# Multiple Changefiles in WEB

**Wolfgang Appelt**
**Karin Horn**
Gesellschaft für Mathematik und Datenverarbeitung mbH

TANGLE and WEAVE usually read one *webfile* and one *changefile* to produce the desired Pascal source file or the corresponding TEX input file. There are, however, situations when it would be useful if TANGLE and WEAVE could read several *changefiles* simultaneously to create their output files.

Imagine, for example, a TEX site where TEX is running on several computers with different operating systems (say $s_1, \ldots, s_n$) and where different output devices ($o_1, \ldots, o_m$) are supported (as is the case at GMD). If a *changefile* for the DVItype program is written to create a driver for the output device $o_j$ running on the system $s_i$, the *changefile* will usually contain a set of changes, say $\mathcal{A}_i$, which only concerns the operating system but not the output device. A second set of changes, $\mathcal{B}_j$, may only concern the output device and a third one, $\mathcal{C}_{ij}$, may concern both the operating system and the output device. (This set may be empty on many systems.) There might even be a set of further changes, $\mathcal{D}$, to support some \special features, e.g. for graphics. In other words, a *changefile* for a specific output device on a specific system can be regarded as the union $\mathcal{A}_i \cup \mathcal{B}_j \cup \mathcal{C}_{ij} \cup \mathcal{D}$ where each of these subsets is logically independent from the others.

Basically, there are two possible ways to store the *changefiles*:

(1) For each combination of an operating system and an output device there exists one complete *changefile*. Not only is space wasted in this way; an even greater disadvantage of this method is that whenever a modification is necessary (maybe because a bug was found or because a new system release was installed), the same modification would have to be applied to several *changefiles*.

(2) All the different sets of changes $\mathcal{A}_i$, $\mathcal{B}_j$, $\mathcal{C}_{ij}$ and $\mathcal{D}$ are kept in separate files. Only if a specific driver program has to be created are the required files merged to create a valid *changefile*. Merging these files, however, might not be a trivial task, since a simple concatenation of $\mathcal{A}_i$, $\mathcal{B}_j$, $\mathcal{C}_{ij}$ and $\mathcal{D}$ is usually *not* sufficient.

A better solution which avoids these problems would be a version of TANGLE and WEAVE that can process more than one *changefile*. We have therefore written two programs which we call KNIT and TWIST which implement that feature. (The sum of the two programs might be called the PATCHWORK system).

The philosophy for handling several *changefiles* is as follows: Assume we have $n$ *changefiles*, called *change_1* ... *change_N* and, furthermore, assume that a line of text which appears between @x and @y in any *change_i* does not appear in any other *change_j*, i.e. all changes appearing in the *changefiles* concern *distinct* parts of the *webfile*. In this case the output files of KNIT and TWIST are identical* to those obtained by TANGLE and WEAVE with a *changefile* which is created by merging *change_1* ... *change_N* properly together.

If, however, two (or more) *changefiles* want to change the same piece of text within the *webfile* only the modifications by the *changefile* which claimed its right first will take effect; the others are ignored. In other words, in such a case the numbering of the *changefiles* (*change_1* is read before *change_2*, etc.) is important. KNIT and TWIST will give a warning if two *changefiles* want to change the same text of a *webfile* since this is probably an error. (Nevertheless, there can be situations where one may deliberately construct "conflicting" *changefiles*.)

* To be precise, there is a *slight* difference: Changed modules are marked with the number of the *changefile* which caused the modification and not just with an asterisk as WEAVE does.

The KNIT and TWIST programs were created by writing two *changefiles*, namely `knit.chg` and `twist.chg`. TANGLE'ing `tangle.web` with `knit.chg` will result in `knit.pas`, the Pascal source program for KNIT, and correspondingly TANGLE'ing `weave.web` with `twist.chg` will give you `twist.pas`. Setting up the KNIT and TWIST processors is therefore similar to bootstrapping the WEB system.