Homebrew Hebrew

by Michael Kupferschmid

August 22, 2023

1 Introduction

As a student and teacher of Hebrew I often need to prepare documents that contain text in that alphabet. Fortunately, I was able to download free files defining the `redis` family of fonts for LaTeX 2ε, and place them in the directory `${HOME}/texmf/fonts/`. Using those fonts I have constructed a software system that lets me easily typeset letters, words, sentences, paragraphs, pages, or whole documents consisting of pointed Hebrew text.

2 One Consonant at a Time

Here is a LaTeX 2ε document that uses one of the `redis` fonts.

```
% example1.tex
\documentclass[12pt]{article}
\font\ivrit=redis12
\begin{document}
\ivrit{‘}
\end{document}
```

When it is compiled and the resulting `.dvi` file is displayed, like this,

```
unix[1] latex example1.tex
unix[2] xdvi example1.dvi
```

the result is a single page containing the character א. Longer strings can obviously be constructed one character at a time; `\ivrit{m}\ivrit{e}\ivrit{l}\ivrit{y}` produces שלום.

This approach is hard to use for several reasons. One must remember the encoding of the consonants (e.g., y=ש), and no way is provided to typeset vowels (which in Hebrew are marks above and below the consonants).

3 One Letter at a Time

It would be more convenient to set Hebrew characters by using their names rather than numerical codes, and for teaching it is important to be able to print vowel points. To accomplish both objectives I wrote a set of LaTeX 2ε macros and collected them in a file named `bin/hebrew.tex`, which can be `\included` at the beginning of a document. The example listed at the top of the next page illustrates how to use `hebrew.tex`. The line numbers on the left are present only so that I can refer to them, and are not part of the document.

The file `${HOME}/bin/hebrew.tex` must be `\input` 3 in the preamble, and `\setivrit` must be used 8 to select a font size before any other `hebrew.tex` command is used. The recognized font sizes are `7`, `8`, `s8` (slanting), `9`, `s9`, `10`, `bx10` (thick), `s10`, `12`, `s12`, `17`, `20`, `24`, `29`, and `35`.

This LaTeX 2ε document generates the output boxed at the bottom of the page, which reproduces page v of *The First Hebrew Primer* (the Hebrew could instead be translated “In the name of Heaven. . . He will make peace upon us and upon all Israel. . . ”).

```
% example2.tex
\documentclass[12pt]{article}
\input{/home/mike/bin/hebrew} % set up to point Hebrew letters
\pagestyle{empty}
\begin{document}

\Large
\setivrit{17}                 % initialize and set a font size
\centerline{%
\hebrew{endmem}\chiriq{yod}\patach{mem}\qamats{shin}~%
\hebrew{endmem}\tsere{shin}\sheva{lamed}%
}

\vspace{3ex}
\normalsize
\setivrit{12}                 % reset the font size
\centerline{%
\ldots~\hebrew{lamed}\tsere{aleph}\qamats{resh}\sheva{sin}\chiriq{yod}~%
\hebrew{lamed}\qamats{kaf}~%
\hebrew{lamed}\patach{ayin}\sheva{vav}~%
\hebrew{oovav}\hebrew{nun}\hebrew{yod}\tsere{lamed}\qamats{ayin}~%
\hebrew{endmem}\lcholam{vav}\hebrew{lamed}\qamats{shin}~%
\segol{sin}\halfpatach{ayin}\patach{yod}~%
\hebrew{aleph}\hebrew{oovav}\hebrew{hay}~\ldots%
}

\vspace{3ex}
\centerline{\rule{1in}{0.1pt}}

\vspace{5ex}
\centerline{\large\em for the G--d of Heaven}

\vspace{3ex}
\centerline{\ldots may He make peace for us and for all Israel \ldots}
\end{document}
```

לְשֵׁם שָׁמַיִם

. . . הוּא יַעֲשֶׂה שָׁלוֹם עָלֵינוּ וְעַל כָּל יִשְׂרָאֵל . . .

