

Web2c

for version 2010
July 2010

Karl Berry
Olaf Weber
<http://tug.org/web2c>

This file documents the installation and use of the programs in Web2c, an implementation of Donald Knuth's TeX system.

Copyright © 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2007, 2008, 2009 Karl Berry & Olaf Weber.

Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of this manual into another language, under the above conditions for modified versions, except that this permission notice may be stated in a translation

Table of Contents

1	Introduction	1
2	Installation	2
2.1	configure options	2
2.2	Compile-time options	3
2.3	Additional targets	4
2.4	Trip, trap, and mptrap: Torture tests	4
2.5	Runtime options	5
3	Commonalities	7
3.1	Option conventions	7
3.2	Common options	7
3.3	Path searching	9
3.4	Output file location	9
3.5	Three programs: Metafont, MetaPost, and T _E X	10
3.5.1	Initial and virgin	10
3.5.2	Memory dumps	10
3.5.2.1	Creating memory dumps	10
3.5.2.2	Determining the memory dump to use	11
3.5.2.3	Hardware and memory dumps	11
3.5.3	Editor invocation	12
3.5.4	\input filenames	12
4	T_EX: Typesetting	14
4.1	tex invocation	14
4.2	Initial T _E X	16
4.3	Formats	17
4.4	Languages and hyphenation	18
4.4.1	MLT _E X: Multi-lingual T _E X	18
4.4.1.1	\charsubdef: Character substitutions	18
4.4.1.2	\tracingcharsubdef: Substitution diagnostics	19
4.4.2	TCX files: Character translations	19
4.4.3	Patgen: Creating hyphenation patterns	21
4.5	Shell escapes	21
4.6	IPC and T _E X	22
4.7	T _E X extensions	23

5	Metafont: Creating typeface families	24
5.1	mf invocation	24
5.2	Initial Metafont	25
5.3	Modes: Device definitions for Metafont	26
5.4	Online Metafont graphics	27
5.5	GFtoDVI: Character proofs of fonts	28
5.6	MFT: Prettyprinting Metafont source	29
6	MetaPost: Creating technical illustrations ..	31
6.1	mpost invocation	31
6.2	Initial MetaPost	33
6.3	DVItoMP: DVI to MPX conversion	34
7	BibT_EX: Bibliographies	35
7.1	BibT _E X invocation	35
7.2	Basic BibT _E X style files	36
8	WEB: Literate programming	37
8.1	Tangle: Translate WEB to Pascal	37
8.2	Weave: Translate WEB to T _E X	38
8.3	Pooltype: Display WEB pool files	38
9	DVI utilities	40
9.1	DVIcopy: Canonicalize virtual font references	40
9.2	DVItype: Plain text transliteration of DVI files	40
9.2.1	DVItype output example	41
10	Font utilities	44
10.1	Font file formats	44
10.2	GFtoPK: Generic to packed font conversion	45
10.3	PKtoGF: Packed to generic font conversion	45
10.4	PKtype: Plain text transliteration of packed fonts	45
10.5	GFtype: Plain text transliteration of generic fonts	47
10.6	TFtoPL: T _E X font metric to property list conversion	49
10.7	PLtoTF: Property list to T _E X font metric conversion	51
10.8	VFtoVP: Virtual font to virtual property lists	51
10.9	VPtoVF: Virtual property lists to virtual font	52
10.10	Font utilities available elsewhere	52
Appendix A	Legalisms	53
Appendix B	References	54
Index	56	

1 Introduction

This manual corresponds to version 2010 of Web2c, released in July 2010.

Web2c is the name of a \TeX implementation, originally for Unix, but now also running under DOS, Amiga, and other operating systems. By *\TeX implementation*, we mean all of the standard programs developed by the Stanford \TeX project directed by Donald E. Knuth: Metafont, DVItyp, GFtoDVI, Bib \TeX , Tangle, etc., as well as \TeX itself. Other programs are also included: DVICopy, written by Peter Breitenlohner, MetaPost and its utilities (derived from Metafont), by John Hobby, etc.

General strategy: Web2c works, as its name implies, by translating the WEB source in which \TeX is written into C source code. Its output is not self-contained, however; it makes extensive use of many macros and functions in a library (the ‘`web2c/lib`’ directory in the sources). Therefore, it will not work without change on an arbitrary WEB program.

Availability: All of Web2c is freely available—“free” both in the sense of no cost (free ice cream) and of having the source code to modify and/or redistribute (free speech). See [Section “unixtex.ftp” in *Kpathsea*](#), for the practical details of how to obtain Web2c.

Different parts of the Web2c distribution have different licensing terms, however, reflecting the different circumstances of their creation; consult each source file for exact details. The main practical implication for redistributors of Web2c is that the executables are covered by the GNU General Public License, and therefore anyone who gets a binary distribution must also get the sources, as explained by the terms of the GPL (see [Section “Copying” in *Kpathsea*](#)). The GPL covers the Web2c executables, including `tex`, because the Free Software Foundation sponsored the initial development of the Kpathsea library that Web2c uses. The basic source files from Stanford, however, have their own copyright terms or are in the public domain, and are not covered by the GPL.

History: Tomas Rokicki originated the \TeX -to-C system in 1987, working from the first change files for \TeX under Unix, which were done primarily by Howard Trickey and Pavel Curtis. Tim Morgan then took over development and maintenance for a number of years; the name changed to Web-to-C somewhere in there. In 1990, Karl Berry became the maintainer. He made many changes to the original sources, and started using the shorter name Web2c. In 1997, Olaf Weber took over. Dozens of other people have contributed; their names are listed in the ‘`ChangeLog`’ files.

Other acknowledgements: The University of Massachusetts at Boston (particularly Rick Martin and Bob Morris) provided computers and ftp access to me for many years. Richard Stallman at the Free Software Foundation employed me while I wrote the original path searching library (for the GNU font utilities). (rms also gave us Emacs, GDB, and GCC, without which I cannot imagine developing Web2c.) And, of course, \TeX would not exist in the first place without Donald E. Knuth.

Further reading: See [Appendix B \[References\]](#), page 54.

2 Installation

(A copy of this chapter is in the distribution file ‘web2c/INSTALL’.)

Installing Web2c is mostly the same as installing any other Kpathsea-using program. Therefore, for the basic steps involved, see [Section “Installation” in *Kpathsea*](#). (A copy is in the file ‘kpathsea/INSTALL’.)

One peculiarity to Web2c is that the source distribution comes in two files: ‘web.tar.gz’ and ‘web2c.tar.gz’. You must retrieve and unpack them both. (We have two because the former archive contains the very large and seldom-changing original WEB source files.) See [Section “unixtex.ftp” in *Kpathsea*](#).

Another peculiarity is the MetaPost program. Although it has been installed previously as `mp`, as of Web2c 7.0 the installed name is now `mpost`, to avoid conflict with the `mp` program that does prettyprinting. This approach was recommended by the MetaPost author, John Hobby. If you as the \TeX administrator wish to make it available under its shorter name as well, you will have to set up a link or some such yourself. And of course individual users can do the same.

For solutions to common installation problems and information on how to report a bug, see the file ‘kpathsea/BUGS’ (see [Section “Bugs” in *Kpathsea*](#)). See also the Web2c home page, <http://www.tug.org/web2c>.

Points worth repeating:

- Before starting the standard compilation and installation you must install the basic fonts, macros, and other library files. See [Section “Installation” in *Kpathsea*](#).
- If you do not wish to use the standard file locations, see [Section “Changing search paths” in *Kpathsea*](#).
- Some Web2c features are enabled or disabled at `configure` time, as described in the first section below.

2.1 configure options

This section gives pointers to descriptions of the ‘`--with`’ and ‘`--enable`’ `configure` arguments that Web2c accepts. Some are specific to Web2c, others are generic to all Kpathsea-using programs.

For a list of all the options `configure` accepts, run ‘`configure --help`’. The generic options are listed first, and the package-specific options come last.

For a description of the generic options (which mainly allow you to specify installation directories) and basic `configure` usage, see [Section “Running configure scripts” in *Autoconf*](#), a copy is in the file ‘kpathsea/CONFIGURE’.

‘`--disable-dump-share`’

Do not make `fmt/base/mem` files sharable across different endian architectures.
See [Section 3.5.2.3 \[Hardware and memory dumps\]](#), page 11.

```
'--without-maketexmf-default'
'--without-maketexpk-default'
'--without-maketextfm-default'
'--with-maketextex-default'
```

Enable or disable the dynamic generation programs. See [Section “mktex configuration” in *Kpathsea*](#). The defaults are the inverse of the options, i.e., everything is enabled except `mktextex`.

```
'--enable-shared'
```

Build Kpathsea as a shared library. See [Section “Shared library” in *Kpathsea*](#).

```
'--with-editor=cmd'
```

Change the default editor invoked by the ‘e’ interactive command. See [Section 3.5.3 \[Editor invocation\]](#), page 12.

```
'--with-epsfwin'
'--with-hp2627win'
'--with-mftalkwin'
'--with-nextwin'
'--with-regiswin'
'--with-suntoolswin'
'--with-tektronixwin'
'--with-unitermwin'
'--with-x'
'--with-x-toolkit=KIT'
'--with-x11win'
'--with-x11'
```

Define Metafont graphics support; by default, no graphics support is enabled. See [Section 5.4 \[Online Metafont graphics\]](#), page 27.

```
'--x-includes=dir'
```

```
'--x-libraries=dir'
```

Define the locations of the X11 include files and libraries; by default, `configure` does its best to guess). See [Section “Optional Features” in *Autoconf*](#). A copy is in ‘`kpathsea/CONFIGURE`’.

2.2 Compile-time options

In addition to the `configure` options listed in the previous section, there are a few things that can be affected at compile-time with C definitions, rather than with `configure`. Using any of these is unusual.

To specify extra compiler flags (‘`-Dname`’ in this case), the simplest thing to do is:

```
make XCFLAGS="coptions"
```

You can also set the `CFLAGS` environment variable before running `configure`. See [Section “configure environment” in *Kpathsea*](#).

Anyway, here are the possibilities:

`‘-DFIXPT’`

`‘-DNO_MF_ASM’`

Use the original WEB fixed-point routines for Metafont and MetaPost arithmetic calculations regarding fractions. By default, assembly-language routines are used on x86 hardware with GNU C (unless `‘NO_MF_ASM’` is defined), and floating-point routines are used otherwise.

`‘-DIPC_DEBUG’`

Report on various interprocess communication activities. See [Section 4.6 \[IPC and T_EX\]](#), page 22.

2.3 Additional targets

Web2c has several Make targets besides the standard ones. You can invoke these either in the top level directory of the source distribution (the one containing `‘kpathsea/’` and `‘web2c/’`), or in the `‘web2c/’` directory.

`‘c-sources’`

Make only the C files, translated from the Web sources, presumably because you want to take them to a non-Unix machine.

`‘formats’`

`‘install-formats’`

Make or install all the memory dumps (see [Section 3.5.2 \[Memory dumps\]](#), page 10). By default, the standard plain formats plus `‘latex.fmt’` are made. You can add other formats by redefining the `fmts`, `bases`, and `mems` variables. See the top of `‘web2c/Makefile’` for the possibilities.

`‘fmts’`

`‘install-fmts’`

Make or install the T_EX `‘.fmt’` files. See [Section 4.2 \[Initial T_EX\]](#), page 16.

`‘bases’`

`‘install-bases’`

Make or install the Metafont `‘.base’` files. See [Section 5.2 \[Initial Metafont\]](#), page 25.

`‘mems’`

`‘install-mems’`

Make or install the MetaPost `‘.mem’` files. See [Section 6.2 \[Initial MetaPost\]](#), page 33.

`‘triptrap’`

`‘trip’`

`‘trap’`

`‘mptrap’`

To run the torture tests for T_EX, Metafont, and MetaPost (respectively). See the next section.

2.4 Trip, trap, and mptrap: Torture tests

To validate your T_EX, Metafont, and MetaPost executables, run `‘make triptrap’`. This runs the trip, trap, and mptrap “torture tests”. See the files `‘triptrap/tripman.tex’`,

`'triptrap/trapman.tex'`, and `'triptrap/mptrap.readme'` for detailed information and background on the tests.

The differences between your executables' behavior and the standard values will show up on your terminal. The usual differences (these are all acceptable) are:

- string usage and table sizes;
- glue set ratios;
- `'down4'`, `'right4'`, and `'y4'` commands in DVItypex output;
- dates and times.

Any other differences are trouble. The most common culprit in the past has been compiler bugs, especially when optimizing. See [Section "T_EX or Metafont failing" in *Kpathsea*](#).

The files `'trip.diffs'`, `'mftrap.diffs'`, and `'mptrap.diffs'` in the `'triptrap'` directory show the standard diffs against the original output. If you diff your diffs against these files, you should come up clean. For example

```
make trip >&mytrip.diffs
diff triptrap/trip.diffs mytrip.diffs
```

To run the tests separately, use the targets `trip`, `trap`, and `mptrap`.

To run simple tests for all the programs as well as the torture tests, run `'make check'`. You can compare the output to the distributed file `'tests/check.log'` if you like.

2.5 Runtime options

Besides the configure- and compile-time options described in the previous sections, you can control a number of parameters (in particular, array sizes) in the `'texmf.cnf'` runtime file read by *Kpathsea* (see [Section "Config files" in *Kpathsea*](#)).

