

Another application is related to the fact that the files `tex.web` and `mf.web` are extremely large; they are in fact too large to be even inspected by our text editor. In order to circumvent this problem one can split the file `tex.web`—D.E. Knuth might forgive this—into smaller files containing the limbo material (`tex00.web`) or the code for one chapter of ‘`TeX: The Program`’ (`tex01.web` through `tex55.web`) and split the change files for `TeX` and `INITEX` accordingly. The primary patch for `TeX` will then consist of a short `WEB` file (say `texskel.web`, the skeleton) containing 56 ‘`Qi`’ commands to invoke these patches. This primary patch requires no change file as all changes are applied to the secondary patches. The skeleton file for `INITEX` will be almost identical. Only one or two secondary patches will be different, their patch files will have to specify an additional change file in order to create `INITEX` instead of `TeX`, but there is no necessity to maintain, for each kind of installation, two almost identical change files, one for `TeX` and one for `INITEX`. Furthermore one could create a L-R `TeX` (`TEX-XET`) and/or a multilingual `TeX` by just adding one additional change file to a few of the secondary patches. These additional change files could probably be installation-independent.

### Turkish Hyphenations for `TeX`

Pierre A. MacKay

Turkish belongs to the class of agglutinative languages, which means that it expresses syntactic relations between words through discrete suffixes, each of which conveys a single idea such as plurality or case in nouns, and plurality, person, tense, voice or any of the other possibilities in verbs. Since each suffix is a distinct syllable (occasionally more than one syllable), Turkish sentences are likely to contain a high proportion of long multi-syllable words, and to need an efficient system of hyphenation for typesetting. Owing to the long association of almost every Turkic-language region with Islam, certain conventions of the language have been deeply influenced by Arabic orthographic habits, and among these is the syllabification scheme on which a system of hyphenation is built.

According to the syllabification pattern of Arabic, a syllable is assumed always to consist of an initial consonant (even when that consonant is no longer written) and to terminate in a vowel `-cv-` or

in the next unvowelled consonant `-cvc-`. This pattern is followed so absolutely that it is permitted to break up native Turkish suffixes. The plural suffix `-ler-` will be hyphenated as `-le-rine` in an environment where the `-cv-cv-cv` pattern predominates. A syllabic division of *çektirilebilecek* provides six places for hyphenation *çek-ti-ri-le-bi-le-cek*, while a morphological division of the word would produce only five *çek-tir-il-e-bil-ecek*.\*

There are almost no exceptions to this pattern. Words which appear to begin with a vowel, like *et-mek*, can also be described as beginning with the now suppressed half-consonant *hamza*. Widely sanctioned orthographic irregularities like *brak-mak* can be found in stricter orthography as *bı-rak-mak*. The only universally practiced violation of the rule is associated with the word *Türk*, in which the `-rk-` combination is inseparable, and contributes to several of the very few three-consonant clusters regularly used in the language—*Türkçe*, *Türkler*. One other significant consonant cluster occurs in the suffix *[i]m-trak*.

The Ottoman Texts Project at the University of Washington has undertaken the development of a set of editing and typesetting tools for the production of texts in modern Latin-letter Turkish, using the full range of diacriticals needed for scholarly editions of historic Arabic-script manuscripts. Because we wish to work in cooperation with scholars in Turkey, who are most likely to have access to unmodified versions of `TeX`, we have chosen a font-based adaptation of the `TeX` environment, which will require no alterations in the program. The work on fonts is largely complete, and one of the last major efforts necessary is the creation of a Turkish hyphenation table.

The obvious way to create such a table in the `TeX` environment, is to run a list of correctly hyphenated words through `Patgen`, but it is not always easy to find such a list. English and German dictionaries quite commonly provide hyphenation patterns, but the dictionaries of the Romance languages rarely do, and in Turkish, the hyphenation pattern is so obvious that the production of such a list is viewed as an unimaginable waste of time. Rather than try to scan a Turkish word-list and supply hyphens, we have taken advantage of the strict formalism of the patterns and generated the Turkish hyphenation file by program.

---

\* The word is a future participle, and describes something as being capable of being extracted at some time in the future—like a tooth.