---

for the G–d of Heaven

. . . may He make peace for us and for all Israel . . .

name	consonant		
aleph	א		01
bet	ב	b	02
vet	ב	v	03
gimel	ג	g	04
dalet	ד	d	05
hay	ה	h	06
vav	ו	v	07
ohvav	וֹ	oh	08
oovav	וּ	oo	09
zayin	ז	z	10
khet	ח	kh	11
tet	ט	t	12
yod	י	y	13
endcoph	ך	ch	14
coph	כ	ch	15
endcaf	ךָ	c	16
caf	כ	c	17
lamed	ל	l	18
endmem	ם	m	19
mem	מ	m	20
endnun	ן	n	21
nun	נ	n	22
samech	ס	s	23
ayin	ע		24
endfay	ף	f	25
fay	פ	f	26
endpay	ף	p	27
pay	פ	p	28
endtsade	ץ	tz	29
tsade	צ	tz	30
kuf	ק	k	31
resh	ר	r	32
shin	שׁ	sh	33
sin	שׂ	s	34
sn	ש		35
taf	ת	t	36
dash	־		37
space			38
hash	״		39
geresh	׳		40

Each Hebrew letter in Example 2 is again set using a separate command; thus 10 `\hebrew{endmem}` sets an ending mem without any vowel and `\chiriq{yod}` sets a yod with a chiriq vowel.

The names, glyphs, and transliterations of the consonants are given in the table on the left. The transliteration `c` is a hard c, and sometimes I use `ts` instead of `tz` for צ or ץ. In Hebrew some compound words contain a `dash` or `space`. The hash mark ״ is used to show that letters have been elided from acronyms such as ק״ת. The geresh is used in writing numbers with Hebrew consonants. Later I will explain the numerical codes appearing in the rightmost column. By default the Hebrew consonants are printed in bold. To turn bold off use `\renewcommand{\hbold}[1]{#1}`; to turn it back on use `\renewcommand{\hbold}[1]{\pmb{#1}}`. To bold a Hebrew string you can make it the argument of `\hbf{}` (this is translated into `renewcommand` commands by the `hebunt.f` program described later).

The names of the vowels are listed below; here □ represents any letter (Hebrew or not). Later I will explain the numerical codes.

name	vowel	
`\hebrew{`letter`}`	□	01
`\chiriq{`letter`}`	□ִ	02
`\tsere{`letter`}`	□ֵ	03
`\segol{`letter`}`	□ֶ	04
`\halfsegol{`letter`}`	□ֱ	05
`\patach{`letter`}`	□ַ	06
`\halfpatach{`letter`}`	□ֲ	07
`\qamats{`letter`}`	□ָ	08
`\halfqamats{`letter`}`	□ֳ	09
`\awe{`letter`}`	□ֻ	10
`\qubbuts{`letter`}`	□ֻ	11
`\sheva{`letter`}`	□ְ	12
`\lcholam{`letter`}`	□ֹ	13
`\rcholam{`letter`}`	□ֹ	14
`\dagesh{x}{y}`	⊡	15
`\meteg{x}{y}`	□ֽ	16
`\hebspc{L}{R}`	□□	17

Vowels appear above and below the consonants, so a line of Hebrew takes more vertical space than a line of English at the same font size. If you will set more than one line, you might want to adjust the baselineskip with `\setlength{\baselineskip}{2.6\hex}`. To include Hebrew in a part command like `\section{}` you must `\protect` each vowel name.

4 Movable Points

Each sounding vowel (having code `01-14`) in the table above is fixed in its location relative to the letter on which it appears. The other "vowels" (codes `15-17`) each have two arguments and are coded *after* the consonant to which they are attached. Each argument is a positive integer in [0,15]. When the arguments are denoted x and y the first tells how far the glyph should be from the right edge of the letter and the second tells how far it should be from the bottom edge of the letter; 0 corresponds to the right edge or the bottom and 15 to the left edge or the top.

Consonants for which a point makes a difference in the sound (בּ, וּ, כּ, גּ, פּ, תּ, שׁ) are named separately with and without the point. If you want to set both a dagesh and a vowel on another consonant, you can use a construct like `\patach{gimel}\dagesh{10}{5}` which yields גַּ.

