The Components of TEX

Joachim Schrod
Detig · Schrod TEXsys

March 1991

Abstract

TEX needs a great amount of supplementary components (files and programs) of which the meaning and interaction often is unknown. This paper explains the components of the kernel system TEX that are visible for the TEX user and their relations.

1 About this Report

TEX is a typesetting system which offers authors easy usage of powerful typesetting features to produce printed matter which is the state of the art of computer typesetting. This is, however, not done by the TEX program alone: A significant number of supplementary programs and files together form the complete typesetting and authoring system. Along with the programs that belong to TEX directly, there exist two other major programs which were built by DONALD KNUTH in connection with TEX and must be included in an explanation of the full system: METAFONT, for the generation of fonts, and `WEB`, a documentation and developing 'language' for programming. TEX and METAFONT are written in `WEB`.

This text describes this 'kernel' TEX from a user's viewpoint: at the end you should have an overview of the ingredients of the TEX system, and about the files and support programs that are essential for you as a user. This will not, however, be an introduction to the capabilities of TEX or how you may run TEX on your computer.

I will use marginal notes to identify the places where terms are explained for the first time. Abbrevations for file types – usually identified by common suffixes or extensions – are set in a monospace type ("`typewriter`") and those abbrevations are put into the margin, too. Please note that these abbreviations are sometimes not identical with the file extensions (see also Table 1).

This report is the start of a series that describes the subsystems mentioned above and their respective components. In that series each report will focus on one subsystem in a special point of view; it should not result in gigantic descriptions which tell everything (and then nothing). In my opinion the following reports will be of interest:

- the structure of a standard installation of TEX
- `DVI` drivers and fonts
- possibilities of graphics inclusion in TEX documents
- the components of METAFONT
- the structure of a standard installation of METAFONT
- `WEB` systems — the concept of Literate Programming
- other (though not yet planned) themes of interest are perhaps
  - differences between TEX and DTP systems
  - the way how TEX works (there exists some good books on this topic!)
  - the limits of TEX
  - TEX as a programming language

The reports will be published in this sequence.

2 What is TEX?

TEX is a typesetting system with great power for the typesetting of formulae. Its basic principle is that structures in the document are marked and transformed into typeset output. Providing such information about the structure of a document is known as *markup*. If the marks describe the look of the document, it is called *optical markup*, while, if document structures are marked, it is called *logical markup*. TEX provides both forms of

Actual address: Joachim Schrod, Detig · Schrod TEXsys, Kranichweg 1, D-6074 Rödermark, FR Germany, Email: `xitijsch@ddathd21.bitnet`

markup, i. e., exact control of the layout of parts of the document and their positioning as well as the markup of the structure of formulae or document components. The logical markup is mapped to the optical one by TEX so that layout may serve for the identification of structures by the reader.[1]

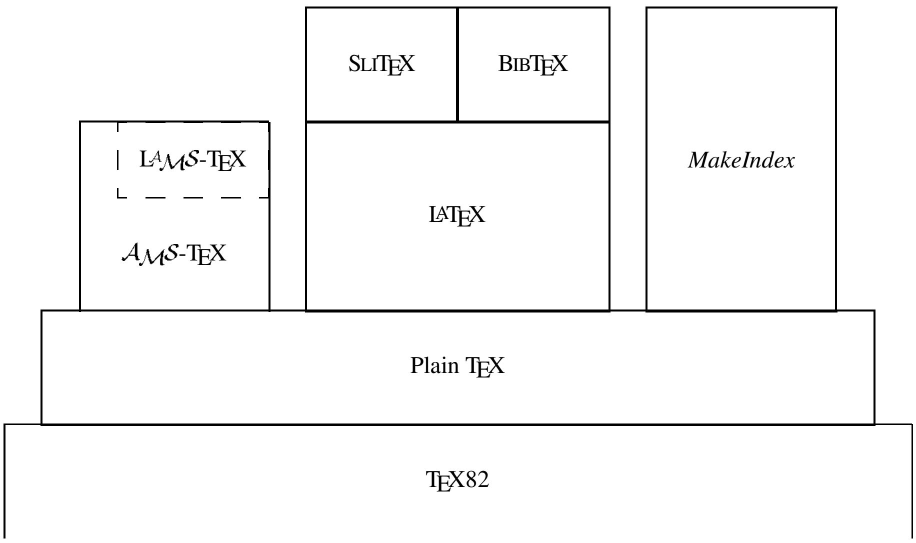
The kernel of the TEX typesetting system is the formatting program TEX82, which is often simply called TEX. This usage shall be adopted here whenever the difference between the complete system and the formatter is unimportant or obvious. TEX82 is a big monolithic program which is published in the book *TEX: The Program* by DONALD KNUTH. Its features may be separated into two levels:

1. TEX82 formats text, i. e., it breaks it into paragraphs (including automatic hyphenation) and produces page breaks.
2. It provides the programming language TEX which incorporates a macro mechanism. This allows new commands to be built to support markup at a higher level. DONALD KNUTH presents an example in the TEXbook: Plain TEX. A collection of macros which supports a special task and has (hopefully) a common philosophy of usage is called a *macro package*.

High level features for optical markup, as represented by Plain TEX, allow one to build additional levels leading to full logical markup. At the moment, two macro packages for logical markup are widespread: AMS-TEX and LATEX. Both systems are built on top of Plain TEX to greater or lesser extents and the user can use the optical markup of Plain TEX in addition to logical markup if desired. This results in the effect that the author can use a mixture of structural information and explicit layout information – a situation with a high potency of features that nevertheless can (and does) lead to a lot of typographic nonsense!

As TEX82 was built only for typesetting texts and to allow the realization of new markup structures, many features are lacking which are required by authors. To provide features like the production of an index or a bibliography or the inclusion of graphics, additional programs have been written, which use information from a TEX82 formatting run, process them, and provide them for the next TEX82 run. Two supplementary programs are in widespread use and available for many computer/operating system combinations: BIBTEX, for the production of a bibliography from a reference collection, and *MakeIndex*, for the production of an index.

A special case of the processing of information provided by a TEX run is the production of a table of contents or the usage of cross references in a text. For this only informations about page numbers, section numbers, etc., are needed. These are provided by TEX82 and can be processed by TEX82 itself, so TEX82 is used as its own post processor in this situation.



Figuur 1: *The Components of TEX*

[1] Layout – and book design in general – do not represent useless beauty. A good book design must first support the understanding of the content to produce readable text. So it is *æsthetic* in its best sense, since it connects form and contents and builts a new quality.

We have now seen that the TEX typesetting system is a collection of tools that consists of the typesetting 'engine' TEX82, macro packages (maybe several that are based on others) and supplementary programs, used together with these macro packages. This relation is illustrated by Figure 1.

3 Formatting

The formatting process of TEX needs information about the dimensions of characters used for the paragraph breaking. A set of characters is grouped in *fonts*. (But this is a simplification as the notion "font" should be used for the realization of a type in a fixed size for a specific output device.) The dimensions of the characters of a font are called *font metrics*.

The format in which the font metrics are used by TEX was defined by DONALD KNUTH and is called `TFM` format ("*TEX font metrics*"). In this format, every character is descibed as a *box* with a height, a depth, and a width. TEX only needs these measurements, it is not interested in the shape of the character. It is even possible that the character may extend outside the box, which may result in an overlap with other characters. The character measures are specified in a device independent dimension because TEX processes its breaking algorithm independent of any output device.

During paragraph breaking, TEX hyphenates automatically, which can be done in an almost language-independent way. For the adaption to different languages, *hyphenation patterns* are needed to parametrize the hyphenation algorithm.

The result of a TEX formatting run is a `DVI` document, in which the type and position on the page are specified for each character to be output. The resolution that is used is so small that every possible output device will have a coarser raster, so that the positioning is effectively device independent. The `DVI` document specifies only types, not the fonts themselves, so that the name `DVI`[2] ("*device independent*") is accurate. To make the result of the formatting run available, the `DVI` file must be output by a so-called `DVI` driver on the desired output device.

If problems occur during the formatting, error messages or warnings are output on the terminal. *Every* message that appears on the terminal will also be written into a protocol file named `LOG` file. In this `LOG` file additional information may be placed that would have been too verbose for the output to the terminal. If this is the case, TEX will tell the user so at the end of the formatting run. The messages of TEX are not built in the program, they are stored in a (string) `POOL` file. These messages must be read in at the beginning of a run.

4 Macro Packages

The basic macro package is Plain TEX, developed by DONALD KNUTH together with TEX82. It parametrizes the TEX82 typesetting machine so that it can typeset English texts with the Computer Modern type family. Additionally, Plain TEX provides optical markup features. Plain TEX is available as one source file, `plain.tex`.