Rather than exhaustively listing them here, please see the last section of the distributed `'kpathsea/texmf.cnf'`. Some of the more interesting values:

`'main_memory'`

Total words of memory available, for T_EX, Metafont, and MetaPost. Must remake the format file after changing.

`'extra_mem_bot'`

Extra space for "large" T_EX data structures: boxes, glue, breakpoints, et al. If you use PiCT_EX, you may well want to set this.

`'font_mem_size'`

Words of font info available for T_EX; this is approximately the total size of all TFM files read.

`'hash_extra'`

Additional space for the hash table of control sequence names. Approximately 10,000 control sequences can be stored in the main hash table; if you have a large book with numerous cross-references, this might not be enough, and thus you will want to set `hash_extra`.

Of course, ideally all arrays would be dynamically expanded as necessary, so the only limiting factor would be the amount of swap space available. Unfortunately, implementing

this is extremely difficult, as the fixed size of arrays is assumed in many places throughout the source code. These runtime limits are a practical compromise between the compile-time limits in previous versions, and truly dynamic arrays. (On the other hand, the Web2c BibTeX implementation does do dynamic reallocation of some arrays.)

3 Commonalities

Many aspects of the \TeX system are the same among more than one program, so we describe all those pieces together, here.

3.1 Option conventions

To provide a clean and consistent behavior, we chose to have all these programs use the GNU function `getopt_long_only` to parse command lines. However, we do use in a restricted mode, where all the options have to come before the rest of the arguments.

As a result, you can:

- use ‘-’ or ‘--’ to start an option name;
- use any unambiguous abbreviation for an option name;
- separate option names and values with either ‘=’ or one or more spaces;
- use filenames that would otherwise look like options by putting them after an option ‘--’.

By convention, non-option arguments, if specified, generally define the name of an input file, as documented for each program.

If a particular option with a value is given more than once, it is the last value that counts.

For example, the following command line specifies the options ‘foo’, ‘bar’, and ‘verbose’; gives the value ‘baz’ to the ‘abc’ option, and the value ‘xyz’ to the ‘quux’ option; and specifies the filename ‘-myfile-’.

```
-foo --bar -verb -abc=baz -quux karl --quux xyz -- -myfile-
```

3.2 Common options

All of these programs accept the standard GNU ‘--help’ and ‘--version’ options, and several programs accept ‘--verbose’. Rather than writing identical descriptions for every program, they are described here.

‘--help’ Print a usage message listing basic usage and all available options to standard output, then exit successfully.

‘--verbose’
 Print progress reports to standard output.

‘--version’
 Print the version number to standard output, then exit successfully.

\TeX , Metafont, and MetaPost have a number of additional options in common:

‘-file-line-error’

‘-no-file-line-error’

Change (or do not change) the way error messages are printed. The alternate style looks like error messages from many compilers and is easier to parse for some editors that invoke \TeX . This option used to be called ‘-file-line-error-style’.

‘-fmt=*dumpname*’
 ‘-base=*dumpname*’
 ‘-mem=*dumpname*’

Use *dumpname* instead of the program name or a ‘%&’ line to determine the name of the memory dump file read (‘fmt’ for T_EX, ‘base’ for Metafont, ‘mem’ for MetaPost). See [Section 3.5.2 \[Memory dumps\]](#), page 10. Also sets the program name to *dumpname* if no ‘-progrname’ option was given.

‘-halt-on-error’

Stop processing and exit when an error occurs, as opposed to the normal process of trying to recover and continue.

‘-ini’ Enable the “initial” form of the program (see [Section 3.5.1 \[Initial and virgin\]](#), page 10). This is implicitly set if the program name is *initex* resp. *inimf* resp. *inimpost*, although these variants are no longer typically installed.

‘-interaction=*string*’

Set the interaction mode from the command line. The *string* must be one of ‘batchmode’, ‘nonstopmode’, ‘scrollmode’, or ‘errorstopmode’.

‘-jobname=*string*’

Set the job name to *string*, instead of deriving it from the name of the input file.

‘-kpathsea-debug=*number*’

Set path searching debugging flags according to the bits of *number* (see [Section “Debugging” in Kpathsea](#)). You can also specify this in KPATHSEA_DEBUG environment variable (for all Web2c programs). (The command line value overrides.) The most useful value is ‘-1’, to get all available output.

‘-output-directory=*dirname*’

Specify the directory *dirname* to which output files are written. Also look for input files in *dirname* first, before looking along the normal search path. This is useful when you are in some read-only distribution directory, perhaps on a CD-ROM, and want to T_EX some documentation, for example. Note that for input files the “search” in *dirname* does not use the full generality of the search mechanism. This means that some files are not found there even though you might expect them to be.

‘-parse-first-line’

‘-no-parse-first-line’

Check or disable checking whether the first line of the main input file starts with ‘%&’, and parse it if it does. This line can be used specify the format and/or a TCX file.

‘-progrname=*string*’

Set program (and memory dump) name to *string*. This may affect the search paths and other values used (see [Section “Config files” in Kpathsea](#)). Using this option is equivalent to making a link named *string* to the binary and then invoking the binary under that name. See [Section 3.5.2 \[Memory dumps\]](#), page 10.

‘-recorder’

Enable the filename recorder. This makes the program save a list of the opened files into a file with (by default) extension ‘.fls’. For Aleph, this option is always on, and the file has extension ‘.ofl’.

‘-translate-file=tcxfile’

Use *tcxfile* to define which characters are printable and translations between the internal and external character sets. Moreover, *tcxfile* can be explicitly declared in the first line of the main input file ‘`%& -translate-file=tcxfile`’. This is the recommended method for portability reasons. See [Section 4.4.2 \[TCX files\]](#), page 19.

‘-8bit’

This option specifies that by default all characters should be considered printable. If ‘-translate-file’ was given as well, then the TCX file may mark characters as non-printable.

‘-oem’

This option is specific to Windows. When specified, T_EX engines will use the OEM code page rather than the ANSI one to display their messages.

3.3 Path searching

All of the Web2c programs, including T_EX, which do path searching use the Kpathsea routines to do so. The precise names of the environment and configuration file variables which get searched for particular file formatted are therefore documented in the Kpathsea manual (see [Section “Supported file formats” in Kpathsea](#)). Reading ‘*texmf.cnf*’ (see [Section “Config files” in Kpathsea](#)), invoking *mktex*... scripts (see [Section “mktex scripts” in Kpathsea](#)), and so on are all handled by Kpathsea.

The programs which read fonts make use of another Kpathsea feature: ‘*texfonts.map*’, which allows arbitrary aliases for the actual names of font files; for example, ‘*Times-Roman*’ for ‘*ptmr8r.tfm*’. The distributed (and installed by default) ‘*texfonts.map*’ includes aliases for many widely available PostScript fonts by their PostScript names.

3.4 Output file location

All the programs generally follow the usual convention for output files. Namely, they are placed in the directory current when the program is run, regardless of any input file location; or, in a few cases, output is to standard output.

For example, if you run ‘*tex /tmp/foo*’, for example, the output will be in ‘*./foo.dvi*’ and ‘*./foo.log*’, not ‘*/tmp/foo.dvi*’ and ‘*/tmp/foo.log*’.

You can use the ‘-output-directory’ option to cause all output files that would normally be written in the current directory to be written in the specified directory instead. See [Section 3.2 \[Common options\]](#), page 7.

If the current directory is not writable, and ‘-output-directory’ is not specified, the main programs (T_EX, Metafont, MetaPost, and BibT_EX) make an exception: if the config file value *TEXMFOUTPUT* is set (it is not by default), output files are written to the directory specified.

TEXMFOUTPUT is also checked for input files, as T_EX often generates files that need to be subsequently read; for input, no suffixes (such as ‘.tex’) are added by default, the input name is simply checked as given.

3.5 Three programs: Metafont, MetaPost, and T_EX

T_EX, Metafont, and MetaPost have a number of features in common. Besides the ones here, the common command-line options are described in the previous section. The configuration file options that let you control some array sizes and other features are described in [Section 2.5 \[Runtime options\]](#), page 5.

3.5.1 Initial and virgin

The T_EX, Metafont, and MetaPost programs each have two main variants, called *initial* and *virgin*. As of Web2c 7, one executable suffices for both variants, and in fact, the `ini...` executables are no longer created.

The initial form is enabled if:

1. the ‘`-ini`’ option was specified; or
2. the program name is ‘`initex`’ resp. ‘`inimf`’ resp. ‘`inimpost`’ (these variants are no longer typically installed); or
3. the first line of the main input file is ‘`%&ini`’;

otherwise, the virgin form is used.

The *virgin* form is the one generally invoked for production use. The first thing it does is read a memory dump (see [Section 3.5.2.2 \[Determining the memory dump to use\]](#), page 11), and then proceeds on with the main job.

The *initial* form is generally used only to create memory dumps (see the next section). It starts up more slowly than the virgin form, because it must do lengthy initializations that are encapsulated in the memory dump file.

3.5.2 Memory dumps

In typical use, T_EX, Metafont, and MetaPost require a large number of macros to be predefined; therefore, they support *memory dump* files, which can be read much more efficiently than ordinary source code.

3.5.2.1 Creating memory dumps

The programs all create memory dumps in slightly idiosyncratic (though substantially similar) way, so we describe the details in separate sections (references below). The basic idea is to run the initial version of the program (see [Section 3.5.1 \[Initial and virgin\]](#), page 10), read the source file to define the macros, and then execute the `\dump` primitive.

Also, each program uses a different filename extension for its memory dumps, since although they are completely analogous they are not interchangeable (T_EX cannot read a Metafont memory dump, for example).

Here is a list of filename extensions with references to examples of creating memory dumps:

T_EX (‘`.fmt`’) See [Section 4.2 \[Initial T_EX\]](#), page 16.

Metafont (‘`.base`’) See [Section 5.2 \[Initial Metafont\]](#), page 25.

MetaPost (‘`.mem`’) See [Section 6.2 \[Initial MetaPost\]](#), page 33.

When making memory dumps, the programs read environment variables and configuration files for path searching and other values as usual. If you are making a new installation and have environment variables pointing to an old one, for example, you will probably run into difficulties.

3.5.2.2 Determining the memory dump to use

The virgin form (see [Section 3.5.1 \[Initial and virgin\], page 10](#)) of each program always reads a memory dump before processing normal source input. All three programs determine the memory dump to use in the same way:

1. If the first non-option command-line argument begins with ‘&’, the program uses the remainder of that argument as the memory dump name. For example, running ‘`tex \&super`’ reads ‘`super.fmt`’. (The backslash protects the ‘&’ against interpretation by the shell.)
2. If the ‘`-fmt`’ resp. ‘`-base`’ resp. ‘`-mem`’ option is specified, its value is used.
3. If the ‘`-progrname`’ option is specified, its value is used.
4. If the first line of the main input file (which must be specified on the command line, not in response to ‘`**`’) is `%&dump`, and *dump* is an existing memory dump of the appropriate type, *dump* is used.

The first line of the main input file can also specify which character translation file is to be used: `%&-translate-file=tcxfile` (see [Section 4.4.2 \[TCX files\], page 19](#)).

These two roles can be combined: `%&dump -translate-file=tcxfile`. If this is done, the name of the dump must be given first.

5. Otherwise, the program uses the program invocation name, most commonly ‘`tex`’ resp. ‘`mf`’ resp. ‘`mpost`’. For example, if ‘`latex`’ is a link to ‘`tex`’, and the user runs ‘`latex foo`’, ‘`latex.fmt`’ will be used.

3.5.2.3 Hardware and memory dumps

By default, memory dump files are generally sharable between architectures of different types; specifically, on machines of different endianness (see [Section “Byte order” in GNU C Library](#)). (This is a feature of the Web2c implementation, and is not true of all T_EX implementations.) If you specify ‘`--disable-dump-share`’ to `configure`, however, memory dumps will be endian-dependent.

The reason to do this is speed. To achieve endian-independence, the reading of memory dumps on LittleEndian architectures, such as PC’s and DEC architectures, is somewhat slowed (all the multibyte values have to be swapped). Usually, this is not noticeable, and the advantage of being able to share memory dumps across all platforms at a site far outweighs the speed loss. But if you’re installing Web2c for use on LittleEndian machines only, perhaps on a PC being used only by you, you may wish to get maximum speed.

T_EXnically, even without ‘`--disable-dump-share`’, sharing of ‘`.fmt`’ files cannot be guaranteed to work. Floating-point values are always written in native format, and hence will generally not be readable across platforms. Fortunately, T_EX uses floating point only to represent glue ratios, and all common formats (plain, L^AT_EX, A^MS^T_EX, ...) do not do any glue setting at ‘`.fmt`’-creation time. Metafont and MetaPost do not use floating point in any dumped value at all.

Incidentally, different memory dump files will never compare equal byte-for-byte, because the program always dumps the current date and time. So don't be alarmed by just a few bytes difference.

If you don't know what endianness your machine is, and you're curious, here is a little C program to tell you. (The `configure` script contains a similar program.) This is from the book *C: A Reference Manual*, by Samuel P. Harbison and Guy L. Steele Jr. (see [Appendix B \[References\]](#), page 54).

```
main ()
{
    /* Are we little or big endian?  From Harbison&Steele.  */
    union
    {
        long l;
        char c[sizeof (long)];
    } u;
    u.l = 1;
    if (u.c[0] == 1)
        printf ("LittleEndian\n");
    else if (u.c[sizeof (long) - 1] == 1)
        printf ("BigEndian\n");
    else
        printf ("unknownEndian");

    exit (u.c[sizeof (long) - 1] == 1);
}
```

3.5.3 Editor invocation

`TeX`, `Metafont`, and `MetaPost` all (by default) stop and ask for user intervention at an error. If the user responds with `e` or `E`, the program invokes an editor.