The `\hebspc` command inserts horizontal space to move the letter after it left (if L > 0) or right (if R > 0). This is useful for kerning two letters together as shown in the example.

The file `/home/mike/bin/hebrew.tex` contains `\usepackage` commands for the LaTeX 2_ε packages `ifthen`, `amsmath`, and `graphicx`, all of which provide functionality that is needed by some of its commands.

5 Flush Right Text and Punctuation

Hebrew is written from right to left, so lines of text begin at the right margin and if unadjusted are ragged on the left. To simplify making lines of Hebrew text flush right, `hebrew.tex` includes the macro `\hebline{}`, whose use is illustrated in this example.

```
 1 % example3.tex
 2 \documentclass[12pt]{article}
 3 \input{/home/mike/bin/hebrew}
 4 \renewcommand{\hbold}[1]{#1}
 5 \pagestyle{empty}
 6 \begin{document}
 7
 8 \setivrit{12}
 9 \hebline{\hebpnk{,}\hebrew{endmem}\segol{coph}\hebrew{yod}\tsere{lamed}\halfpatach{ayin}
10         \hebrew{endmem}\hebrew{ohvav}\hebrew{lamed}\qamats{shin}}
11 \hebline{\hebpnk{,}\hebrew{taf}\tsere{resh}\qamats{shin}\patach{hay}
12         \hebrew{dash}\hebrew{yod}\tsere{coph}\halfpatach{aleph}\sheva{lamed}\patach{mem}}
13 \hebline{\hebrew{endnun}\hebrew{ohvav}\hebrew{yod}\sheva{lamed}\segol{ayin}
14         \hebrew{dash}\hebrew{yod}\tsere{coph}\halfpatach{aleph}\sheva{lamed}\patach{mem}}
15
16 \end{document}
```

This LaTeX 2ε document produces the output below, in which the first three phrases of the song *Shalom Aleikhem* (page 722 in Siddur Sim Shalom or page 375 in the Sacks Koren Shalem siddur) are right-justified on separate lines.

שָׁלוֹם עֲלֵיכֶם،
מַלְאֲכֵי־ הַשָּׁרֶת،
מַלְאֲכֵי־ עֶלְיוֹן

To print a right-to-left comma at the end of the first two lines I used the macro `\hebpnk{}`, which produces the mirror image of its argument.

6 Words and Transliterations

One thing that makes Hebrew hard to learn is that many vowel forms, word-pair forms, and words having prefixes and suffixes do not appear in printed dictionaries. To facilitate my study of vocabulary I constructed my own dictionary, which helps me look up and remember these words. The dictionary is a plain text file that I named `Hebrew/milon.dat` (מִלּוֹן is the Hebrew word for a dictionary). Each line of `milon.dat` contains a single Hebrew word, its translation, and its transliteration; the line that contains the word מִלּוֹן looks like this:

```
\hebrew{endnun}\hebrew{ohvav}\hebrew{lamed}\chiriq{mem} (a) dictionary % milon
```

First comes the string of four LaTeX 2ε commands that set the letters of the word in right-to-left order, then the translation, and finally, separated by a percent sign, the transliteration.

It is convenient for the lines of `milon.dat` to be in arbitrary order, so that they can be arranged in groups that are related by meaning, or by sound, or by the occasion on which they were added to the dictionary. To search for words it is better to have a file that is in alphabetical order, so I wrote a program called `hebsort.f` (see its `man` page) to produce from `milon.dat` a file `Utility/hebrew.hsh` that has its lines arranged in alphabetical order of the transliterations. I call this file the hashed dictionary. Here is the `hebrew.hsh` line for מִלּוֹן along with the lines that immediately precede and follow it (the vertical ellipses indicate that there are other lines before and after these).

```
:
milmayl             4 02140C120314011200000000000000000000000000000000 mumble
milon               4 021401120108011500000000000000000000000000000000 (a) dictionary
milooy              5 0214011201091102010D0000000000000000000000000000 filling
:
```