All other macro packages known to the author are based on Plain TEX, i. e., they contain the source file `plain.tex` either originally or with modifications of less important parts. Next to Plain TEX, the most important (free) available macro packages are AMS-TEX by MICHAEL SPIVAK and LATEX by LESLIE LAMPORT. Other free macro packages are often of only local importance (e. g. BlueTEX, TEXT1, or TEXsis) or are used in very special environments only (e. g. `texinfo` in the GNU project or `webmac` for `WEB`). Important commercial macro packages are MacroTEX by AMY HENDRICKSON and LAMS-TEX, also written by MICHAEL SPIVAK.

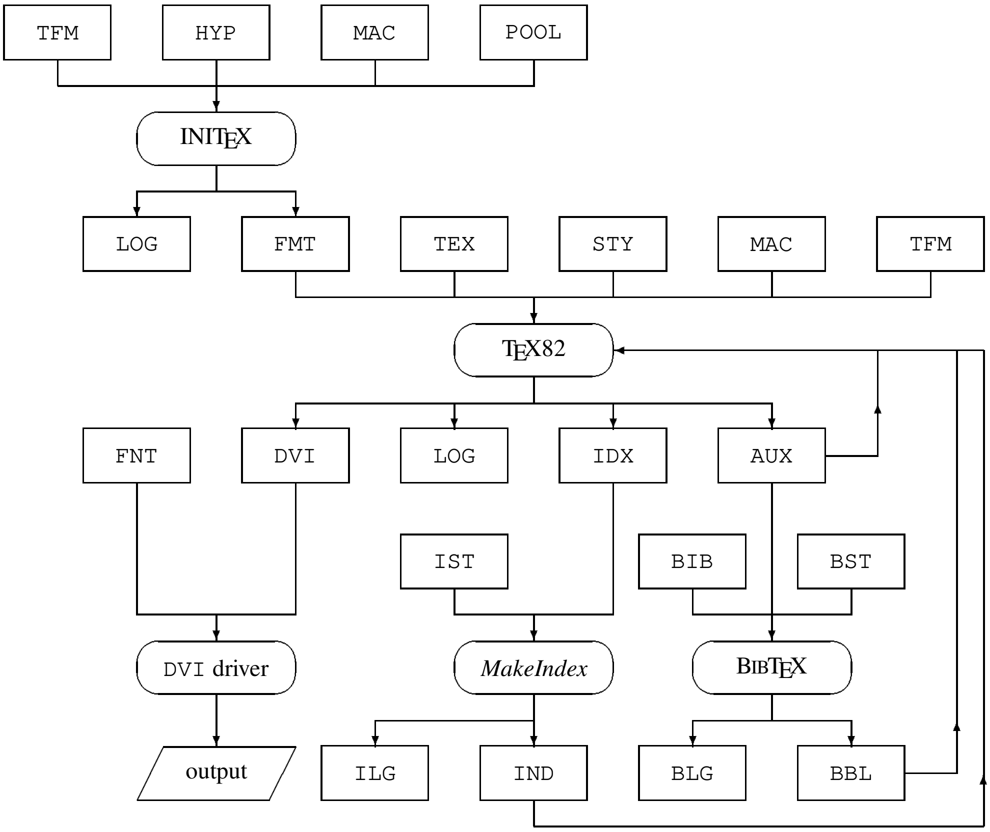
These macro packages usually consist of a kernel that provides additional markup primitives. With such primitives, *document styles* can be built which realize logical markups by a corresponding layout. This layout can often be varied by *sub-styles* or *style options* which may also provide additional markups.

The macro packages produce *supplementary files* which contain information about the page breaks or the document markup. This information may be used by support programs – e. g., the specification of a reference from a bibliography database or the specification of an index entry with corresponding page number for the construction of an index. A special case is the information about cross references and headings for the building of a table of contents, as this information can be gathered and reused by TEX directly.

SLITEX is a special component of LATEX for the preparation of slides with overlays. In TUGBoat volume 10, no. 3 (1989) LAMS-TEX was announced, which will provide the functionality of LATEX within AMS-TEX. MacroTEX is a toolbox of macro "modules" which may be used to realize new markups but, as it became available only short time ago, it is not yet widespread.

For the usage of these (and other) macro packages, one must check whether they need additional fonts which do not belong to the Computer Modern type family. For LATEX, e. g., fonts with additional symbols and with invisible characters (for the slide overlays) are needed, while AMS-TEX needs several additional font sets with mathematical and Cyrillic characters.