Specifying `--with-editor=cmd` to `configure` sets the default editor command string to `cmd`. The environment variables/configuration values `TEXEDIT`, `MFEDIT`, and `MPEDIT` (respectively) override this. If `--with-editor` is not specified, the default is `vi +%d %s`.

In this string, `%d` is replaced by the line number of the error, and `%s` is replaced by the name of the current input file.

3.5.4 `\input` filenames

`TeX`, `Metafont`, and `MetaPost` source programs can all read other source files with the `\input` (`TeX`) and `input` (`MF` and `MP`) primitives:

```
\input name % in TeX
```

The file *name* can always be terminated with whitespace; for `Metafont` and `MetaPost`, the statement terminator `;` also works. (`LaTeX` and other macro packages provide other interfaces to `\input` that allow different notation; here we are concerned only with the primitive operation.)

As of `Web2c` version 7.5.3, double-quote characters can be used to include spaces or other special cases. In typical use, the `"` characters surround the entire filename:


```
\input "filename with spaces"
```

Technically, the quote characters can be used inside the name, and can enclose any characters, as in:

```
\input filename" "with" "spaces"
```

One more point. In \LaTeX , the quotes are needed inside the braces, thus

```
\input{a b}      % fails
\input{"a b"}    % ok
```

This quoting mechanism comes into play *after* \TeX has tokenized and expanded the input. So, multiple spaces and tabs may be seen as a single space, active characters such as ‘~’ are expanded first, and so on. (See below.)

On the other hand, various C library routines and Unix itself use the null byte (character code zero, ASCII NUL) to terminate strings. So filenames in Web2c cannot contain nulls, even though \TeX itself does not treat NUL specially. In addition, some older Unix variants do not allow eight-bit characters (codes 128–255) in filenames.

For maximal portability of your document across systems, use only the characters ‘a’–‘z’, ‘0’–‘9’, and ‘.’, and restrict your filenames to at most eight characters (not including the extension), and at most a three-character extension. Do not use anything but simple filenames, since directory separators vary among systems; instead, add the necessary directories to the appropriate search path.

Finally, the present Web2c implementation does ‘~’ and ‘\$’ expansion on *name*, unlike Knuth’s original implementation and older versions of Web2c. Thus:

```
\input ~jsmith/$foo.bar
```

will dereference the environment variable or Kpathsea config file value ‘foo’ and read that file extended with ‘.bar’ in user ‘jsmith’’s home directory. You can also use braces, as in ‘\${foo}bar’, if you want to follow the variable name with a letter, numeral, or ‘_’.

(So another way to get a program to read a filename containing whitespace is to define an environment variable and dereference it.)

In all the common \TeX formats (plain \TeX , \LaTeX , \AMSTeX), the characters ‘~’ and ‘\$’ have special category codes, so to actually use these in a document you have to change their catcodes or use `\string`. (The result is unportable anyway, see the suggestions above.) The place where they are most likely to be useful is when typing interactively.

4 T_EX: Typesetting

T_EX is a typesetting system: it was especially designed to handle complex mathematics, as well as most ordinary text typesetting.

T_EX is a batch language, like C or Pascal, and not an interactive “word processor”: you compile a T_EX input file into a corresponding device-independent (DVI) file (and then translate the DVI file to the commands for a particular output device). This approach has both considerable disadvantages and considerable advantages. For a complete description of the T_EX language, see *The T_EXbook* (see [Appendix B \[References\]](#), page 54). Many other books on T_EX, introductory and otherwise, are available.

4.1 tex invocation

T_EX (usually invoked as `tex`) formats the given text and commands, and outputs a corresponding device-independent representation of the typeset document. This section merely describes the options available in the Web2c implementation. For a complete description of the T_EX typesetting language, see *The T_EXbook* (see [Appendix B \[References\]](#), page 54).

T_EX, Metafont, and MetaPost process the command line (described here) and determine their memory dump (fnt) file in the same way (see [Section 3.5.2 \[Memory dumps\]](#), page 10). Synopses:

```
tex [option]... [texname[.tex]] [tex-commands]
tex [option]... \first-line
tex [option]... &fmt args
```

T_EX searches the usual places for the main input file *texname* (see [Section “Supported file formats” in Kpathsea](#)), extending *texname* with `.tex` if necessary. To see all the relevant paths, set the environment variable `KPATHSEA_DEBUG` to `-1` before running the program.

After *texname* is read, T_EX processes any remaining *tex-commands* on the command line as regular T_EX input. Also, if the first non-option argument begins with a T_EX escape character (usually `\`), T_EX processes all non-option command-line arguments as a line of regular T_EX input.

If no arguments or options are specified, T_EX prompts for an input file name with `**`.

T_EX writes the main DVI output to the file `basetexname.dvi`, where *basetexname* is the basename of *texname*, or `texput` if no input file was specified. A DVI file is a device-independent binary representation of your T_EX document. The idea is that after running T_EX, you translate the DVI file using a separate program to the commands for a particular output device, such as a PostScript printer (see [Section “Introduction” in Dvips](#)) or an X Window System display (see `xdvi(1)`).

T_EX also reads TFM files for any fonts you load in your document with the `\font` primitive. By default, it runs an external program named `mktexfm` to create any nonexistent TFM files. You can disable this at configure-time or runtime (see [Section “mktex configuration” in Kpathsea](#)). This is enabled mostly for the sake of the EC fonts, which can be generated at any size.

T_EX can write output files, via the `\openout` primitive; this opens a security hole vulnerable to Trojan horse attack: an unwitting user could run a T_EX program that overwrites,

say, ‘~/rhosts’. (MetaPost has a `write` primitive with similar implications). To alleviate this and similar problems the functions `kpathsea_out_name_ok` and `kpathsea_in_name_ok` from the Kpathsea library (see [Section “Calling sequence” in Kpathsea](#)) are used to determine if a given filename is acceptable to be opened for output or input, depending on the setting of the configuration variables `openout_any` and `openin_any`: ‘a’ (for “any”, the default for `openin_any`), ‘r’ (for “restricted”), or ‘p’ (for “paranoid”, the default for `openout_any`).

In any case, all `\openout` filenames are recorded in the log file, except those opened on the first line of input, which is processed when the log file has not yet been opened.

The program accepts the following options, as well as the standard ‘-help’ and ‘-version’ (see [Section 3.2 \[Common options\], page 7](#)):

```

'-enc'
'-[no]-file-line-error'
'-fmt=fmtname'
'-halt-on-error'
'-ini'
'-interaction=string'
'-ipc'
'-ipc-start'
'-jobname=string'
'-kpathsea-debug=number'
'-[no]parse-first-line'
'-output-directory'
'-progname=string'
'-recorder'
'-translate-file=tcxfile'
'-8bit'
'-enc'
'-ipc'
'-ipc-start'

```

These options are common to T_EX, Metafont, and MetaPost. See [Section 3.2 \[Common options\], page 7](#).

Enable encT_EX extensions, such as `\mubyte`. This can be used to support Unicode UTF-8 input encoding. See <http://www.olsak.net/enc tex.html>.

With either option, T_EX writes its DVI output to a socket as well as to the usual ‘.dvi’ file. With ‘-ipc-start’, T_EX also opens a server program at the other end to read the output. See [Section 4.6 \[IPC and T_EX\], page 22](#).

These options are available only if the ‘--enable-ipc’ option was specified to `configure` during installation of Web2c.

```

'-mktex=filetype'
'-no-mktex=filetype'

```

Turn on or off the ‘mktex’ script associated with *filetype*. For T_EX proper, *filetype* can only be ‘tex’ and ‘tfm’, but for pdfT_EX and luaT_EX, it can also be ‘pk’.

```

'-mltex'

```

If we are INITEX (see [Section 3.5.1 \[Initial and virgin\], page 10](#)), enable MLT_EX extensions such as `\charsubdef`. Implicitly set if the program name is `mltex`. See [Section 4.4.1 \[MLT_EX\], page 18](#).

`'-output-comment=string'`

Use *string* as the DVI file comment. Ordinarily, this comment records the date and time of the T_EX run, but if you are doing regression testing, you may not want the DVI file to have this spurious difference. This is also taken from the environment variable and config file value `'output_comment'`.

`'-shell-escape'`

`'-no-shell-escape'`

`'-shell-restricted'`

Enable, or disable, or enable with restrictions the `\write18{shell-command}` feature for external executing shell commands. See [Section 4.5 \[Shell escapes\]](#), [page 21](#).

`'-enable-write18'`

`'-disable-write18'`

Synonyms for `'-shell-escape'` and `'-no-shell-escape'`, for compatibility with MiK_TE_X. (MiK_TE_X also accepts both pairs of options.) See [Section 4.5 \[Shell escapes\]](#), [page 21](#).

`'-src-specials'`

`'-src-specials=string'`

This option makes T_EX output specific source information using `'\special'` commands in the DVI file. These `'\special'` track the current file name and line number.

Using the first form of this option, the `'\special'` commands are inserted automatically.

In the second form of the option, *string* is a comma separated list of the following values: `'cr'`, `'display'`, `'hbox'`, `'math'`, `'par'`, `'parent'`, `'vbox'`. You can use this list to specify where you want T_EX to output such commands. For example, `'-src-specials=cr,math'` will output source information every line and every math formula.

These commands can be used with the appropriate DVI viewer and text editor to switch from the current position in the editor to the same position in the viewer and back from the viewer to the editor.

This option works by inserting `'\special'` commands into the token stream, and thus in principle these additional tokens can be recovered or seen by the tricky-enough macros. If you run across a case, let us know, because this counts as a bug. However, such bugs are very hard to fix, requiring significant changes to T_EX, so please don't count on it.

Redefining `'\special'` will not affect the functioning of this option. The commands inserted into the token stream are hard-coded to always use the `'\special'` primitive.

T_EX does not pass the trip test when this option is enabled.

4.2 Initial T_EX

The *initial* form of T_EX is invoked by `'tex -ini'`. It does lengthy initializations avoided by the “virgin” (`vir`) form, so as to be capable of dumping `'fmt'` files (see [Section 3.5.2](#)

[Memory dumps], page 10). For a detailed comparison of virgin and initial forms, see Section 3.5.1 [Initial and virgin], page 10. In past releases, a separate program `initex` was installed to invoke the initial form, but this is no longer the case.

For a list of options and other information, see Section 4.1 [tex invocation], page 14.

Unlike Metafont and MetaPost, many format files are commonly used with T_EX. The standard one implementing the features described in the *T_EXbook* is ‘`plain.fmt`’, also known as ‘`tex.fmt`’ (again, see Section 3.5.2 [Memory dumps], page 10). It is created by default during installation, but you can also do so by hand if necessary (e.g., if an update to ‘`plain.tex`’ is issued):

```
tex -ini '\input plain \dump'
```

(The quotes prevent interpretation of the backslashes from the shell.) Then install the resulting ‘`plain.fmt`’ in ‘`$(fmdir)`’ (‘`/usr/local/share/texmf/web2c`’ by default), and link ‘`tex.fmt`’ to it.

The necessary invocation for generating a format file differs for each format, so instructions that come with the format should explain. The top-level ‘`web2c`’ Makefile has targets for making most common formats: `plain latex amstex texinfo eplain`. See Section 4.3 [Formats], page 17, for more details on T_EX formats.

4.3 Formats

T_EX *formats* are large collections of macros, often dumped into a ‘`.fmt`’ file (see Section 3.5.2 [Memory dumps], page 10) by `tex -ini` (see Section 4.2 [Initial TeX], page 16). A number of formats are in reasonably widespread use, and the Web2c Makefile has targets to make the versions current at the time of release. You can change which formats are automatically built by setting the `fmts` Make variable; by default, only the ‘`plain`’ and ‘`latex`’ formats are made.

You can get the latest versions of most of these formats from the CTAN archives in subdirectories of ‘`CTAN:/macros`’ (for CTAN info, see Section “`unixtex.ftp`” in *Kpathsea*). The archive [ftp://ftp.tug.org/tex/lib.tar.gz](http://ftp.tug.org/tex/lib.tar.gz) (also available from CTAN) contains most of these formats (although perhaps not the absolute latest version), among other things.

- | | |
|----------------|---|
| latex | The most widely used format. The current release is named ‘L ^A T _E X 2 _ε ’; new versions are released approximately every six months, with patches issued as needed. The old release was called ‘L ^A T _E X 2.09’, and is no longer maintained or supported. L ^A T _E X attempts to provide generic markup instructions, such as “emphasize”, instead of specific typesetting instructions, such as “use the 10pt Computer Modern italic font”. The L ^A T _E X home page: http://www.latex-project.org . |
| context | ConT _E Xt is an independent macro package which has a basic document structuring approach similar to L ^A T _E X. It also supports creating interactive PDF files and has integrated MetaPost support, among many other interesting features. The ConT _E Xt home page: http://www.pragma-ade.com . |
| amstex | The official typesetting system of the American Mathematical Society. Like L ^A T _E X, it encourages generic markup commands. The AMS also provides many L ^A T _E X package for authors who prefer L ^A T _E X. Taken together, they are used |

to produce nearly all AMS publications, e.g., *Mathematical Reviews*. The AMST_EX home page: <http://www.ams.org/tex>.