The separate (decimal) numbers `4`, `4`, and `5` show that these words have respectively 4, 4, and 5 Hebrew letters. A letter is a consonant with a vowel (in this scheme `\hebrew` is just a "vowel" that has no points). To save space this file stores each LaTeX 2ε command for the letters of a word as (hexadecimal) numbers, using the codes listed in the tables of §2. The numbers or hash code describing the Hebrew for `milon`, which are `0214011201080115`, translate back to LaTeX 2ε commands like this:

$\mathbf{02}_{16} = 2\{\mathbf{14}_{16} = 20\}$ $\mathbf{01}_{16} = 1\{\mathbf{12}_{16} = 18\}$ $\mathbf{01}_{16} = 1\{\mathbf{08}_{16} = 8\}$ $\mathbf{01}_{16} = 1\{\mathbf{15}_{16} = 21\}$
`\chiriq{mem}` `\hebrew{lamed}` `\hebrew{ohvav}` `\hebrew{endnun}`

so in this file the Hebrew letters read from left to right (storing the letters in this order facilitates sorting the dictionary into alephbetical order by the Hebrew, which `hebsort.f` can also do). The `hebsort.f` program uses a subroutine named `HB2HSH` (see its `man` page) to translate a string of LaTeX 2_ε commands into their corresponding hash codes. In the hashed dictionary a transliteration can be up to 18 characters long, the hash code for the LaTeX 2_ε commands that set the Hebrew can represent up to 12 Hebrew letters, and the English translation can be up to 80 characters long. A transliteration that ends in `A` denotes a word that is Aramaic and one that ends in `Y` denotes a word that is Yiddish (an Aramaic or Yiddish word is included only if it differs from the Hebrew word having the same meaning).

7 Finding and Displaying Dictionary Words

To search the hashed dictionary I wrote the program `hebcheck.f` (see its `man` page) which can find a word by either its translation or its transliteration. Below is the beginning of the list it produces of translations best matching the English word `even`. Several translations match exactly while others resemble `even` in spelling but miss the meaning. The numbers at the left in the table are the line numbers of the words in `hebrew.hsh`; below I will explain how they can be used. The percentage score for each dictionary word indicates how closely it matches the query. Next comes the transliteration, and after the colon the English translation. Parenthesized strings in the translations are ignored in finding a match.

```
unix[1] hebcheck even
 4649  100%  yashar    : straight, even, right
 0065  100%  afeeloo   : even, even though, even if
 0007  100%  aaf       : also, though, even, surely; (a) nose; anger
 3950   51%  shivah    : seven (m)
 3907   51%  sheva     : seven (f)
 0067   50%  afpa'am   : even once
 0056   50%  af-kee    : indeed, even though
 1082   44%  esray     : ten (f pausal); teen
```

If you want `hebcheck` to look for a transliteration, put an equals sign `=` before and after it. Below is the beginning of the list the program produces of transliterations best matching `even`.

```
unix[2] hebcheck =even=
 1102  100%  even   : (a) stone
 3692   67%  seen   : name of Hebrew letter
 2693   67%  meen   : (a) kind, sort, variety; (a) sex, gender
 1100   67%  eved   : (a) servant
```

One transliteration matches `even` exactly while the others resemble it. Slight variations are often possible in how a word is transliterated, so showing imprecise matches helps to ensure that you will find the word you are looking for even if your guess at its transliteration is not exactly right.

To display the Hebrew of a word in the dictionary I wrote the shell script `hebshow` (see its `man` page). It constructs a LaTeX 2$_\varepsilon$ source file with commands appropriate to display the word or words that are requested, translates the LaTeX 2$_\varepsilon$ into Postscript, and invokes the `gv` program to display it in a window.

```
unix[3] hebshow =afeeloo= 1102
```

afeeloo	אֲפִילוּ	even, even though, even if
even	אֶבֶן	(a) stone

Here I specified two words to be displayed, the first by its transliteration and the second by its line number in `hebrew.hsh`. The `hebshow` script invokes a program named `hebxtr.f` (see its `man` page) to extract the hash code for a word from `hebrew.hsh` and translate the hash code into LaTeX 2$_\varepsilon$ commands; `hebxtr.f` in turn invokes the subroutine `HSH2HB` (see its `man` page).