Turkish orthography uses a very large number of accented characters. The Latin-letter character set which has been in use since the orthographic reform of 1928 is extended, even in Modern Turkish, by means of a considerable number of diacriticals and accents. A diligent search through the modern dictionary will produce several five- and six-letter words in which every character is accented, and an intensive search might come up with words as much as nine letters long with every character accented. In critical editions of Ottoman texts, the number of accents more than doubles. Modern Turkish knows only the accented and unaccented pair of letters 's' and 'ş', but Ottoman Turkish has 's', 'ş', 'ş' and 'ş', which represent four completely distinct characters in the Arabic alphabet. The letter 'h' shows almost as much variety, and so do several others. Our Ottoman Turkish font has twenty-seven accent and letter composites, in addition to the basic twenty-six simple Latin letters. Moreover, all composites can exist in upper case forms as well as in lower case. To accommodate these composite characters in the normal ASCII character set, we use an input coding convention in which accented letters are treated as a class of ligatures, and three characters from the ASCII symbol set are borrowed for use as postpositive pseudo-letters, to trigger the selection of accented letters in the Turkish fonts. The three symbols are the exclamation point '!', the equals sign '=', and the colon ':'. The

The choice of these symbols is based on a proposal made more than ten years ago at the Orientalist Congress held in Paris, in 1974. Owing to the extraordinary richness of the Ottoman Turkish character set, it has been necessary to extend the old proposal, but it still retains the original principles, which are closely associated with the coding scheme used by the Onomasticon Arabicum project, which is coordinated at the Centre National de la Recherche Scientifique in Paris. (The Onomasticon Arabicum uses a post-positive dot and a post-positive hyphen to indicate diacriticals, which is acceptable in a data-base of names, but not in continuous prose text.) The current set of conventions, using (! = :), produces an input file which can, if necessary, be edited on a ordinary terminal lacking any special Turkish character features, and which a Turkish speaker can become accustomed to without too much difficulty. When coupled with a well-designed macro file and a rewritten hyphenation table, it provides the possibility of naturalizing a T<sub>E</sub>X environment into Turkish without any large investment in special purpose hardware and rewritten versions of non-standard (non-)T<sub>E</sub>X.

The exclamation point is used for all the "emphatic" letters of the Arabic alphabet (the alphabet in which Turkish was written until 1928). These are the letters *Ḍad* (usually pronounced as 'z' in Turkish, and hence paired with a non-Arabic letter known as *Ẓad*), *Ṣad*, *Ḥa*, *Ṭa* and *Ẓa*. The equals sign is used for all the consonants which are represented in Latin-letter transcriptions by a letter with a bar under, such as 'ḏ' (*dhal*), more commonly written in Turkish as 'z', and also for vowels with a macron or, following the Turkish convention, a 'hat' accent, and similar forms, chosen like the cupped 'ğ', because the equals sign is visually closer than the colon is. (Moreover, the colon is needed for a different variety of the letter 'g'.) The colon is a catch-all for everything else, but works out rather well visually, as it happens. The three post-positives are not accents, but regular characters, which use the T<sub>E</sub>X convention of ligatures to invoke accented characters from the font, just as the second 'f' in the normal T<sub>E</sub>X 'ff' ligature pair does. If a standard Latin-letter character does not have an associated ligature table in the font, a following diacritical postpositive will be unaffected. Thus, the letter 'o', when followed by a colon will produce 'ö', but the letter 'e' when followed by a colon will produce 'e:'. The equals sign retains its normal function in math mode because the math font TFM files do not call it into ligature pairings, and the colon and exclamation point can be invoked by the command sequences \: and \bang when the simple character will not work.