- texinfo** The documentation system developed and maintained by the Free Software Foundation for their software manuals. It can be automatically converted into plain text, a machine-readable on-line format called ‘info’, HTML, etc. The Texinfo home page: <http://www.gnu.org/software/texinfo>.
- eplain** The “expanded plain” format provides various common features (e.g., symbolic cross-referencing, tables of contents, indexing, citations using BibT_EX), for those authors who prefer to handle their own high-level formatting. The Eplain home page: <http://www.tug.org/eplain>.
- slitex** An obsolete L^AT_EX 2.09 format for making slides. It is replaced by the ‘slides’ document class, along with the ‘beamer’, ‘texpower’, and other packages.

4.4 Languages and hyphenation

T_EX supports most natural languages. See also [Section 4.7 \[T_EX extensions\]](#), page 23.

4.4.1 MLT_EX: Multi-lingual T_EX

Multi-lingual T_EX (`mltex`) is an extension of T_EX originally written by Michael Ferguson and now updated and maintained by Bernd Raichle. It allows the use of non-existing glyphs in a font by declaring glyph substitutions. These are restricted to substitutions of an accented character glyph, which need not be defined in the current font, by its appropriate `\accent` construction using a base and accent character glyph, which do have to exist in the current font. This substitution is automatically done behind the scenes, if necessary, and thus MLT_EX additionally supports hyphenation of words containing an accented character glyph for fonts missing this glyph (e.g., Computer Modern). Standard T_EX suppresses hyphenation in this case.

MLT_EX works at ‘.fmt’-creation time: the basic idea is to specify the ‘-mltex’ option to T_EX when you `\dump` a format. Then, when you subsequently invoke T_EX and read that .fmt file, the MLT_EX features described below will be enabled.

Generally, you use special macro files to create an MLT_EX .fmt file.

The sections below describe the two new primitives that MLT_EX defines. Aside from these, MLT_EX is completely compatible with standard T_EX.

4.4.1.1 `\charsubdef`: Character substitutions

The most important primitive MLT_EX adds is `\charsubdef`, used in a way reminiscent of `\chardef`:

```
\charsubdef composite [=] accent base
```

Each of *composite*, *accent*, and *base* are font glyph numbers, expressed in the usual T_EX syntax: ‘`\e`’ symbolically, ‘145’ for octal, ‘0x65’ for hex, ‘101’ for decimal.

MLT_EX’s `\charsubdef` declares how to construct an accented character glyph (not necessarily existing in the current font) using two character glyphs (that do exist). Thus it defines whether a character glyph code, either typed as a single character or using the `\char` primitive, will be mapped to a font glyph or to an `\accent` glyph construction.

For example, if you assume glyph code 138 (decimal) for an e-circumflex (ê) and you are using the Computer Modern fonts, which have the circumflex accent in position 18 and lowercase ‘e’ in the usual ASCII position 101 decimal, you would use `\charsubdef` as follows:

```
\charsubdef 138 = 18 101
```

For the plain T_EX format to make use of this substitution, you have to redefine the circumflex accent macro `\^` in such a way that if its argument is character ‘e’ the expansion `\char138` is used instead of `\accent18 e`. Similar `\charsubdef` declaration and macro redefinitions have to be done for all other accented characters.

To disable a previous `\charsubdef c`, redefine `c` as a pair of zeros. For example:

```
\charsubdef '321 = 0 0 % disable N tilde
```

(Octal ‘321 is the ISO Latin-1 value for the Spanish N tilde.)

`\charsubdef` commands should only be given once. Although in principle you can use `\charsubdef` at any time, the result is unspecified. If `\charsubdef` declarations are changed, usually either incorrect character dimensions will be used or MLT_EX will output missing character warnings. (The substitution of a `\charsubdef` is used by T_EX when appending the character node to the current horizontal list, to compute the width of a horizontal box when the box gets packed, and when building the `\accent` construction at `\shipout`-time. In summary, the substitution is accessed often, so changing it is not desirable, nor generally useful.)

4.4.1.2 `\tracingcharsubdef`: Substitution diagnostics

To help diagnose problems with ‘`\charsubdef`’, MLT_EX provides a new primitive parameter, `\tracingcharsubdef`. If positive, every use of `\charsubdef` will be reported. This can help track down when a character is redefined.

In addition, if the T_EX parameter `\tracinglostchars` is 100 or more, the character substitutions actually performed at `\shipout`-time will be recorded.

4.4.2 TCX files: Character translations

TCX (T_EX character translation) files help T_EX support direct input of 8-bit international characters if fonts containing those characters are being used. Specifically, they map an input (keyboard) character code to the internal T_EX character code (a superset of ASCII).

Of the various proposals for handling more than one input encoding, TCX files were chosen because they follow Knuth’s original ideas for the use of the ‘`xchr`’ and ‘`xord`’ tables. He ventured that these would be changed in the WEB source in order to adjust the actual version to a given environment. It turns out, however, that recompiling the WEB sources is not as simple a task as Knuth may have imagined; therefore, TCX files, providing the possibility of changing of the conversion tables on on-the-fly, have been implemented instead.

This approach limits the portability of T_EX documents, as some implementations do not support it (or use a different method for input-internal reencoding). It may also be problematic to determine the encoding to use for a T_EX document of unknown provenance; in the worst case, failure to do so correctly may result in subtle errors in the typeset output. But we feel the benefits outweigh these disadvantages.

This is entirely independent of the MLT_EX extension (see [Section 4.4.1 \[MLT_EX\]](#), [page 18](#)): whereas a TCX file defines how an input keyboard character is mapped to T_EX’s internal code, MLT_EX defines substitutions for a non-existing character glyph in a font with a `\accent` construction made out of two separate character glyphs. TCX files involve no new primitives; it is not possible to specify that an input (keyboard) character maps to more than one character.

Information on specifying TCX files:

- The best way to specify a TCX file is to list it explicitly in the first line of the main document:

```
%& -translate-file=tcxfile
```

- You can also specify a TCX file to be used on a particular T_EX run with the command-line option ‘`-translate-file=tcxfile`’.
- TCX files are searched for along the WEB2C path.
- Initial T_EX (see [Section 4.2 \[Initial T_EX\]](#), [page 16](#)) ignores TCX files.

The Web2c distribution comes with a number of TCX files. Two important ones are ‘`il1-t1.tcx`’ and ‘`il2-t1.tcx`’, which support ISO Latin 1 and ISO Latin 2, respectively, with Cork-encoded fonts (a.k.a. the L^AT_EX T1 encoding). TCX files for Czech, Polish, and Slovak are also provided.

One other notable TCX file is ‘`empty.tcx`’, which is, well, empty. Its purpose is to reset Web2C’s behavior to the default (only visible ASCII being printable, as described below) when a format was dumped with another TCX being active—which is in fact the case for everything but plain T_EX in the TeX Live and other distributions. Thus:

```
latex somefile8.tex
⇒ terminal etc. output with 8-bit chars
latex --translate-file=empty.tcx somefile8.tex
⇒ terminal etc. output with ^^ notation
```

Syntax of TCX files:

1. Line-oriented. Blank lines are ignored.
2. Whitespace is ignored except as a separator.
3. Comments start with ‘`%`’ and continue to the end of the line.
4. Otherwise, a line consists of one or two character codes, optionally followed by 0 or 1. The last number indicates whether *dest* is considered printable.

```
src [dest [prnt]]
```

5. Each character code may be specified in octal with a leading ‘`0`’, hexadecimal with a leading ‘`0x`’, or decimal otherwise. Values must be between 0 and 255, inclusive (decimal).
6. If the *dest* code is not specified, it is taken to be the same as *src*.
7. If the same *src* code is specified more than once, it is the last definition that counts.

Finally, here’s what happens: when T_EX sees an input character with code *src*, it 1) changes *src* to *dest*; and 2) makes the *dest* code “printable”, i.e., printed as-is in diagnostics and the log file rather than in ‘`^^`’ notation.

By default, no characters are translated, and character codes between 32 and 126 inclusive (decimal) are printable.

Specifying translations for the printable ASCII characters (codes 32–127) will yield unpredictable results. Additionally you shouldn't make the following characters printable: `^^I` (TAB), `^^J` (line feed), `^^M` (carriage return), and `^^?` (delete), since T_EX uses them in various ways.

Thus, the idea is to specify the input (keyboard) character code for *src*, and the output (font) character code for *dest*.

By default, only the printable ASCII characters are considered printable by T_EX. If you specify the `'-8bit'` option, all characters are considered printable by default. If you specify both the `'-8bit'` option and a TCX file, then the TCX can set specific characters to be non-printable.

Both the specified TCX encoding and whether characters are printable are saved in the dump files (like `'tex.fmt'`). So by giving these options in combination with `'-ini'`, you control the defaults seen by anyone who uses the resulting dump file.

When loading a dump, if the `'-8bit'` option was given, then all characters become printable by default.

When loading a dump, if a TCX file was specified, then the TCX data from the dump is ignored and the data from the file used instead.

4.4.3 Patgen: Creating hyphenation patterns

Patgen creates hyphenation patterns from dictionary files for use with T_EX. Synopsis:

```
patgen dictionary patterns output translate
```

Each argument is a filename. No path searching is done. The output is written to the file *output*.

In addition, Patgen prompts interactively for other values.

For more information, see *Word hyphen-a-tion by com-puter* by Frank Liang (see [Appendix B \[References\]](#), page 54), and also the `'patgen.web'` source file.

The only options are `'-help'` and `'-version'` (see [Section 3.2 \[Common options\]](#), page 7).

4.5 Shell escapes

T_EX can execute *shell escapes*, that is, arbitrary shell commands. Although tremendously useful, this also has obvious security implications. Therefore, as of T_EX Live 2009, a *restricted* mode for shell escapes is the default mode of operation, which allows executing only certain commands, as specified in the `'texmf.cnf'` configuration file.

- Unrestricted shell escapes are allowed if the option `'--shell-escape'` is specified, or if the environment variable or config file value `shell_escape` is set to `'t'` or `'y'` and `'1'`.
- Restricted shell escapes are allowed if `shell_escape` is set to `'p'`. This is the default.
- Shell escapes are completely disabled if `'--no-shell-escape'` is specified, or if `shell_escape` is set to anything else.

When enabled, the T_EX construct to execute a system command is `\write18{shell-command}`; for example:

```
\write18{echo "hello, world"}
```

From T_EX's point of view, this is a normal `\write` command, and is therefore subject to the usual T_EX expansions. Also, the system call either happens during the ‘`\output`’ routine or right away, according to the absence or presence of the `\immediate` prefix, as usual for `\write`.

The *shell-command* string is passed to the command shell (via the C library function `system`). The output of *shell-command* is not diverted anywhere, so it will not appear in the log file, or anywhere but the terminal output. The exit status of the system call is also not available to T_EX.

In unrestricted mode, the argument is simply passed straight to `system` unaltered.

In restricted mode, ASCII double quote characters (") should always be used in the argument to `\write18` where quoting of arguments is needed, as in the example above. This is to achieve some measure of system independence. On Unix systems, these are replaced with single quote (') characters to avoid insecure further expansion. Care is also taken on Windows to avoid additional expansions (from, e.g., ‘...’). Mismatched quotation marks in the command string result in a diagnostic message in the log file; no execution is performed.

After quotation processing, if the first word (delimited by a space or tab) of the command is in the list specified by the `shell_escape_commands` configuration value, the command is executed. Otherwise it is not. In any case, a message is written to the log file.

The `shell_escape_commands` value is a comma-separated list of words. Whitespace is significant, and typically should not be present. The default definition looks like this, but with more commands included:

```
shell_escape_commands = bibtex,dvips,epstopdf,...,tex
```

pdfT_EX and luaT_EX support reading (via `\input` and `\openin`) and writing (via `\openout`) from pipes if the first character is ‘|’. The following command is then treated exactly the same as the argument to `\write18`. In these engines, the primitive variable `\pdfshellescape` is set to 0 if shell escapes are disabled, 1 if they are enabled, and 2 if they are enabled with restrictions.

The purpose of this feature is to make it possible for T_EX documents to perform useful external actions in the common case of an individual user running a known document on his or her own machine. In such environments as CGI scripts or wikis where the input has to be considered untrustworthy, shell escapes should be completely disabled.

4.6 IPC and T_EX

(If anyone uses this feature and needs documentation, write tex-k@tug.org.)

This functionality is available only if the ‘`--enable-ipc`’ option was specified to `configure` during installation of Web2c (see [Chapter 2 \[Installation\]](#), page 2).

If you define `IPC_DEBUG` before compilation (e.g., with ‘`make XCFLAGS=-DIPC_DEBUG`’), T_EX will print messages to standard error about its socket operations. This may be helpful if you are, well, debugging.

4.7 T_EX extensions

The base T_EX program has been extended in many ways. Here's a partial list.

e-T _E X	Adds many new primitives, including right-to-left typesetting and more registers. Now frozen.
Aleph	This adds Unicode support, right-to-left typesetting, and more. Omega was the original program. Aleph is an updated version with a variety of bug fixes, and includes e-T _E X. Aleph is not actively maintained.
pdfT _E X	Can produce PDF as well as DVI files. It also incorporates the e-T _E X extensions, new primitives for hypertext and micro-typography, reading/writing from pipes, and much more. Home page: http://pdfTeX.org .
luaT _E X	Based on pdfT _E X, this also embeds the Lua programming language (http://lua.org) and opens up the T _E X typesetting engine to control from Lua. Home page: http://luatex.org .
XeT _E X	Combines support for Unicode input and OpenType- and system fonts with the capabilities of pdfT _E X. Home page: http://tug.org/xetex .

