NAME

HB2HSH − Hash a string of Hebrew text.

SYNOPSIS

CALL HB2HSH(MXHSTR,HSTR,MXHASH, NCH,CHASH,VHASH,RC,K)

MXHSTR	is the INTEGER*4 dimensioned length of HSTR
HSTR(MXHSTR)	is the CHARACTER*1 Hebrew string to hash
MXHASH	is the INTEGER*4 dimensioned length of CHASH and VHASH
NCH	is the INTEGER*1 number of Hebrew characters in HSTR
CHASH(MXHASH)	is the INTEGER*1 vector of letter numbers
VHASH(MXHASH)	is the INTEGER*1 vector of vowel numbers
RC	is the INTEGER*4 return code
K	is the INTEGER*4 position in HSTR where an error occurred

DESCRIPTION

This routine parses a line HSTR of LaTeX Hebrew typesetting commands and returns in CHASH and VHASH the letter and vowel numbers corresponding to the letters and vowels in the string. On average this compresses the storage required for each character from 13 bytes to 2 bytes or about 15% of the full-text size.

The routine also recognizes the names of trope symbols. When Hebrew is typeset using the macros in /bin/hebrew.tex, a trope symbol is overlaid on the previous character. Thus, for example, \dagesh{x}{y} produces a dagesh at the coordinates [x,y] in the preceding character. The names of the trope symbols follow the names of the vowels, so this routine gives the VHASH entry for a trope symbol its index in that list. The CHASH entry for a trope symbol consists of two 4-bit integers corresponding to its x and y coordinate values. Thus, for example, \dagesh{13,7} yields a CHASH entry of Z'C7'.

The routine assumes that a blank terminates the Hebrew typesetting command sequence, so the only way to embed a blank in a Hebrew string is with \hebrew{space} or \hebspc{l}{r}.

If there are fewer than MXHASH characters in the Hebrew string represented by the typesetting commands in HSTR, then CHASH and VHASH will have trailing 00 bytes, but no hash ever has embedded 00 bytes.

SEE ALSO

HSH2HB expands the hash of a Hebrew string back into the string.

DIAGNOSTICS

On output these are the possible RC values:

0	all went well
1	HSTR has zero length, or MXHSTR=0
2	first character of HSTR is not backslash
3	premature end of HSTR
4	more than 10 characters in a vowel or trope name
5	vowel or trope name not recognized
6	more than MXHASH vowel or trope specifications in HSTR
7	more than 10 characters in a letter name
8	letter name not recognized
9	bad coordinate in a trope specification
10	more than MXHASH letters or coordinate pairs in HSTR
11	character after a letter is not backslash or space

NOTES

The document "Homebrew Hebrew" describes Hebrew typesetting.

LINKAGE

gfortran source.f −L${HOME}/lib −lmisc

AUTHOR

Michael Kupferschmid

EXAMPLE

```
CHARACTER*25 HSTR/'\hebrew{hay}\lcholam{caf}'/
INTEGER*1 NCH,CHASH(3),VHASH(3)
INTEGER*4 RC
CALL HB2HSH(25,HSTR,3, NCH,CHASH,VHASH,RC,K)
PRINT *,RC,NCH,(VHASH(I),CHASH(I),I=1,3)
STOP
END
```

This example produced the following output:

```
unix[1] a.out
          0    2   13   17    1    6    0    0
unix[2]
```

The return code of 0 indicates that the translation was successful. Two Hebrew characters were found in the string. The codes "13 17" represent an lcholam vowel on a caf, and the codes "1 6" represent no vowel on a hay (this routine assigns a number to each letter and a number to each vowel). The "0 0" shows that there were not enough characters to fill up VHASH and CHASH. The word "co" would be written [blank hay caf] right justified in the 3 character positions and right to left, while the hash codes are stored left justified in the opposite order. Thus the hash codes of the letters are stored in mirror order to the Hebrew letters they represent (this facilitates the lexicographical sorting of hash code vectors).