Software

Still another aspect of multiple change files: The PATCH processor

Peter Breitenlohner

Recently there have been quite a few TUGboat articles about extensions of the WEB system, either extensions to other languages like C or Modula-2, or extensions which allow multiple change files (W. Appelt and K. Horn in TUGboat 7(1986)20, K. Guntermann and W. Rülling in 7(1986)134, E.W. Sewell in 8(1987)117). Surprisingly enough (at least for me) another extension which allows one to *include* or *insert* a WEB file together with its change file(s) into another one has never been discussed. I would therefore like to present a program which does exactly this.

Before describing this program let me recall my motivation to write it. My experience with the WEB system dates back to the time when I had just installed METAFONT, had proudly produced the first GF file and was then told by GFtype that all kinds of backpointers were wrong. This was repaired easily enough — the two programs had different opinions about the record length of GF files — but it has brought the following fact to my attention. The WEB source files for TEX, METAFONT and their friends contain large sections of code which occur in many files in more or less identical form and the changes applied to them had better be consistent. The chapters 'The character set' or 'Packed file format' are typical examples and the updates to GFtoPK, PKtoPX, PKtype and PXtoPK published in TUGboat 7(1986)140 are a characteristic symptom.

If one could make such sections of code completely (not just almost) identical one could keep *one* copy of this code in a separate file and *include* it whenever needed. This would clearly save a rather large amount of disk space. Much more important this would guarantee that the same changes are applied whenever this section of code is used and would greatly facilitate the task of creating a new set of change files for a new computer, compiler or operating system.

These considerations motivated me to write a program which I have named PATCH because it takes various patches and combines them into one program file. Each patch consists of one WEB file and $n \geq 0$ change files *change_j* $(0 < j \leq n)$ which are applied one after the other. Here PATCH operates exactly like TIE as described by Guntermann and Rülling: first *change_1* is applied to the WEB file as usual, then *change_2* is applied to the result, and so on. In addition there is one short 'patch file' which specifies the names of the WEB and change files. An important milestone is reached when the result of this merging of files yields a record starting with '@i', a control code which is normally undefined. Such a record must contain the file name of a new patch file and the resulting new patch is inserted instead of this record. This new (secondary) patch may, of course, again yield records starting with '@i' and thus invoke further secondary patches and so on.

The PATCH processor has actually three modes of operation: In *merge_mode* PATCH produces a WEB file and *one* change file, and can thus serve as preprocessor for TANGLE and WEAVE. In this mode all change files *change_j* are combined into one change file and all secondary patches are inserted in a suitable way. In *insert_mode* PATCH produces just a WEB file containing all the changes and insertions. This WEB file can also serve as input to TANGLE or WEAVE but in this case the information about changed modules would not be available to WEAVE. Finally, in *update_mode*, all changes are applied to the primary patch. Requests for secondary patches are, however, not honored but copied to the resulting WEB file. This mode of operation is intended to incorporate modifications into a WEB file once they have been fully tested and are frozen.

A next step was to combine PATCH with TANGLE and WEAVE to new programs TPATCH and WPATCH in order to avoid the necessity of a separate preprocessing step. Using PATCH, this turned out to be surprisingly easy due to the clear structure of TANGLE and WEAVE. All the code for the actual processing (modules 37–178 of TANGLE and modules 36–257 of WEAVE) required practically no changes except that the parts which merge the WEB and change file (modules 124–138 resp. the almost identical modules 71–85) had to be replaced by the corresponding code from PATCH. All three processors PATCH, TPATCH and WPATCH have now been used for several months and are thoroughly tested.

Coming back to the original motivation for PATCH one can now try to create patches for things like 'Reading GF files', 'Reading PK files' and 'Reading PXL files', or maybe two versions of them for sequential and random access to the files. At the moment I am in fact doing just this. Such patches should be tools which can be inserted into a program whenever needed. They can be extremely useful for all kinds of DVI drivers which could use any one of them or even two of them alternatively.

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 '`@i`' 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.