5 Metafont: Creating typeface families

Metafont is a system for producing shapes; it was designed for producing complete typeface families, but it can also produce geometric designs, dingbats, etc. And it has considerable mathematical and equation-solving capabilities which can be useful entirely on their own.

Metafont is a batch language, like C or Pascal: you compile a Metafont program into a corresponding font, rather than interactively drawing lines or curves. This approach has both considerable disadvantages (people unfamiliar with conventional programming languages will be unlikely to find it usable) and considerable advantages (you can make your design intentions specific and parameterizable). For a complete description of the Metafont language, see *The METAFONTbook* (see [Appendix B \[References\]](#), page 54).

5.1 mf invocation

Metafont (usually invoked as `mf`) reads character definitions specified in the Metafont programming language, and outputs the corresponding font. This section merely describes the options available in the Web2c implementation. For a complete description of the Metafont language, see *The Metafontbook* (see [Appendix B \[References\]](#), page 54).

Metafont processes its command line and determines its memory dump (base) file in a way exactly analogous to MetaPost and T_EX (see [Section 4.1 \[tex invocation\]](#), page 14, and see [Section 3.5.2 \[Memory dumps\]](#), page 10). Synopses:

```
mf [option]... [mfname[.mf]] [mf-commands]
mf [option]... \first-line
mf [option]... &base args
```

Most commonly, a Metafont invocation looks like this:

```
mf '\mode:=mode; mag:=magnification; input mfname'
```

(The single quotes avoid unwanted interpretation by the shell.)

Metafont searches the usual places for the main input file *mfname* (see [Section “Supported file formats” in Kpathsea](#)), extending *mfname* with `.mf` if necessary. To see all the relevant paths, set the environment variable `KPATHSEA_DEBUG` to `-1` before running the program. By default, Metafont runs an external program named `mktexmf` to create any nonexistent Metafont source files you input. You can disable this at configure-time or runtime (see [Section “mktex configuration” in Kpathsea](#)). This is mostly for the sake of the EC fonts, which can be generated at any size.

Metafont writes the main GF output to the file `‘basemfname.nnngf’`, where *nnn* is the font resolution in pixels per inch, and *basemfname* is the basename of *mfname*, or `‘mfput’` if no input file was specified. A GF file contains bitmaps of the actual character shapes. Usually GF files are converted immediately to PK files with GFtoPK (see [Section 10.2 \[gftopk invocation\]](#), page 45), since PK files contain equivalent information, but are more compact. (Metafont output in GF format rather than PK for only historical reasons.)

Metafont also usually writes a metric file in TFM format to `‘basemfname.tfm’`. A TFM file contains character dimensions, kerns, and ligatures, and spacing parameters. T_EX reads only this `.tfm` file, not the GF file.

The *mode* in the example command above is a name referring to a device definition (see [Section 5.3 \[Modes\]](#), page 26); for example, `localfont` or `ljfour`. These device definitions

must generally be precompiled into the base file. If you leave this out, the default is **proof** mode, as stated in *The Metafontbook*, in which Metafont outputs at a resolution of 2602 dpi; this is usually not what you want. The remedy is simply to assign a different mode—`localfont`, for example.

The *magnification* assignment in the example command above is a magnification factor; for example, if the device is 600 dpi and you specify `mag:=2`, Metafont will produce output at 1200 dpi. Very often, the *magnification* is an expression such as `magstep(.5)`, corresponding to a T_EX “magstep”, which are factors of $1.2\sqrt{2}$.

After running Metafont, you can use the font in a T_EX document as usual. For example:

```
\font\myfont = newfont
\myfont Now I am typesetting in my new font (minimum hamburgers).
```

The program accepts the following options, as well as the standard ‘`-help`’ and ‘`-version`’ (see [Section 3.2 \[Common options\]](#), page 7):

```
‘-[no]-file-line-error’
‘-fmt=fmtname’
‘-halt-on-error’
‘-ini’
‘-interaction=string’
‘-jobname=string’
‘-kpathsea-debug=number’
‘-[no]parse-first-line’
‘-output-directory’
‘-progname=string’
‘-recorder’
‘-translate-file=tcxfile’
‘-8bit’
‘-mktex=filetype’
‘-no-mktex=filetype’
```

These options are common to T_EX, Metafont, and MetaPost. See [Section 3.2 \[Common options\]](#), page 7.

Turn on or off the ‘`mktex`’ script associated with *filetype*. The only value that makes sense for *filetype* is ‘`mf`’.

5.2 Initial Metafont

`inimf` is the “initial” form of Metafont, which does lengthy initializations avoided by the “virgin” (`vir`) form, so as to be capable of dumping ‘`.base`’ files (see [Section 3.5.2 \[Memory dumps\]](#), page 10). For a detailed comparison of virgin and initial forms, see [Section 3.5.1 \[Initial and virgin\]](#), page 10. In past releases, a separate program `inimf` was installed to invoke the initial form, but this is no longer the case.

For a list of options and other information, see [Section 5.1 \[mf invocation\]](#), page 24.

The only memory dump file commonly used with Metafont is the default ‘`plain.base`’, also known as ‘`mf.base`’ (again, see [Section 3.5.2 \[Memory dumps\]](#), page 10). It is created by default during installation, but you can also do so by hand if necessary (e.g., if a Metafont update is issued):

```
mf -ini '\input plain; input modes; dump'
```

(The quotes prevent interpretation of the backslashes from the shell.) Then install the resulting ‘plain.base’ in ‘\$(basedir)’ (‘/usr/local/share/texmf/web2c’ by default), and link ‘mf.base’ to it.

For an explanation of the additional ‘modes.mf’ file, see [Section 5.3 \[Modes\]](#), page 26. This file has no counterpart in T_EX or MetaPost.

In the past, it was sometimes useful to create a base file ‘cmmf.base’ (a.k.a. ‘cm.base’), with the Computer Modern macros also included in the base file. Nowadays, however, the additional time required to read ‘cmbase.mf’ is exceedingly small, usually not enough to be worth the administrative hassle of updating the ‘cmmf.base’ file when you install a new version of ‘modes.mf’. People actually working on a typeface may still find it worthwhile to create their own base file, of course.

5.3 Modes: Device definitions for Metafont

Running Metafont and creating Metafont base files requires information that T_EX and MetaPost do not: *mode* definitions which specify device characteristics, so Metafont can properly rasterize the shapes.

When making a base file, a file containing modes for locally-available devices should be input after ‘plain.mf’. One commonly used file is <ftp://ftp.tug.org/tex/modes.mf>; it includes all known definitions.

If, however, for some reason you have decreased the memory available in your Metafont, you may need to copy ‘modes.mf’ and remove the definitions irrelevant to you (probably most of them) instead of using it directly. (Or, if you’re a Metafont hacker, maybe you can suggest a way to redefine `mode_def` and/or `mode_setup`; right now, the amount of memory used is approximately four times the total length of the `mode_def` names, and that’s a lot.)

If you have a device not included in ‘modes.mf’, please see comments in that file for how to create the new definition, and please send the definition to tex-fonts@math.utah.edu to get it included in the next release of ‘modes.mf’.

Usually, when you run Metafont you must supply the name of a mode that was dumped in the base file. But you can also define the mode characteristics dynamically, by invoking Metafont with an assignment to `smode` instead of `mode`, like this:

```
mf '\smode:="newmode.mf"; mag:=magnification; input mfname'
```

This is most useful when you are working on the definition of a new mode.

The *magnification* and *mfname* arguments are explained in [Section 5.1 \[mf invocation\]](#), page 24. In the file ‘newmode.mf’, you should have the following (with no `mode_def` or `enddef`), if you are using ‘modes.mf’ conventions:

```
mode_param (pixels_per_inch, dpi);
mode_param (blacker, b);
mode_param (fillin, f);
mode_param (o_correction, o);
mode_common_setup_;
```

(Of course, you should use real numbers for *dpi*, *b*, *f*, and *o*.)

For more information on the use of `smode`, or if you are not using ‘modes.mf’, see page 269 of *The Metafontbook*.

5.4 Online Metafont graphics

The Web2c implementation of Metafont can do online graphics with a number of devices. (See the Metafont manual for more information about how to draw on your screen.) By default, no graphics support is enabled.

Metafont examines the `MFTERM` environment variable or config file value at runtime, or the `TERM` environment variable if `MFTERM` is not set, to determine the device support to use. Naturally, only the devices for which support has been compiled in can be selected.

Here is a table of the possibilities, showing the `MFTERM` value and the corresponding `configure` option(s) in parentheses.

<code>epsf</code>	(<code>--enable-epsfwin</code>) Pseudo-window server for Encapsulated PostScript (see <code>'web2c/window/epsf.c'</code>). This device produces an EPS file containing the graphics which would be displayed online on other devices. The name of the EPS file defaults to <code>metafont.eps</code> but can be changed by setting the <code>MFEPSPS</code> environment variable to the new filename. Contributed by Mathias Herberts.
<code>hp2627</code>	(<code>--enable-hp2627win</code>) HP2627a color graphics terminals.
<code>mftalk</code>	(<code>--enable-mftalkwin</code>) Generic window server (see <code>'web2c/window/mftalk.c'</code>). ■
<code>next</code>	(<code>--enable-next</code>) NeXT window system. This requires a separate program, called <code>DrawingServant</code> , available separately. See the <code>'web2c/window/next.c'</code> .
<code>regis</code>	(<code>--enable-regiswin</code>) Regis terminals.
<code>sun</code>	(<code>--enable-suntoolswin</code>) The old Suntools (not any flavor of X) window system. (You can get the even older SunWindows <code>gfx</code> system by using <code>'sun-gfx.c'</code> .)
<code>tek</code>	(<code>--enable-tektronixwin</code>) Tektronix terminals.
<code>uniterm</code>	(<code>--enable-unitermwin</code>) Uniterm, Simon Poole's emulator of a smart Tektronix 4014 terminal. This may work with regular Tektronix terminals as well; it's faster than the driver <code>'--enable-tektronixwin'</code> selects.
<code>xterm</code>	<code>'--with-x'</code> The X window system (version 11).

There are two variants of the X11 support, one that works with the Xt toolkit, and another that works directly with Xlib. The Xt support is more efficient and has more functionality, so it is the default. If you must use the Xlib support, use `'configure --with-x --with-kf-x-toolkit=no'`.

Specify `'--disable-mf-nowin'` in order not to build a separate non-windows-capable Metafont executable `mf-nowin` (or `mf-nowin.exe`).

You cannot specify any of the usual X options (e.g., `'-geometry'`) on the Metafont command line, but you can specify X resources in your `'~/.Xdefaults'` or `'~/.Xresources'` file. The class name is `Metafont`. If you're using the Xt support, all the usual X toolkit resources are supported. If you're using the Xlib support, only the `geometry` resource is supported.

You specify the X display to which Metafont connects in the `DISPLAY` environment variable, as usual.

Writing support for a new device is straightforward. Aside from defining the basic drawing routines that Metafont uses (see ‘`mf.web`’), you only have to add another entry to the tables on the last page of ‘`web2c/lib/texmfmp.c`’. Or you can write an independent program and use MFTalk (see ‘`web2c/window/mftalk.c`’).

5.5 GFtoDVI: Character proofs of fonts

GFtoDVI makes *proof sheets* from a GF bitmap file as output by, for example, Metafont (see [Chapter 5 \[Metafont\]](#), [page 24](#)). This is an indispensable aid for font designers or Metafont hackers. Synopsis:

```
gftodvi [option]... gfname[gf]
```

The font *gfname* is searched for in the usual places (see [Section “Glyph lookup” in Kpathsea](#)). To see all the relevant paths, set the environment variable `KPATHSEA_DEBUG` to ‘-1’ before running the program.

The suffix ‘*gf*’ is supplied if not already present. This suffix is not an extension, no ‘.’ precedes it; for instance, ‘`cmr10.600gf`’.

The output filename is the basename of *gfname* extended with ‘`.dvi`’, e.g., ‘`gftodvi /wherever/foo.600gf`’ creates ‘`./foo.dvi`’.

The characters from *gfname* appear one per page in the DVI output, with labels, titles, and annotations, as specified in Appendix H (Hardcopy Proofs) of *The Metafontbook*.

GFtoDVI uses several fonts besides *gfname* itself:

- *gray font* (default ‘`gray`’): for the pixels that actually make up the character. Simply using black is not right, since then labels, key points, and other information could not be shown.
- *title font* (default ‘`cmr8`’): for the header information at the top of each output page.
- *label font* (default ‘`cmtt10`’): for the labels on key points of the figure.
- *slant font* (no default): for diagonal lines, which are otherwise simulated using horizontal and vertical rules.

To change the default fonts, you must use `special` commands in your Metafont source file, typically via commands like `slantfont slantlj4`. There is no default slant font since no one printer is suitable as a default. You can make your own by copying one of the existing files, such as ‘`.../fonts/source/public/misc/slantlj4.mf`’ and then running `mf` on it.

For testing purposes, you may find it useful to run `mf rtest` (hit RETURN when it stops) to get a ‘*gf*’ file of a thorn glyph. Or use `mfw` instead of `mf` to have the glyph(s) displayed on the screen. After that, `gftodvi rtest.2602gf` should produce ‘`rtest.dvi`’, which you process as usual.