8 Embedding Transliterations in Text

By using the LaTeX 2$_\varepsilon$ commands described in §2-§4 it is possible to typeset documents that include arbitrary Hebrew words. But if you want to include words that are in `hebrew.hsh` then it is possible to embed their transliterations in your document rather than spelling out the words one letter at a time. The example below shows how this can be done.

```
 1 % example4.heb
 2 \documentclass[12pt]{article}
 3 \input{/home/mike/bin/hebrew}
 4 \renewcommand{\hbold}[1]{#1}
 5 \pagestyle{empty}
 6 \begin{document}
 7
 8 \setivrit{12}
 9
10 \noindent The Tall Tale on page 68 of {\em The First Hebrew
11 Primer\/} is entitled\\
12
13 \hebline{\hebpnk{.}<khayah> <amar> <asher> <na'ar>\patach{hay}}
14
15 \end{document}
```

Now the `\hebline` command includes transliterations for the words נַעַר, אֲשֶׁר, אָמַר, and חַיָּה, rather than strings of LaTeX 2$_\varepsilon$ commands to spell them out. Here each transliteration is enclosed in <angle> brackets, rather than the equals signs we used in marking transliterations for `hebcheck` and `hebshow`. Angle brackets denote input and output redirection on the Unix command line so we couldn't use them to delimit transliterations there, but they cause no trouble here and using them instead makes the LaTeX 2$_\varepsilon$ code much easier to read. The first word of the Hebrew, הַנַּעַר, is not a dictionary word, but נַעַר is so I simply 13 prepended

the ך. Transliterations and LaTeX 2ε commands for setting Hebrew letters can be mixed freely. I used `\hebpnk` to set the period at the end of the Hebrew, but because the period is the same as its mirror image this does not change its appearance.

To translate input files like `example4.heb` into Postscript, I wrote the shell script `hebtex` (see its `man` page) which invokes the program `hebunt.f` (see its `man` page). The `hebunt.f` program reads a `.heb` input file containing transliterations, looks up each transliteration in `hebrew.hsh`, and expands the corresponding hash code into LaTeX 2ε commands. Then `hebtex` translates the resulting `.tex` file into Postscript. The terminal session below shows how to use `hebtex`.

```
unix[4] hebtex example4.heb
unix[5] gv example4.ps
```

The `gv` command displays this window.

> The Tall Tale on page 68 of *The First Hebrew Primer* is entitled
>
> הַנַּעַר אֲשֶׁר אָמַר חַיָּה.

9 Typing Paragraphs Left to Right

Typing transliterations is much simpler than typing words letter by letter, but putting the words in right-to-left order is a nuisance because editors such as `vi` type from left to right. Rather than typing Hebrew words in their lexical order of right to left on the page, it is faster and easier to type them in the temporal order that they are read, as in this example.

```
 1 % example5.ltr
 2 \documentclass[12pt]{article}
 3 \input{/home/mike/bin/hebrew}
 4 \pagestyle{empty}
 5
 6 \begin{document}
 7 \setivrit{12}
 8 \renewcommand{\hbold}[1]{#1} % turn off bold
 9
10 % LTR
11 \qamats{hay}<av> <shel> <khanah> <kholeh>.  <lak'khoo> <oto> <el>
12 <bayt-hakholim> \sheva{vav}<atsav> <gadol> <hayah>
13 \patach{bet}<bayit>.
14
15 \sheva{bet}<chol> <yom> <halchah> <khanah> <im> <imah> <el>
16 <bayt-hakholim> <l'vakayr> <et> <aba>.  <yom> <ekhad>\hebpnk{,}
17 <ca'asher> <halchoo> <el> <bayt-hakholim>\hebpnk{,} <sha'alah>
18 <khanah> ``<ima>\hebpnk{,} <madooa> <bara> <eloheem> <et>
19 \patach{hay}<ra> \qamats{bet}<olam>?''
20 % LTR
21
22 \end{document}
```

The `\setivrit{12}` command [7] in this example sets the Hebrew type size to 12 points. Then [11-19] we find two paragraphs of text, delimited by `%LTR` flags [10] [20] to indicate that the Hebrew between them (the first two sentences in the second chapter of *Hannah Senesh*) has been entered left-to-right. The blank line [14] produces a paragraph break. The `%LTR` flags must be entered exactly as shown, and any Hebrew appearing outside of them is assumed to be right-to-left.