[2] This name is a problem because "DVI" is a trademark of Intel Corp. now, but the name `DVI` for TEX output files pre-dates this.



Figuur 2: *The Connection of Components and File Types*

5 Support Programs

Only two support programs will be discussed here: BIBTeX by OREN PATASHNIK for the preparation of bibliographies and *MakeIndex* by PEHONG CHEN and MICHAEL HARRISON for the preparation of a sorted index. For both tasks exist other, funcionally equivalent, support programs. But the abovementioned are available on many operating systems, and have an "official" state as they are LESLIE LAMPORT encourages their usage with LaTeX in his documentation, and the TUG supports them for general use.

There is no totally portable mechanism for the inclusion of general graphics in TeX documents, so that there are no machine independent support programs available.

BIBTeX is used to handle references collected in `BIB` files. TeX produces supplementary files which contain information about the required references, and BIBTeX generates from them a sorted bibliography in a `BBL` file which may be subsequently used by TeX. The kind of sorting and the type of cite keys are defined by *bibliography styles*, specified in `BST` files. The messages of a BIBTeX run are written to a `BLG` logfile.

MakeIndex reads an `IDX` support file that contains the index entries and the according page numbers, sorts these items, unifies them and writes them as TeX input in an `IND` file. The formatting style may be specified by an *index style*. The messages of a *MakeIndex* run are written to a `ILG` file.

6 Performance Improvements

Much of the work that TeX82 has to do is the same for every document:

1. All text has to be broken into lines. Text pieces in the same language are hyphenated with the same hyphenation patterns.
2. The basic markups of the corresponding macro packages must be available.
3. The required font metrics are much alike for many documents, as the font set used usually doesn't differ that much.

To improve TeX's performance, hyphenation, markup, and font metrics descriptions are converted from an external, for (1) and (2) textual, representation into an internal representation which can easily be used by TeX82. It is sensible to do this transformation only once, not for every document. The internal representation is stored in a `FMT` file. The storing is done with the TeX command `\dump`, so that `FMT` files often are

FILE TYPE	EXPLANATION	FILE IDENTIFICATION (SUFFIX, EXTENSION, ETC.)
TEX	Text input	tex, ltx
DVI	TEX82 output, formatted text	dvi
LOG	TEX82 log file	log, lis, list
HYP	Hyphenation patterns	tex
TFM	Font metrics	tfm
POOL	String pool	pool, poo, pol
FMT	Format file	fmt
MAC	TEX macro file	tex, doc
STY	TEX style file	sty, tex, st, doc
AUX	Support files	aux, toc, lot, lof, glo, tmp, tex
BIB	Reference collections	bib
BBL	References or bibliographies	bbl
BLG	BIBTEX log file	blg
BST	BIBTEX style file	bst
IDX	Unsorted index	idx
IND	Sorted index	ind
IST	Index markup specification	
ILG	*MakeIndex* log file	ilg

Tabel 1: *File Types*

called "*dumped formats*." A FMT file can be read at the beginning of a TEX82 run and is thus available for the processing of the actual text.

As the creation of a FMT file is done infrequently – usually for the update of a macro package – the formatting of texts can be done with a reduced version of the TEX82 program that doesn't contain the storage and the program parts for the transformation of the hyphenation patterns and for the dumping. The complete version of TEX82 is needed in an initialization phase only and therefore called INITTEX. Additional improvements of the performance can be reached by the usage of production versions of TEX82 from which parts for statistical analysis and for debugging are stripped.

TEX versions that have no dumped formats preloaded, have the ability to load a dumped format (i. e. a FMT file), and have no ability to dump a FMT file (i. e., they are not INITTEX) are often called VirTEX, which stands for *virgin TEX*.

7 Connections Between File Types and Components

In the above sections, the components of the TEX authoring system were described, and the files that are read or written by these components mentioned. The connections between them all is demonstrated graphically in Figure 2. In this graphic, file types are represented by rectangles, and programs by ovals. The arrows mean "is read by" or "is produced by." The abbreviations of the file types are explained in Table 1, which also lists the file identifications (suffixes or extensions) that these files usually have (but note that other file identifications are also in use).

Acknowledgements

I would like to thank CHRISTINE DETIG who was so kind to provide the English translation. NELSON BEEBE suggested performing the translation. KLAUS GUNTERMANN made valuable comments on the first (German) version. NICO POPPELIER contributed a new version of figure 2, better than my original one.