The program accepts the following option, as well as the standard ‘`-verbose`’, ‘`-help`’, and ‘`-version`’ (see [Section 3.2 \[Common options\]](#), [page 7](#)):

```
‘-overflow-label-offset=points’
```

Typeset the so-called overflow labels, if any, *points* T_EX points from the right edge of the character bounding box. The default is a little over two inches (ten million scaled points, to be precise). Overflow equations are used to locate coordinates when their actual position is too crowded with other information.

5.6 MFT: Prettyprinting Metafont source

MFT translates a Metafont program into a T_EX document suitable for typesetting, with the aid of T_EX macros defined in the file ‘`mftmac.tex`’. Synopsis:

```
mft [option]... mfname[.mf]
```

MFT searches the usual places for *mfname* (see [Section “Supported file formats” in Kpathsea](#)). To see all the relevant paths, set the environment variable `KPATHSEA_DEBUG` to ‘-1’ before running the program. The output goes to the basename of *mfname* extended with ‘.tex’, e.g., ‘`mft /wherever/foo.mf`’ creates ‘`./foo.tex`’.

Line breaks in the input are carried over into the output; moreover, blank spaces at the beginning of a line are converted to quads of indentation in the output. Thus, you have full control over the indentation and line breaks. Each line of input is translated independently of the others.

Further control is allowed via Metafont comments:

- Metafont comments following a single ‘%’ should be valid T_EX input. But Metafont material can be included within vertical bars in a comment; this will be translated by MFT as if it were regular Metafont code. For example, a comment like ‘% |x2r| is the tip of the bowl’ will be translated into the T_EX ‘% `x_{2r}` is the ...’, i.e., the ‘x2r’ is treated as an identifier.
- ‘%%’ indicates that the remainder of an input line should be copied verbatim to the output. This is typically used to introduce additional T_EX material at the beginning or an MFT job, e.g. code to modify the standard layout or the formatting macros defined in ‘`mftmac.tex`’, or to add a line saying ‘%%\bye’ at the end of the job. (MFT doesn’t add this automatically in order to allow processing several files produces by MFT in the same T_EX job.)
- ‘%% token1 other-tokens’ introduces a change in MFT’s formatting rules; all the *other-tokens* will henceforth be translated according to the current conventions for *token1*. The tokens must be symbolic (i.e., not numeric or string tokens). For example, the input line

```
%% addto fill draw filldraw
```

says to format the ‘fill’, ‘draw’, and ‘filldraw’ operations of plain Metafont just like the primitive token ‘addto’, i.e., in boldface type. Without such reformatting commands, MFT would treat ‘fill’ like an ordinary tag or variable name. In fact, you need a ‘%%’ command even to get parentheses to act like delimiters.

- ‘%%’ introduces an MFT comment, i.e., MFT ignores the remainder of such a line.
- Five or more ‘%’ signs should not be used.

(The above description was edited from ‘`mft.web`’, written by D.E. Knuth.)

The program accepts the following options, as well as the standard ‘-help’ and ‘-version’ (see [Section 3.2 \[Common options\]](#), [page 7](#)):

```
‘-change=chfile[.ch]’
```

Apply the change file *chfile* as with Tangle and Weave (see [Chapter 8 \[WEB\]](#), [page 37](#)).

`'-style=mftfile[.mft]'`

Read *mftfile* before anything else; a MFT style file typically contains only MFT directives as described above. The default style file is named `'plain.mft'`, which defines this properly for programs using plain Metafont. The MFT files is searched along the `MFTINPUTS` path; see [Section “Supported file formats” in *Kpathsea*](#).

Other examples of MFT style files are `'cmbase.mft'`, which defines formatting rules for the macros defined in `'cm.base'`, and `'e.mft'`, which was used in the production of Knuth’s Volume E, *Computer Modern Typefaces*.

Using an appropriate MFT style file, it is also possible to configure MFT for typesetting MetaPost sources. However, MFT does not search the usual places for MetaPost input files.

If you use eight-bit characters in the input file, they are passed on verbatim to the \TeX output file; it is up to you to configure \TeX to print these properly.

6 MetaPost: Creating technical illustrations

MetaPost is a picture-drawing language similar to Metafont (see [Chapter 5 \[Metafont\]](#), [page 24](#)), but instead of outputting bitmaps in a “font”, it outputs PostScript commands. It’s primarily intended for creating technical illustrations.

MetaPost also provides for arbitrary integration of text and graphics in a natural way, using any typesetter (T_EX and Troff are both supported) and a number of other subsidiary programs, described below.

6.1 mpost invocation

MetaPost (installed as `mpost`) reads a series of pictures specified in the MetaPost programming language, and outputs corresponding PostScript code. This section merely describes the options available in the Web2c implementation. For a complete description of the MetaPost language, see AT&T technical report CSTR-162, generally available in ‘*texmf/doc/metapost/*’, where *texmf* is the root of T_EX directory structure. See also:

- <http://cm.bell-labs.com/who/hobby/MetaPost.html> (the MetaPost author’s home page);
- <http://tug.org/metapost> (papers, packages, and related information).

Also, a standard MetaPost package for drawing graphs is documented in AT&T technical report CSTR-164, available as the file ‘*mpgraph.ps*’, generally stored alongside ‘*mpman.ps*’.

MetaPost processes its command line and determines its memory dump (`mem`) file in a way exactly analogous to Metafont and T_EX (see [Section 4.1 \[tex invocation\]](#), [page 14](#), and see [Section 3.5.2 \[Memory dumps\]](#), [page 10](#)). Synopses:

```
mpost [option]... [mpname[.mp]] [mp-commands]
mpost [option]... \first-line
mpost [option]... &mem args
```

MetaPost searches the usual places for the main input file *mpname* (see [Section “Supported file formats” in Kpathsea](#)), extending *mpname* with ‘*.mp*’ if necessary. To see all the relevant paths, set the environment variable `KPATHSEA_DEBUG` to ‘*-1*’ before running the program.

MetaPost writes its PostScript output to a series of files ‘*basempname.nnn*’ (or perhaps ‘*basempname.ps*’, very occasionally ‘*basempname.tfm*’), where *nnn* are the figure numbers specified in the input, typically to the `beginfig` macro, and *basempname* is the basename of *mpname*, or ‘*mpout*’ if no input file was specified. MetaPost uses the ‘*.ps*’ extension when the figure number is out of range, e.g., if you say `beginfig(-1)`.

You can use the output files as figures in a T_EX document just as with any other PostScript figures. For example, with this T_EX command:

```
\special{psfile="filename"}
```

or by using ‘*epsf.tex*’ (see [Section “EPSF macros” in Dvips](#)).

The MetaPost construct

```
btex ... tex-input ... etex
```

generates a MetaPost picture expression corresponding to *tex-input*.

The construct

```
verbatimtex ... tex-input ... etex
```

simply passes the *tex-input* through to T_EX. For example, if you are using L^AT_EX, your MetaPost input file must start with a `verbatimtex` block that gives the necessary `\documentclass` (or `\documentstyle`) `\begin{document}` command. You will also need to set the environment variable `TEX` to `'latex'`.

tex-input need not be specifically T_EX input; it could also be Troff. In that case, you will need the `'-m pictures'` Troff macro package (unfortunately absent from many Troff implementations), or an equivalent such as the `'-m pspic'` macros from GNU groff described in [grops\(1\)](#).

Naturally, you must use fonts that are supported by the typesetter; specifically, you'll probably want to use standard PostScript fonts with Troff. And only the T_EX system understands Computer Modern or other Metafont fonts; you can also use PostScript fonts with T_EX, of course.

MetaPost-generated PostScript figures which do use Computer Modern fonts for labels cannot be directly previewed or printed. Instead, you must include them in a T_EX document and run the resulting DVI file through Dvips to arrange for the downloading of the required fonts (see [Section “Fonts in figures” in Dvips](#)). To help with this, the MetaPost distribution provides a small T_EX file `'mproof.tex'` which is typically called as:

```
tex mproof mp-output-files... ; dvips mproof -o
```

The resulting file `'mproof.ps'` can then be printed or previewed.

To generate EPSF files, set the internal MetaPost variable `prologues` positive. To make the output files self-contained, use only standard PostScript fonts. MetaPost reads the same `'psfonts.map'` file as Dvips, to determine PostScript fonts that need to be downloaded (see [Section “psfonts.map” in Dvips](#)).

It is possible for pdfT_EX to read MetaPost output directly; this is in contrast to general EPSF files, which have to be converted for use with PDF output. The easiest way is to name the MetaPost output files with the `.mps` extension. Then the L^AT_EX `\includegraphics` command, for example, will be able to read them, even when outputting PDF.

MetaPost can write output files, via the `write` primitive; this opens a security hole. See [Section 4.1 \[tex invocation\]](#), page 14.

The program accepts the following options, as well as the standard `'-help'` and `'-version'` (see [Section 3.2 \[Common options\]](#), page 7):

```

'-[no]-file-line-error'
'-fmt=fmtname'
'-halt-on-error'
'-ini'
'-interaction=string'
'-jobname=string'
'-kpathsea-debug=number'
'-[no]parse-first-line'
'-output-directory'
'-progname=string'
'-recorder'
'-translate-file=tcxfile'
'-8bit'      These options are common to TEX, Metafont, and MetaPost. See Section 3.2
              [Common options], page 7.

'-T'
'-troff'     Set the prologues internal variable to 1.

'-tex=texprogram'
              When this option is given, the program texprogram is used to typeset the labels.

```

6.2 Initial MetaPost

inimpost is the “initial” form of MetaPost, which does lengthy initializations avoided by the “virgin” (*vir*) form, so as to be capable of dumping ‘.mem’ files (see Section 3.5.2 [Memory dumps], page 10). For a detailed comparison of virgin and initial forms, see Section 3.5.1 [Initial and virgin], page 10. In past releases, a separate program *inimpost* was installed for the initial form, but this is no longer the case.

For a list of options and other information, see Section 6.1 [*impost* invocation], page 31.

The principal memory dump file used with MetaPost is the default, ‘plain.mem’, also known as ‘mpost.mem’ (again, see Section 3.5.2 [Memory dumps], page 10). It is created by default during installation, but you can also do so by hand if necessary (e.g., if a ‘plain.mp’ update is issued):

```
mpost -ini '\input plain dump'
```

(The quotes prevent interpretation of the backslashes from the shell.) Then install the resulting ‘plain.mem’ in ‘\$(memdir)’ (‘/usr/local/share/texmf/web2c’ by default), and link ‘mpost.mem’ to it.

MetaPost also provides a mem file with all the features of plain Metafont, called ‘mfplain.mem’. You can create that in the same way; just replace ‘plain’ in the above command with ‘mfplain’. ‘mfplain.mem’ file lets you directly process Metafont source files with MetaPost, producing character proofs (one file for each character) similar to those produced with Metafont in proof mode and G_FtoD_VI (see Section 5.5 [*gftodvi* invocation], page 28).

Another MetaPost mem file, named MetaFun, provides many advanced features. It was developed and maintained by Pragma ADE. The home page for MetaFun, other MetaPost-related things, and lots more from Pragma: <http://www.pragma-ade.com/>.

6.3 DVItMP: DVI to MPX conversion

DVItMP converts DVI files into low-level MetaPost commands in a so-called MPX file. Synopsis:

```
dvitomp dvifile[.dvi] [mpxfile[.mpx]]
```

If *mpxfile* is not specified, the output goes to the basename of *dvifile* extended with ‘.mpx’, e.g., ‘`dvitomp /wherever/foo.dvi`’ creates ‘`./foo.mpx`’.

DVItMP supports Dvips-style color specials, such as ‘`color push name`’ and ‘`color pop`’, outputting them as `withcolor` MetaPost commands.

The only options are ‘`-help`’ and ‘`-version`’ (see [Section 3.2 \[Common options\]](#), page 7).

7 BibTeX: Bibliographies

BibTeX automates much of the job of typesetting bibliographies, and makes bibliography entries reusable in many different contexts.

7.1 BibTeX invocation

BibTeX creates a printable bibliography (‘.bbl’) file from references in a ‘.aux’ file, generally written by TeX or LaTeX. The ‘.bbl’ file is then incorporated on a subsequent run. The basic bibliographic information comes from ‘.bib’ files, and a BibTeX style (‘.bst’) file controls the precise contents of the ‘.bbl’ file. Synopsis:

```
bibtex [option]... auxfile[.aux]
```

The output goes to the basename of *auxfile* extended with ‘.bbl’; for example, ‘bibtex /wherever/foo.aux’ creates ‘./foo.bbl’. BibTeX also writes a log file to the basename of *auxfile* extended with ‘.blg’.

The names of the ‘.bib’ and ‘.bst’ files are specified in the ‘.aux’ file as well, via the ‘\bibliography’ and ‘\bibliographystyle’ (La)TeX macros. BibTeX searches for ‘.bib’ files using the BIBINPUTS and TEXBIB paths, and for ‘.bst’ files using BSTINPUTS (see [Section “Supported file formats” in Kpathsea](#)). It does no path searching for ‘.aux’ files.

The program accepts the following options, as well as the standard ‘-help’ and ‘-version’ (see [Section 3.2 \[Common options\], page 7](#)):

‘-terse’ Suppress the program banner and progress reports normally output.

‘-min-crossrefs=*n*’

If at least *n* (2 by default) bibliography entries refer to another entry *e* via their `crossref` field, include *e* in the .bbl file, even if it was not explicitly referenced in the .aux file. For example, *e* might be a conference proceedings as a whole, with the cross-referencing entries being individual articles published in the proceedings. In some circumstances, you may want to avoid these automatic inclusions altogether; to do this, make *n* a sufficiently large number.

