```
/* flow.the:  (patterned after KEDIT's FLOW.)

This THE macro aligns two or more lines of a text-type file being edited
(such as a NOTE). It tries to place as many words as possible on a line,
within the right margin as defined by SET MARGIN.
    Paragraph breaks are respected, and can be either style - indented
with space or tab characters, as this paragraph itself is, or a
completely empty line, as the following paragraph is. A side effect of
this is that a set of lines which are indented are protected from
flowing; this is useful for examples.

This paragraph is the other style. Note that text containing tab
characters for any purpose other than to start a paragraph or indent an
example, may not appear as you intended after flowing.

If a word is encountered that is longer than the margin, it is left by
itself on a line, without truncation. This is most frequently
encountered with URLs.

USE:

           .--*---------.
  >>--flow--+------------+--><
           '--|target|--'


where |target| is a standard editor target defining the first line not
to be flowed. If it is not specified, the default is to flow from the
current line to the end of the file. Typically, the alignment process
will result in there being a different number of lines in the block than
there were before alignment. This will not always be true.

UNIQUE CAPABILITY OF THIS PROGRAM:

This macro can, unlike other parts of THE, shorten lines. If you SET
MARGIN to a value shorter than some of the lines in your file, they will
be handled correctly by this macro. Elsewhere in THE, XEDIT, and KEDIT,
results are unpredictable and can involve unintended truncation.

These comments have been aligned with this program.

PLATFORM:

This program is intended to run on any THE Version 2.7 or greater. It is
not intended for KEDIT, which has this capability built in.

LICENSE:

flow.the, a Hessling Editor macro for flowing text
Copyright (C) 2000 Roger D. Deschner

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.
```

```
The GNU General Public License is available at:
http://www.gnu.org/licenses/gpl.html

AUTHOR, BUG REPORTING PROCEDURE:

This program was written by Roger Deschner, University of Illinois at
Chicago. Send comments, including bug reports, to rogerd@uic.edu.

MODIFICATION HISTORY:

2000-12-11 - Restore original current line position. Leading tab can
             also start a new paragraph. Fix problem with upward targets.
2000-12-04 - Avoid use of SET POINT, which can overlay existing points
             set by the user.
2000-12-02 - Roger Deschner - Partial rewrite to fix bug where if a
             word is encountered that is longer than an output line,
             the remainder of the range is discarded. This bug has
             existed since the earliest CMS versions. Also, move temp
             file to Rexx stem, avoiding portability issues with external
             scratch files or the stack.
2000-07-18 - Roger Deschner - port to The Hessling Editor
1997-01-07 - Roger Deschner - Convert to regular command FLOW
1989-10-24 - Roger Deschner - Protect from LINEND character
1988-02-14 - Roger Deschner - Allow lines to be shortened; use PUTD
1988-01-02 - Roger Deschner - Replace call to "JOIN", for performance
1986-11-16 - Roger Deschner - Original version

*/
PARSE ARG targ
targ = STRIP(targ)
IF (targ = '') THEN targ = '*'

tabchar = '09'x

/* Turn off the linend setting. It gets in our way. */
'COMMAND EXTRACT /LINEND'
'COMMAND SET LINEND OFF'

/* Demark the start and end of our range */
'COMMAND EXTRACT /LINE'
oldline = line.1
justify1 = line.1
/* Is Line Number 0 (i.e. "Top Of File") the current line? */
IF (justify1 = 0) THEN justify1 = 1 /* We can't flow Line 0. */
/* Find end of range. */
'COMMAND LOCATE' targ
'COMMAND EXTRACT /LINE'
justify2 = line.1

/* Upwards target? Reverse start and end. Note adding one to top because
it is the end of the range (i.e. moving it down one line) and also to
the bottom (i.e. moving it it down one line) because it is the current
line - the start of the range. */
IF (justify1 > justify2) THEN DO
  /* Is the current line the *** Bottom of File *** line? If so, go up
  one line (i.e. subtract 1) to avoid flowing it. */
  'COMMAND EXTRACT /SIZE'
  IF (justify1 > size.1) THEN justify1 = justify1 - 1
  /* Swap */
  xyzzy = justify1
  justify1 = justify2 + 1
  justify2 = xyzzy + 1
```

```
END

/* Do it to it */

/* Save the lines in the range into a rexx stem */
'COMMAND LOCATE :'justify1
i = 0
DO FOREVER
  i = i + 1
  'COMMAND EXTRACT /CURLINE'
  inbuf.i = curline.3
  'COMMAND DOWN 1'
  'COMMAND EXTRACT /LINE'
  IF (line.1 >= justify2) THEN LEAVE
END
inbuf.0 = i

/* Delete the lines in the range */
'COMMAND LOCATE :'justify1
'COMMAND DELETE :'justify2

'COMMAND UP 1'
'COMMAND EXTRACT /MARGINS' /* We are interested in margins.2 */

rotbuf = ''
DO j = 1 TO inbuf.0 /* The magic part lives inside this loop. */
  ibuf = inbuf.j
  /* Paragraph break, either kind */
  char1 = SUBSTR(ibuf,1,1)
  IF ((char1 = ' ') | (char1 = tabchar)) THEN DO
    IF (rotbuf ^= ' ') THEN DO    /* Anything left in buffer? */
      'COMMAND INPUT' rotbuf    /* put it out */
      rotbuf = ''
    END
  END /* of IF (SUBSTR(ibuf,1,1) = ' ') THEN DO */
  IF (ibuf = ' ') THEN 'COMMAND INPUT   '    /* Blank line */
  ELSE DO                       /* duit tuit */
    /* concatenate the new stuff */
    IF (rotbuf = '') THEN rotbuf = STRIP(ibuf,'T')
                     ELSE rotbuf = rotbuf STRIP(ibuf,'T')
    SELECT
      WHEN (LENGTH(rotbuf) = margins.2) THEN DO /* perfect fit */
        'COMMAND INPUT' rotbuf
        rotbuf = ''
      END
      WHEN (LENGTH(rotbuf) > margins.2) THEN DO /* more than enough */
        DO FOREVER
          /* Find last blank, starting at margins.2+1 working backwards */
          i = LASTPOS(' ',SUBSTR(rotbuf,1,margins.2+1))
          IF (i > 0) THEN DO
            'COMMAND INPUT' SUBSTR(rotbuf,1,i-1)
            rotbuf = STRIP(SUBSTR(rotbuf,i),'B')
            IF (LENGTH(rotbuf) < margins.2) THEN LEAVE /* Split enough? */
          END
          ELSE DO /*OneWordLineThatIsTooLongToSplit*/
            'COMMAND EXTRACT /LINE'
            'COMMAND EMSG Word encountered in line' line.1,
              'that is longer than margin.'
            /* Just write out what we've got. */
            'COMMAND INPUT' rotbuf
            rotbuf = ''
            /* And resume with the next input line. */
```

```
            LEAVE
          END /* of ELSE DO /*OneWordLineThatIsTooLongToSplit*/ */
        END /* of DO FOREVER */
      END /* of WHEN (LENGTH(rotbuf) > margins.2) THEN DO */
      OTHERWISE NOP /* not long enough - read another line */
    END /* of SELECT */
  END /* of ELSE DO                    /* duit tuit */ */
END /* of DO j = 1 TO inbuf.0*/
IF (rotbuf ^= ' ') THEN DO        /* Anything left in buffer? */
  'COMMAND INPUT' rotbuf          /* put it out */
  rotbuf = ''
END

/* Clean up our toys and go home */
'COMMAND LOCATE :'oldline
'COMMAND SET LINEND' linend.1 linend.2
EXIT
```