I wrote the program `hebjst.f` (see its `man` page) to read a file that contains left-to-right text and reset it right-to-left within `\hebline` commands. In the terminal session below I use `hebjst` to read `example5.ltr` and write `example5.heb`. This invocation of `hebjst` tells that program to assume in right-justifying the left-to-right text that the typesize is `17.00` points rather than the 12 point size used for the Hebrew letters; I did this only so that the resulting lines would be short enough to conveniently display below. Then I used `hebtex` to produce `example5.ps` for display by `gv`.

```
unix[6] cat example5.ltr | hebjst 17.00 > example5.heb
unix[7] hebtex example5.heb
unix[8] gv example5.ps
```

The `gv` command displays a window like this.

הָאָב שֶׁל חַנָּה חוֹלֶה. לָקְחוּ אוֹתוֹ אֶל בֵּית־הַחוֹלִים וְעָצָב
גָּדוֹל הָיָה בַּבַּיִת.

בְּכֹל יוֹם הָלְכָה חַנָּה עִם אִמָּה אֶל בֵּית־הַחוֹלִים לְבַקֵּר אֶת
אַבָּא. יוֹם אֶחָד, בַּאֲשֶׁר הָלְכוּ אֶל בֵּית־הַחוֹלִים, שָׁאֲלָה
חַנָּה "אִמָּא, מַדּוּעַ בָּרָא אֱלֹהִים אֶת הָרַע בָּעוֹלָם?"

10 Constructing a Vocabulary List

You can list the unique transliterations contained in a document, with their English equivalents and optionally sorted, by using `heblist.f` (see its `man` page).

```
unix[9] cat example5.heb | heblist > list.heb
found  38 transliterations of which  30 are unique
```

Now `list.heb` contains LaTeX 2ε commands for setting a table whose first column contains the transliterations in the order they were encountered and whose second column contains the English translations of the corresponding Hebrew words. This source text can be included in a LaTeX 2ε document (such as the one whose vocabulary is listed).

11 Constructing a Lexicon

In Hebrew stories printed for beginners, one often finds that a few of the words have been footnoted on first appearance to give their English meanings. Unfortunately, when I read such a story I often find that the footnoted words are familiar while no translation is provided for others that are new. To make it easier for me to learn vocabulary by reading stories, I wrote a program called `heblex.f` (see its `man` page) to construct a list of all the distinct words in a page of text along with their meanings as given in `Hebrew/milon.dat`. The program formats the text of a story on right-hand pages with the lexicon for all of the words in that page on the facing left-hand page. That way, when I get stuck on a word I can easily find its meaning and then continue reading the Hebrew.

I wrote `heblex` to process the individual chapter files of a story book, so it expects that its input will *not* contain an `\end{document}` command. The Unix session below begins `[10]` by copying `example5.heb` to `example6.raw` but omitting its `\end{document}` command. Then `[11]` it uses `heblex` to produce the file `example6.heb` containing the Hebrew text and a lexicon of its words. Because this is output from `heblex` it does not end in an `\end{document}` command, so `[12]` one must be appended. Then `hebunt` can be used `[13]` to expand the transliterations and `[14] latex` to translate the result to a `dvi` file. Finally `dvips` can produce `[15] example6a.ps` containing the lexicon and `[16] example6b.ps` containing the Hebrew text. It is these files that are printed on the facing pages 14 and 15 of this document (page 13 is a right-hand page so it is blank).

```
unix[10] cat example5.heb | sed -e"/enddocument/d" > example6.raw
unix[11] heblex 1=example6.raw 3=/dev/null 4=temp 2>&1 > example6.heb
unix[12] echo "\end{document}" >> example6.heb
unix[13] cat example6.heb | hebunt 2>&1 > example6.tex
unix[14] latex example6.tex
unix[15] dvips -p=1 -l=1 -o example6a.ps example6.dvi
unix[16] dvips -p=2 -l=2 -o example6b.ps example6.dvi
```

Normally `heblex` is used in a `make` file to manage the assembly of a book from chapters, and then many of the complications required for this demonstration do not arise.