Since the hyphenation evaluation loop in T<sub>E</sub>X dismantles all ligatures before it looks for acceptable hyphenation positions, it will have to accept the post-positive symbols (! = :) as part of the alphabet, so each of these symbols receives its own value as an \lccode. The full Turkish-T<sub>E</sub>X alphabet is:

```
a â e i î o ö ô u ü
' ' b c d f g h j k l m n p r s t v y z
ḏ ḥ ḵ ṣ ṭ ṭ
ḏ ḡ ḥ ṇ ṣ ṭ ṭ
ç ğ ş ź
```

In the hyphenation loop of T<sub>E</sub>X, these characters resolve into the set:

```
! = : @ # a b c d e f g h i
j k l m n o p r s t u v y z
```

and it is this latter set only which will appear in the hyphenation patterns. The dotted i in the above list really stands for the Turkish undotted

'i'. The input code convention for Turkish uses i: for the Turkish 'i'. The @ sign stands for the Arabic letter *hamza* and the # stands for *ayn*. To avoid conflict with `plain.tex` uses of these two characters, they appear explicitly only in the hyphenation pattern file. Turkish text input uses \ ' to generate \char'43 (*ayn*) and \ ' to generate \char'100 (*hamza*).

We begin constructing the table by considering the pseudo-letters (! = :). Since these are used exclusively in ligature pairs, no hyphenation is ever permissible between them and the preceding letter. Odd values permit, and even values in the hyphenation code prohibit hyphenation, so we give the highest possible even value (8) to the region preceding each pseudo-letter. The pseudo-letters can follow both vowels and consonants, so hyphenation will often, but not always, be possible after them. We give that region the lowest possible odd value (1) to show that hyphens are permitted here.

8!1 8=1 8:1

In strict orthography, a vowel cannot be separated from the preceding consonant, and the few apparent instances of hyphenation between two adjacent vowels (suppressed consonant) can be treated later. In all normal instances a vowel cannot accept a hyphen in the preceding region and will probably accept one in the following region, so the vowels are set thus.

2a1 2e1 2i1 2o1 2u1

A consonant may begin a -cv- sequence or end a -cvc- sequence, so we give it a 1 on either side:

1b1 ... 1z1

This simple lot of patterns will provide for all normal -cv- instances such as

1h1  
8=1  
2a1  
1h8=2a1

which will result in the sequence -h=a-, with hyphens fore and aft.

The next group of patterns controls hyphenation at the end of words. `TeX` will usually not break off two-letter fragments in its hyphenation loop, but owing to the nature of the input coding we have chosen, it may see a three- or four-letter sequence where a two-letter result is intended. We do not want to find *lû*, *çû* and *si* isolated at the beginning of a line, nor do we really want the *cek* of -*ecek* broken off if it is at the end of a word. To prevent hyphenations of this sort, the program generates all possible patterns of the type:

2ba= . ... 2z:u:.

using the conventional . for end-of-word. The resultant list includes sequences that are phonetically impossible in Turkish but these take up so little additional space in the file that they can be left there. The pattern 2e2cek is added as a special case.

The break after -cvc- syllables is almost taken care of:

1h1  
8=1  
1h1  
1h8=1h1

but it makes the thoroughly undesirable -cv-ccv- sequence as acceptable as the correct -cvc-cv- sequence. To prevent this error, all possible Turkish two-consonant sequences (e.g. h=h= → 'hh') are covered by patterns such as 2h=h=, in which the value 2 will override the 1 after the preceding vowel.

The few undesirable hyphenations at the beginning of words which appear to start with a vowel are prevented by generating the patterns .a=2 through .u:2 and similarly, the few instances where an apparent -cv-v- hyphenation stands for -cv-[c]v- can be allowed by adding the full range of patterns a3a2 through u:3u:2 which includes a large number of impossible pairings.

The last patterns to be added are m1t4ra4k and t2u8:2r4k1. At the price of slightly excessive strictness (the prohibition against the r-k division is only valid when the word begins with an upper-case T) we can ensure that *Türk* always stays in one piece.

Files of this sort, when generated by program, tend to be larger than hand-worked files, but if it seems that all the redundancies mentioned above might be seriously wasteful of space, consider the following statistics:

	Entries	Trie size	Ops
English	4460	5492	181
Turkish	1840	616	16

The format file that makes use of this set of patterns will no longer serve very well for English language `TeX`. The font-based solution to foreign-language typesetting is definitely monolingual, since only one `hyphen.tex` file can be read in at a time. A multilingual system, good for both English and Turkish, would require modifications of the program code. This simple solution, however, will be quite satisfactory in a purely Turkish environment, and can be made even more successful by taking the `tex.pool` file and translating it all into Turkish.