwort (oder nicht angegeben), endet der String am Zeilenende.

Für den Fall E gilt: Verschieben von maske bzw. wort zum Zeilenende hin.

+g[[|.][.]] [maske [wort]] :

In jeder aktuellen Zeile wird der String zwischen maske und wort gelöscht, falls maske gefunden wird.

Der erste Punkt - falls kodiert -- bedeutet, dass maske nicht gelöscht wird; entsprechend der zweite Punkt für wort.

+e[A|E|.][E|.][maske [wort]] :

In jeder aktuellen Zeile wird der String zwischen maske und wort ausgegeben; ist maske das Leerwort, beginnt der String am Zeilenanfang; ist wort das Leerwort (oder nicht angegeben), endet der String am Zeilenende.

Für den Fall E gilt: Verschieben von maske bzw. wort zum Zeilenende hin. Wenn maske nicht gefunden wurde, bleibt die betreffende Zeile erhalten.

+if_empty : Wenn keine aktuelle Zeile existiert, wird fortgefahren; andernfalls wird bis zur nächsten Leeranweisung ++ vorgerückt.

+if_notempty : Negation von +if_empty.

+if_defined var : Wenn die Umgebungsvariable var existiert, wird fortgefahren; andernfalls wird zur nächsten Leeranweisung ++ vorgerückt.

+if_notdefined var : Negation von +if_defined.

+if_exist filename : Wenn eine Datei mit dem angegebenen Namen existiert, wird fortgefahren; andernfalls wird zur nächsten Leeranweisung ++ vorgerückt.

+if_notexist filename : Negation von +if_exist.

+q [maske] [wort] :

Trennzeichenersetzung: Alle auf maske stehenden Zeichen werden als Trennzeichen interpretiert; alle Trennzeichen am Anfang und Ende jeder Zeile werden gelöscht; ihr kompaktes Auftreten innerhalb einer Zeile wird durch das Zeichen auf wort ersetzt. Standard: Löschen von Leerzeichen am Anfang und Ende jeder Zeile sowie von mehrfachen Leerzeichen innerhalb einer Zeile.


```

+r[A|E|=|ep] maske wort :
    Alle Zeilen werden ausgegeben; in den Zeilen mit maske wird jedes
    Vorkommen von maske durch wort ersetzt; im Falle A wird nur am An-
    fang, im Falle E nur am Ende ersetzt. Im Falle ep wird in jeder Zeile
    nur das erste Vorkommen bearbeitet.
+R[A][z] maske wort :
    Beim ersten Vorkommen von maske wird die betreffende Zeile
    durch wort ersetzt; falls keine gefunden wurde, wird eine Zeile
    mit wort angehängt, sofern kein Zeichen z codiert ist.
++ : Leeraanweisung.
++Azahl :
    Der Operationsanfang wird um zahl Zeilen vorgesetzt;
++dwort :
    Die vorhandenen Zeilen bilden eine aufsteigend sortierte Liste.
    Es werden alle Zeilen ausgewählt die am Anfang maximale
    Übereinstimmung mit wort haben. Die Umgebungsvariable .start
    erhält als Wert das gemeinsame Anfangswort.
++Ezahl :
    Es sind zahl Zeilen ab Operationsanfang aktuell;
++tabzahl :
    Umwandlung von Tabulator-Zeichen in Leerzeichen;
    auf zahl steht die Anzahl der zu verwendenden Leerzeichen.

```

Ein Parametersatz darf mehrfach auftreten (innere Pipe); das letzte Ergebnis ist Eingabe für die nächste Operation; neue Eingaben werden an das letzte Ergebnis angehängt bzw. diesem vorangestellt. Insbesondere können mittels der -O-Option Zwischenausgaben erstellt und mittels der -I-Option Dateien an ein Zwischenergebnis angehängt werden.

2. A short documentation of the auxiliary programs

Auxiliary programs are collected in the program `#.c` (obtainable for free); this is available as `#.exe` in the distribution. Here, the communication between program and user via environment variables is essential which is not supported by the script language as desired. This functionality is achieved by the following method: If an option of the form `:-var` occurs in the call of a program, then the output of the program is assigned as one line to the environment variable `var` and this assignment is written into that `cmd`-file whose name is stored in the environment variable `..` (the value of this environment variable has the form `call aa.cmd`); after executing this file, the output is available in the current command.

Example:

```

set ..=call aa.cmd
# cc.txt -A1 -E1 -:bb & %..%

```

In this case the environment variable `bb` has the value of the second line of the file `cc.txt`.

Pay attention that the arising lines are not too long.

Transformation of a text file into a LaTeX document:

```
#txt2ltx inputname [outputname]
```

The output file is - by application of the possibilities available in this system - usable as a stand-alone LaTeX document as well as an input file for other documents without any modification.

#####

Write each parameter as a line into a file:

```
#e f output_filename parm1 ...
```

Append each parameter as a line to a file:

```
#e fa output-filename parm1 ...
```

Translation from ansi to unicode:

```
#e ansi_uni input_file output_file
```

Here, reg-files obtain the correct registry identifier.

Translation from unicode to ansi:

```
#e uni_ansi input_file output_file
```

In reg-files, the registry identifier is removed.

Standard error output and wait for input:

```
#e parm -|+:var
```

parm contains (enter=...) and input to environment variable var or standard output otherwise; if parm contains

the string (enter=), then this one is not displayed.

The input of < leads to the output of the empty word.

In case of +:var the output is exit, if the empty word is the value.