See also:

‘btxdoc.tex’

Basic LaTeXable documentation for general BibTeX users.

‘btXHak.tex’

LaTeXable documentation for style designers.

‘btxdoc.bib’

BibTeX database file for the two above documents.

‘xampl.bib’

Example database file with all the standard entry types.

‘<ftp://ftp.math.utah.edu/pub/tex/bib/>’

A very large ‘.bib’ and ‘.bst’ collection, including references for all the standard TeX books and a complete bibliography for TUGboat.

7.2 Basic BibTeX style files

Here are descriptions of the four standard and four semi-standard basic BibTeX styles. ‘CTAN:/biblio/bibtex’ contains these and many more (for CTAN info, see [Section “unix-tex.ftp”](#) in *Kpathsea*).

plain	Sorts entries alphabetically, with numeric labels. Generally formatted according to van Leunen’s <i>A Handbook for Scholars</i> . The other style files listed here are based on plain .
abbrv	First names, month names, and journal names are abbreviated.
acm	Names are printed in small caps.
alpha	Alphanumeric labels, e.g., ‘Knu66’.
apalike	No labels at all; instead, the year appears in parentheses after the author. Use this in conjunction with ‘ apalike.tex ’ (plain TeX) or ‘ apalike.sty ’ (LaTeX), which also changes the citations in the text to be ‘(<i>author</i> , <i>year</i>)’.
ieeetr	Numeric labels, entries in citation order, IEEE abbreviations, article titles in quotes.
siam	Numeric labels, alphabetic order, <i>Math. Reviews</i> abbreviations, names in small caps.
unsrt	Lists entries in citation order, i.e., unsorted.
btxbst.doc	The template file and documentation for the standard styles.

8 WEB: Literate programming

WEB languages allow you to write a single source file that can produce both a compilable program and a well-formatted document describing the program in as much detail as you wish to prepare. Writing in this kind of dual-purpose language is called *literate programming*. (The Usenet newsgroup ‘comp.programming.literate’ is devoted to this subject.)

WEB-like languages have been implemented with many pairs of base languages: Cweb provides C and Troff (see [Appendix B \[References\]](#), page 54); CWEB provides C and T_EX (‘CTAN:/web/c_cpp/cweb’); Spiderweb provides C, C++, Awk, Ada, many others, and T_EX (‘CTAN:/web/spiderweb’); and, of course, the original WEB provides Pascal and T_EX, the implementation languages for the original T_EX, Metafont, MetaPost, and related programs to come from the T_EX project at Stanford.

The original WEB language is documented in the file ‘webman.tex’, which is included in the <ftp://ftp.tug.org/tex/lib.tar.gz> archive (and available in many other places, of course).

8.1 Tangle: Translate WEB to Pascal

Tangle creates a compilable Pascal program from a WEB source file (see [Chapter 8 \[WEB\]](#), page 37). Synopsis:

```
tangle [option]... webfile[.web] [changefile[.ch]]
```

The Pascal output is written to the basename of *webfile* extended with ‘.p’; for example, ‘tangle /wherever/foo.web’ creates ‘./foo.p’. Tangle applies *changefile* to *webfile* before writing the output; by default, there is no change file.

If the program makes use of the WEB string facility, Tangle writes the string pool to the basename of *webfile* extended with ‘.pool’.

The Pascal output is packed into lines of 72 characters or less, with the only concession to readability being the termination of lines at semicolons when this can be done conveniently.

The program accepts the following options, as well as the standard ‘--help’ and ‘--version’ (see [Section 3.2 \[Common options\]](#), page 7):

‘-length=*number*’

The number of characters that are considered significant in an identifier. Whether underline characters are counted depends on the ‘-underline’ option. The default value is 32, the original tangle used 7, but this proved too restrictive for use by Web2c.

‘-lowercase’

‘-mixedcase’

‘-uppercase’

These options specify the case of identifiers in the output of tangle. If ‘-uppercase’ (‘-lowercase’) is specified, tangle will convert all identifiers to uppercase (lowercase). The default is ‘-mixedcase’, which specifies that the case will not be changed.

`‘-underline’`

When this option is given, tangle does not strip underline characters from identifiers.

`‘-loose’`

`‘-strict’` These options specify how strict tangle must be when checking identifiers for equality. The default is `‘-loose’`, which means that tangle will follow the rules set by the case-smashing and underline options above. If `‘-strict’` is set, then identifiers will always be stripped of underlines and converted to uppercase before checking whether they collide.

8.2 Weave: Translate WEB to T_EX

Weave creates a T_EX document from a WEB source file (see [Chapter 8 \[WEB\], page 37](#)), assuming various macros defined in `‘webmac.tex’`. It takes care of typographic details such as page layout, indentation, and italicizing identifiers. It also automatically gathers and outputs extensive cross-reference information. Synopsis:

```
weave [option]... webfile[.web] [changefile[.ch]]
```

The output is to the basename of *webfile* extended with `‘.tex’`; for example, `‘weave /wherever/foo.web’` creates `‘./foo.tex’`. Weave applies *changefile* to *webfile* before writing the output; by default, there is no change file.

The program accepts the following option, as well as the standard `‘-verbose’`, `‘-help’` and `‘-version’` (see [Section 3.2 \[Common options\], page 7](#)):

`‘-x’` Omit the cross-reference information: the index, the list of WEB module names, and the table of contents (an empty `‘CONTENTS.tex’` file will still be written when the Weave output file is processed by T_EX using the default `‘webmac.tex’`, though).

Conventionally, WEB programmers should define the T_EX `\title` macro at the beginning of the source file. Also, to get output of only changed modules, one can say `\let\maybe=\iffalse` (usually as the first change in the change file).

8.3 Pooltype: Display WEB pool files

Pooltype shows the so-called *string number* of each string in a WEB pool file (see [Chapter 8 \[WEB\], page 37](#)), as output by Tangle (see [Section 8.1 \[tangle invocation\], page 37](#)), including the first 256 strings corresponding to the possible input characters. Pooltype primarily serves as an example of WEB conventions to implementors of the T_EX system. Synopsis:

```
pooltype [option]... poolfile[.pool]
```

No path searching is done for *poolfile*. Output is to standard output.

The only options are `‘--help’` and `‘--version’` (see [Section 3.2 \[Common options\], page 7](#)).

As an example of the output, here is the (edited) output for `‘tex.pool’`:

```
0: "^^@"
1: "^^A"
...
255: "^^ff"
```

```
256: "pool size"  
...  
1314: "Using character substitution: "  
(23617 characters in all.)
```

In Metafont and MetaPost, the first 256 characters are actually represented as single bytes (i.e., themselves), not in the ‘^^’ notation. Consider Pooltype as showing the results after conversion for output.

9 DVI utilities

T_EX outputs a file in *DVI* (DeVice Independent) format as a compact representation of the original document. DVI files can be translated to meet the requirements of a real physical device, such as PostScript printers (see [Section “Introduction” in Dvips](#)), PCL printers (see `dvilj(1)`), and X displays (see `xdvi(1)`). In fact, DVI translators are available for virtually all common devices: see ‘*CTAN:/dviware*’ (for CTAN info, see [Section “unixtex.ftp” in Kpathsea](#)).

For the precise definition of the DVI file format, see (for example) the source file ‘*web2c/dvitype.web*’.

The DVI-processing programs in the Web2c distribution are not device drivers; they perform generic utility functions.

9.1 DVICopy: Canonicalize virtual font references

DVICopy reads a DVI file, expands any references to virtual fonts (see [Section “Virtual fonts” in Dvips](#)) to base fonts, and writes the resulting DVI file. Thus you can use virtual fonts even if your DVI processor does not support them, by passing the documents through DVICopy first. Synopsis:

```
dvicopy [option]... [indvi[.dvi] [outdvi[.dvi]]]
```

DVICopy reads standard input if *indvi* is not specified, and writes standard output if *outdvi* is not specified.

The program accepts the following options, as well as the standard ‘*-help*’ and ‘*-version*’ (see [Section 3.2 \[Common options\], page 7](#)):

‘*-magnification=integer*’

Override existing magnification in *indvi* with *integer*; 1000 specifies no magnification. This is equivalent to setting T_EX’s `\mag` parameter.

‘*-max-pages=n*’

Process *n* pages; default is one million.

‘*-page-start=page-spec*’

Start at the first page matching *page-spec*, which is one or more (signed) integers separated by periods, corresponding to T_EX’s `\count0...9` parameters at `\shipout` time; ‘***’ matches anything. Examples: ‘3’, ‘1.*.-4’.

9.2 DVItypE: Plain text transliteration of DVI files

DVItypE translates a DeVice Independent (DVI) file (as output by T_EX, for example) to a plain text file that humans can read. It also serves as a DVI-validating program, i.e., if DVItypE can read a file, it’s correct. Synopsis:

```
dvitype [option]... dvifile[.dvi]
```

DVItypE does not read any bitmap files, but it does read TFM files for fonts referenced in *dvifile*. The usual places are searched (see [Section “Supported file formats” in Kpathsea](#)). To see all the relevant paths, set the environment variable `KPATHSEA_DEBUG` to ‘*-1*’ before running the program.

Output goes to standard output.

The program accepts the following options, as well as the standard ‘`-help`’ and ‘`-version`’ (see [Section 3.2 \[Common options\]](#), page 7):

‘`-dpi=real`’

Do pixel movement calculations at *real* pixels per inch; default 300.0.

‘`-magnification=integer`’

Override existing magnification in *indvi* with *integer*; 1000 specifies no magnification. This is equivalent to setting T_EX’s `\mag` parameter.

‘`-max-pages=n`’

Process *n* pages; default is one million.

‘`-output-level=n`’

Verbosity level of output, from 0 to 4 (default 4):

- 0: Global document information only.
- 1: Most DVI commands included, and typeset characters summarized.
- 2: Character and movement commands explicitly included.
- 3: DVI stack and current position calculations included.
- 4: Same information as level 3, but DVIt_ype does random positioning in the file, reading the DVI postamble first.

‘`-page-start=page-spec`’

Start at the first page matching *page-spec*, which is one or more (signed) integers separated by periods, corresponding to T_EX’s `\count0...9` parameters at `\shipout` time; ‘`*`’ matches anything. Examples: ‘1’, ‘5.*.-9’.

‘`-show-opcodes`’

Show numeric opcode values (in decimal) for DVI commands, in braces after the command name. This can help in debugging DVI utilities. We use decimal because in the DVI format documentation (in ‘*dvitype.web*’, among others) the opcodes are shown in decimal.

9.2.1 DVIt_ype output example

As an example of the output from DVIt_ype (see section above), here is its (abridged) translation of the ‘*story.dvi*’ resulting from running the example in *The T_EXbook*, with ‘`-output-level=4`’ and ‘`-show-opcodes`’ on.

```
...
Options selected:
  Starting page = *
  Maximum number of pages = 1000000
  Output level = 4 (the works)
  Resolution = 300.00000000 pixels per inch
numerator/denominator=25400000/473628672
magnification=1000;      0.00006334 pixels per DVI unit
' TeX output 1992.05.17:0844'
Postamble starts at byte 564.
maxv=43725786, maxh=30785863, maxstackdepth=3, totalpages=1
Font 33: cmsl10---loaded at size 655360 DVI units
```

```

Font 23: cmbx10---loaded at size 655360 DVI units
Font 0: cmr10---loaded at size 655360 DVI units

42: beginning of page 1
87: push {141}
level 0:(h=0,v=0,w=0,x=0,y=0,z=0,hh=0,vv=0)
88: down3 -917504 {159} v:=0-917504=-917504, vv:=-58
92: pop {142}
...
104: putrule {137} height 26214, width 30785863 (2x1950 pixels)
113: down3 5185936 {159} v:=655360+5185936=5841296, vv:=370
117: push {141}
level 1:(h=0,v=5841296,w=0,x=0,y=0,z=0,hh=0,vv=370)
118: right4 12265425 {146} h:=0+12265425=12265425, hh:=777
[ ]
123: fntdef1 23 {243}: cmbx10
145: fntnum23 {194} current font is cmbx10
146: setchar65 h:=12265425+569796=12835221, hh:=813
147: w3 251220 {150} h:=12835221+251220=13086441, hh:=829
151: setchar83 h:=13086441+418700=13505141, hh:=856
...
164: setchar82 h:=17448202+565245=18013447, hh:=1142
165: x0 -62805 {152} h:=18013447-62805=17950642, hh:=1138
166: setchar89 h:=17950642+569796=18520438, hh:=1174
[A SHORT STORY]
167: pop {142}
level 1:(h=0,v=5841296,w=0,x=0,y=0,z=0,hh=0,vv=370)
...
550: pop {142}
level 0:(h=0,v=42152922,w=0,x=0,y=0,z=0,hh=0,vv=2670)
551: down3 1572864 {159} v:=42152922+1572864=43725786, vv:=2770
555: push {141}
level 0:(h=0,v=43725786,w=0,x=0,y=0,z=0,hh=0,vv=2770)
556: right4 15229091 {146} h:=0+15229091=15229091, hh:=965
561: setchar49 h:=15229091+327681=15556772, hh:=986
[ 1]
562: pop {142}
level 0:(h=0,v=43725786,w=0,x=0,y=0,z=0,hh=0,vv=2770)
563: eop {140}

```

Explanation:

- The DVItypes options are recorded at the beginning, followed by global information about the document, including fonts used.
- Each DVI command is preceded by its byte position in the file ('42:', '87:', ...), and (because of the '-show-opcodes') followed by its decimal opcode value in braces ('{141}', '{142}', ...).
- The 'level' lines record information about the DVI stack; 'h' and 'v' define the current

position in DVI units, while ‘**hh**’ and ‘**vv**’ are the same in pixels.