The first occurrence of each lexicon word is printed in boldface to show that it is new. Because this example has only one page, all of its words are new so all of them are printed in bold.

אָב	father; (month of) Av
שֶׁל	of
חַנָּה	Hannah
חוֹלֶה	patient, sick man; sick (adj)
לָקְחוּ	they took
אוֹתוֹ	him; same
אֶל	to
בֵּית־הַחוֹלִים	(the) hospital
עֲצָב	sadness
גָּדוֹל	big, great (ms)
הָיָה	he was
בַּיִת	(a) house
כֹּל	all, everything, whole (n)
יוֹם	(a) day
הָלְכָה	she walked, went
עִם	with; while; beside
אִמָּהּ	her mother
לְבַקֵּר	to visit
אֶת	direct object marker; with
אַבָּא	daddy
אֶחָד	one (m)
כַּאֲשֶׁר	when, just as
הָלְכוּ	they walked, went
שָׁאֲלָה	she asked, questioned
אִמָּא	mommy
מַדּוּעַ	why
בָּרָא	he created
אֱלֹהִים	God (of nature); judges
רַע	bad, evil (ms)
עוֹלָם	(a) world; forever

הָאָב שֶׁל חַנָּה חוֹלֶה. לָקְחוּ אֹתוֹ אֶל בֵּית–הַחוֹלִים וְעֶצֶב גָּדוֹל הָיָה בַּבַּיִת.

בְּכֹל יוֹם הָלְכָה חַנָּה עִם אִמָּהּ אֶל בֵּית–הַחוֹלִים לְבַקֵּר אֶת אַבָּא. יוֹם אֶחָד, כַּאֲשֶׁר הָלְכוּ אֶל בֵּית–הַחוֹלִים, שָׁאֲלָה חַנָּה "אִמָּא, מַדּוּעַ בָּרָא אֱלֹהִים אֶת הָרַע בָּעוֹלָם?"

12 Printing Flashcards

Another way to learn vocabulary words is by using flashcards. A flashcard has a Hebrew word (or words) printed on one side and the translation (or translations) on the other side. To use a flashcard you look at one side and try to recall the other, then turn the card over to check. If you do this many times eventually you will remember the English that goes along with the Hebrew and the Hebrew that goes along with the English.

Flashcards are not hard to make by hand, and some learning does occur in the process of doing that, but it is much easier and almost as good to generate them automatically by using the program `flashcards.f` (see its `man` page). It reads an input file of transliterations and generates a `.heb` file that can be processed by `hebtex`. When the resulting `.ps` file is printed 2-sided each page contains three 3×5 flashcards each with a Hebrew word or words on one side and the corresponding English translations on the other. The cards can be cut out for convenient handling for review as described above. Many printers will accommodate card stock or paper heavy enough to serve that purpose, but you might find that flashcards printed on ordinary 20-pound paper work well enough.

The terminal session below shows how to use the program. The `lpr` option you need for two-sided printing depends on what kind of printer you have.

```
unix[17] more in.flash
<shalom>
<aba> <ima>
unix[18] cat in.flash | flashcards > out.heb
unix[19] hebtex out.heb
unix[20] lpr -o Duplex=DuplexTumble out.ps
```

I have printed the result `out.ps` on the following two pages back-to-back, so that you can cut out the three flashcards (the bottom one is blank). If you imagine that the guide number in the upper left corner of each card has an unshown leading decimal point, then filing the cards in the order of those decimal fractions will put them into alephbetical order. In the example the top card has guide number 0.33180819 while the middle one has guide number 0.010201012001, and filing them in the order of those numbers would put them in alephbetical order.

If you want to make flashcards for all of the transliterations in a document you can use `heblist` to produce a vocabulary list and use that as the input to `flashcards`.

33180819

שָׁלוֹם

010201012001

אַבָּא
אִמָּא

peace; hello; good bye

daddy
mommy