Output of a number:

```
decimal: #e number parm -:var
```

```
hexadecimal: #e hex parm -:var
```

The parameter parm contains at the beginning a number which is stored in var.

Output of a real operation:

```
#e : var=1.operand[+|-|*|/]2.operand
```

The operands are numbers; op will be missing.

Default output:

```
#e parm1 ...
```

#####

Call:

```
# {input [anfang] [ende] operation [output]}
```

```
input      : -I {inputfile| ""}|-L stringlist1
anfang     : {-|+}A[+[A|E|=]] startword
ende       : {-|+}E[+[A|E|=]] endword
operation  : {-o stringliste2} | {+o [mask] [word]} | --o| ++o
output     : -O[+] outputfile
```

From a file or a list of words, lines or words can be modified selected and output formatted in different manners.

`-I[A|E] {inputfile | ""} :`

If inputfile is an empty word or not given, then the standard input is used. The files will be read in the given order and treated internally as one file. In the case A, the data are inserted at the beginning; in the case of E, at the end; default is the end.

`-L[A|E|v] stringlist1 :`

Every word of stringlist1 is taken as one line. The data are inserted at the beginning (case A) or at the end (case E); by default at the end.

`-[:|;|~]var :` Output to the environment variable var.

`{-|+}A[A|E|=] startword :`

The subsequent operations begin after the first line containing start word; in the case of -A, the preceding lines are deleted, in the case of +A, they remain. The start option may end with one of the characters A, E, or =; in the case of A, it is checked for coincidence at the beginning of a line, for E at the end of a line, and for =, the whole line must coincide. If none of these characters is given, start word is searched within a line. This kind of option specification is also used for further options.

`{-|+}E[A|E|=] endword :`

The subsequent operations end before the first line with end word; in the case of -E, the lines thereafter will be deleted, for +E, they remain.

The lines which are included with A and E are the current lines for the following operations.

`-e[A|E|=][t] stringlist2 :`

Gives all lines with a word of stringlist2.

`-O[+] [outputfile] :`

Output of a file. In the case of option + the output is appended; If the outputfile is missing in the current call, the last input file is used. Please note, that all rows are

outputed into the last inputfile, if outputfile is missing; otherwise only everything in the operation range (between beginning and end) is output.

-set maske ... :
 maske is an environment variable; the first founded row starting with "set maske=" will be replaced with "set maske=value".

-v[A|E|=] stringlist2 :
 Yields all lines without a word from stringlist2.

-? "text of request" :
 Line selection.
 The current lines, increasingly numbered, appear on the screen together with the demand to make a selection. For selecting, numbers may be entered in the form i i-k with repetition. The corresponding words are the new lines; in the case of the input <, the value is the empty string. In the text of request, a pre-selection may be given that is output in the case that just <enter> is pressed; the preselection must be enclosed by (enter= and); in the case of (enter=all), all lines will be output. If no pre-selection is given, the input <enter> yields the empty string as the value. The length of a line must not exceed 70 characters. Example for the application:

```
# -L %files% -? "=="> Please select: " -::file & call %set%
```

The file names (given with the variable files) are given for selection; after the execution, the environment variable file contains the name selected. In the case of the call

```
# -? "Select, please: " %files% -::file & call %set%
```

the variable file obtains all names selected.

The following --options work without other parameters.

-- : The beginning and end of the operation are
 set to the actual beginning and end, respectively.

--A[zahl|z] :
 The operation started after rows zahl; all rows before are deleted.

--Ezahl :
 The operation ended zahl lines after the starting line; the following lines will be deleted.

--x[[a][b]] :
 The actual rows will be deleted;

--lines[z] :

Each block with started with the character z and ended with a blanc line will be considered as a whole line.

--rows : Each word into a single line.

--x : all actual lines are removed.

--S[u] :
 Sort of the lines case-insensetive;
 u: multiples are output only once;

--t : Transformation into a table; each line becomes an entry.
 An entry may have 79 characters at most.

Behind the following +options, there are mostly the words mask and word in the call. The respective operations are executed only if no column structure is active. By default, mask and word are searched at an arbitrary position within a line.

+:var : the same as -:var, but the output is exit,
 if the value is the empty word.

+f[A|E|.][E|.][mask [word]] :
 In every current line, the string between mask and word is output; if mask is empty, the string starts at the beginning of the line; if word is empty (or not given), then the string ends at the end of the line.
 For the case A: mask is the binning of the line.
 For case E: mask resp. word is moved to the right.

+q mask [word] :
 Replacement of separators: All characters contained in mask are interpreted as separators; all separators at the beginning or end of a line are deleted; their compact occurrence within a line will be replaced by *word.
 Default: Deletion of blanks at the beginning and end of each line as well as multiple occurrences of blanks within a line.

+r[A|E][n|t] mask word :
 All lines are output; in the lines with mask, each occurrence of mask is replaced by word; in the case of A, the replacement is done only at the beginning of a line, in the case of E, only at the end. The options n, t, respectively, are related to the case A: The replacement will be started with a newline letter (n) or a tabulator (t).

+R[A|E][n|t] mask word : same as +r; but only for the first occurence.

++Anumb :
 The operations will be started numb rows later.

++Ezahl :

The end of operations will be started numb rows later.

A set of parameters may occur more than once (internal pipe); the last result is input for the next operation; new inputs are appended to the last result or inserted before. Especially, by the -O option, you can output intermediate results and, by the -I option, files can be added to an intermediate result. Also the option -: may occur repeated.