- Text sequences are summarized in brackets, as in ‘**[A SHORT STORY]**’ and the ‘**[1]**’.

10 Font utilities

The Web2c programs described here convert between various T_EX-related font formats; the first section below briefly describes the formats. GFtoPK is the only one that is routinely used, as Metafont outputs GF format, but it's most efficient for device drivers to use PK.

The precise definitions of the PK, GF, TFM, PL, VF, and VPL formats mentioned below are in the source files that read them; `'pktype.web'`, `'gftype.web'`, `'tftopl.web'`, etc.

10.1 Font file formats

(For another perspective on this, see [Section “Font concepts” in Dvips](#)).

Font files come in several varieties, with suffixes like:

`.tfm` `.pk` `.gf` `.pxl` (obsolete) `.pl` `.mf` `.vf` `.vpl`

Each represents a file format.

A TFM (T_EX font metric) file is a compact binary file that contains information about each character in a font, about combinations of characters within that font, and about the font as a whole. The font metric information contained in TFM files is device-independent units is used by T_EX to do typesetting. Unlike the bitmap (raster) fonts described below, TFM font files contain no information about the shapes of characters. They describe rectangular areas and combinations thereof, but not what will eventually be printed in those areas.

Since T_EX does scaling calculations, one TFM file serves for all magnifications of a given typeface. On the other hand, the best printed results are obtained when magnified (or reduced fonts) are not produced geometrically (as done by PostScript, for example) but rather optically, with each size a separate design (as done with Computer Modern and the EC fonts, for example); then a separate TFM file is needed for each size.

At any rate, T_EX produces a DVI (DeVice Independent) file from your source document. In order to print DVI files on real devices, you need font files defining digitized character shapes and other data. Then previewers and printer-driver programs can translate your DVI files into something usable by your monitor or printer. Bitmap fonts come with suffixes such as `'600pk'` or `'600gf'` or `'3000pxl'`, where the `'600'` is the horizontal dots-per-inch resolution at which the font was produced, and the `'pk'` or `'gf'` or `'pxl'` indicates the font format. Outline fonts in PostScript Type 1 format have suffixes such as `'pfa'` or `'pfb'`.

Fonts in pk (packed) format are in the tightly packed raster format that is pretty much the standard today. They take up less space than fonts in the gf (generic font) format that Metafont generates, and far less space than fonts in pxl format. Fonts in pxl format take up gross amounts of disk space and permit only 128 characters. They are obsolete.

Font files with the `'pl'` (property list) suffix are the plain text (human-readable) analog of the binary `'tfm'` files. The TFtoPL and PLtoTF programs convert between the two formats (see [Section 10.6 \[tftopl invocation\]](#), page 49 and [Section 10.7 \[pltotf invocation\]](#), page 51).

Font files with the `'mf'` suffix are in Metafont source format. These are the files used by Metafont to generate rastered fonts for specific typefaces at specific magnifications for the specific resolution and type of mapping used by your device.

The suffix ‘.vf’ identifies “virtual font” files, for which ‘.vpl’ is the human-readable analog. See [Section 10.8 \[vftovp invocation\], page 51](#), and [Section 10.9 \[vptovf invocation\], page 52](#). For further discussion of virtual fonts, see ‘[CTAN:/doc/virtual-fonts.knuth](#)’, ‘[CTAN:/help/virtualfonts.txt](#)’, and [Section “Virtual fonts” in Dvips](#).

(This section is based on documentation in the original Unix T_EX distribution by Pierre MacKay and Elizabeth Tachikawa.)

10.2 GFtoPK: Generic to packed font conversion

GFtoPK converts a generic font (GF) file output by, for example, Metafont (see [Section 5.1 \[mf invocation\], page 24](#)) to a packed font (PK) file. PK files are considerably smaller than the corresponding gf files, so they are generally the bitmap font format of choice. Some DVI-processing programs, notably Dvips, only support PK files and not GF files. Synopsis:

```
gftopk [option]... gfname.dpi[gf] [pkfile]
```

The font *gfname* is searched for in the usual places (see [Section “Glyph lookup” in Kpathsea](#)). To see all the relevant paths, set the environment variable KPATHSEA_DEBUG to ‘-1’ before running the program.

The suffix ‘gf’ is supplied if not already present. This suffix is not an extension; no ‘.’ precedes it: for instance, ‘cmr10.600gf’.

If *pkfile* is not specified, the output is written to the basename of ‘*gfname.dpipk*’, e.g., ‘gftopk /wherever/cmr10.600gf’ creates ‘./cmr10.600pk’.

The only options are ‘--verbose’, ‘--help’, and ‘--version’ (see [Section 3.2 \[Common options\], page 7](#)).

10.3 PKtoGF: Packed to generic font conversion

PKtoGF converts a packed font (PK) file to a generic font (GF) file. Since PK format is much more compact than GF format, the most likely reason to do this is to run GFtype (see [Section 10.5 \[gftype invocation\], page 47](#)) on the result, so you can see the bitmap images. Also, a few old utility programs do not support PK format. Synopsis:

```
pktoGF [option]... pkname.dpi[pk] [gffile]
```

The font *pkname* is searched for in the usual places (see [Section “Glyph lookup” in Kpathsea](#)). To see all the relevant paths, set the environment variable KPATHSEA_DEBUG to ‘-1’ before running the program.

The suffix ‘pk’ is supplied if not already present. This suffix is not an extension; no ‘.’ precedes it: for instance, ‘cmr10.600pk’.

If *gffile* is not specified, the output is written to the basename of ‘*pkname.dpigf*’, e.g., ‘pktoGF /wherever/cmr10.600pk’ creates ‘./cmr10.600gf’.

The only options are ‘--verbose’, ‘--help’, and ‘--version’ (see [Section 3.2 \[Common options\], page 7](#)).

10.4 PKtype: Plain text transliteration of packed fonts

PKtype translates a packed font (PK) bitmap file (as output by GFtoPK, for example) to a plain text file that humans can read. It also serves as a PK-validating program, i.e., if PKtype can read a file, it’s correct. Synopsis:

```
pktype pfname.dpi [pk]
```

The font *pfname* is searched for in the usual places (see [Section “Glyph lookup” in *Kpathsea*](#)). To see all the relevant paths, set the environment variable `KPATHSEA_DEBUG` to ‘-1’ before running the program.

The suffix ‘pk’ is supplied if not already present. This suffix is not an extension; no ‘.’ precedes it: for instance, ‘cmr10.600pk’.

The translation is written to standard output.

The only options are ‘-help’ and ‘-version’ (see [Section 3.2 \[Common options\], page 7](#)).

As an example of the output, here is the (abridged) translation of the letter ‘K’ in ‘cmr10’, as rendered at 600dpi with the mode ‘ljfour’ from `modes.mf` (available from `ftp://ftp.tug.org/tex/modes.mf`).

```
955:  Flag byte = 184  Character = 75  Packet length = 174
      Dynamic packing variable = 11
      TFM width = 815562  dx = 4259840
      Height = 57  Width = 57  X-offset = -3  Y-offset = 56
      [2]23(16)17(8)9(25)11(13)7(27)7(16)7(28)4(18)7(28)2(20)7(27)...
      ...
      (14)9(24)12(5)[2]23(13)21
```

Explanation:

‘955’ The byte position in the file where this character starts.

‘Flag byte’

‘Dynamic packing variable’

Related to the packing for this character; see the source code.

‘Character’

The character code, in decimal.

‘Packet length’

The total length of this character definition, in bytes.

‘TFM width’

The device-independent (TFM) width of this character. It is 2^{24} times the ratio of the true width to the font’s design size.

‘dx’

The device-dependent width, in *scaled pixels*, i.e., units of horizontal pixels times 2^{16} .

‘Height’

‘Width’ The bitmap height and width, in pixels.

‘X-offset’

‘Y-offset’

Horizontal and vertical offset from the upper left pixel to the reference (origin) pixel for this character, in pixels (right and down are positive). The *reference pixel* is the pixel that occupies the unit square in Metafont; the Metafont reference point is the lower left hand corner of this pixel. Put another way, the x-offset is the negative of the left side bearing; the right side bearing is the horizontal escapement minus the bitmap width plus the x-offset.

‘[2]23(16)...’

Finally, run lengths of black pixels alternate with parenthesized run lengths of white pixels, and brackets indicate a repeated row.

10.5 GFontype: Plain text transliteration of generic fonts

GFontype translates a generic font (GF) bitmap file (as output by Metafont, for example) to a plain text file that humans can read. It also serves as a GF-validating program, i.e., if GFontype can read a file, it’s correct. Synopsis:

```
gfontype [option]... gfname.dpi[gf]
```

The font *gfname* is searched for in the usual places (see [Section “Glyph lookup” in Kpathsea](#)). To see all the relevant paths, set the environment variable KPATHSEA_DEBUG to ‘-1’ before running the program.

The suffix ‘gf’ is supplied if not already present. This suffix is not an extension; no ‘.’ precedes it: for instance, ‘cmr10.600gf’.

The translation is written to standard output.

The program accepts the following options, as well as the standard ‘-help’ and ‘-version’ (see [Section 3.2 \[Common options\], page 7](#)):

‘-images’ Show the characters’ bitmaps using asterisks and spaces.

‘-mnemonics’

Translate all commands in the GF file.

As an example of the output, here is the (abridged) translation of the letter ‘K’ in ‘cmr10’, as rendered at 600dpi with the mode ‘ljfour’ from ‘modes.mf’ (available from <ftp://ftp.tug.org/tex/modes.mf>), with both ‘-mnemonics’ and ‘-images’ enabled.

GFontype outputs the information about a character in two places: a main definition and a one-line summary at the end. We show both. Here is the main definition:

```
2033: beginning of char 75: 3<=m<=60 0<=n<=56
      (initially n=56) paint(0)24(12)20
2043: newrow 0 (n=55) paint 24(12)20
2047: newrow 0 (n=54) paint 24(12)20
2051: newrow 0 (n=53) paint 24(12)20
2055: newrow 7 (n=52) paint 10(21)13
2059: newrow 8 (n=51) paint 8(23)9
...
2249: newrow 8 (n=5) paint 8(23)11
2253: newrow 7 (n=4) paint 10(22)12
2257: newrow 0 (n=3) paint 24(11)22
2261: newrow 0 (n=2) paint 24(11)22
2265: newrow 0 (n=1) paint 24(11)22
2269: newrow 0 (n=0) paint 24(11)22
2273: eoc
```

```
.<--This pixel's lower left corner is at (3,57) in METAFONT coordinates
```

*****	*****
*****	*****
*****	*****
*****	*****
*****	*****
*****	*****
...	
*****	*****
*****	*****
*****	*****
*****	*****
*****	*****
*****	*****

```
.<--This pixel's upper left corner is at (3,0) in METAFONT coordinates
```

Explanation:

‘2033’

‘2043’

‘...’ The byte position in the file where each GF command starts.

```
'beginning of char 75'
```

The character code, in decimal.

$$3 \leq m \leq 60 \quad 0 \leq n \leq 56$$

The character's bitmap lies between 3 and 60 (inclusive) horizontally, and between 0 and 56 (inclusive) vertically. (m is a column position and n is a row position.) Thus, 3 is the left side bearing. The right side bearing is the horizontal escapement (given below) minus the maximum m .

```
'(initially n=56) paint (0)24(12)20'
```

The first row of pixels: 0 white pixels, 24 black pixels, 12 white pixels, etc.

```
'newrow 0 (n=55) paint 24(12)20'
```

The second row of pixels, with zero leading white pixels on the row.

`'eoc'` The end of the main character definition.

Here is the GF postamble information that GFtype outputs at the end:

Character 75: dx 4259840 (65), width 815562 (64.57289), loc 2033

Explanation:

‘dx’ The device-dependent width, in *scaled pixels*, i.e., units of horizontal pixels times 2¹⁶. The ‘(65)’ is simply the same number rounded. If the vertical escapement is nonzero, it would appear here as a ‘dy’ value.

<code>'width'</code>	The device-independent (TFM) width of this character. It is 2^{24} times the ratio of the true width to the font's design size. The <code>'64.57289'</code> is the same number converted to pixels.
----------------------	---

'loc'	The byte position in the file where this character starts.
-------	--

10.6 TFMtoPL: T_EX font metric to property list conversion

TFMtoPL translates a T_EX font metric (TFM, see [Section “Metric files” in Dvips](#)) file (as output by Metafont, for example) to *property list format* (a list of parenthesized items describing the font) that humans can edit or read. This program is mostly used by people debugging T_EX implementations, writing font utilities, etc. Synopsis:

```
tftopl [option]... tfmname[.tfm] [plfile[.pl]]
```

The font *tfmname* (extended with ‘.tfm’ if necessary) is searched for in the usual places (see [Section “Supported file formats” in Kpathsea](#)). To see all the relevant paths, set the environment variable KPATHSEA_DEBUG to ‘-1’ before running the program.

If *plfile* (which is extended with ‘.pl’ if necessary) is not specified, the property list file is written to standard output. The property list file can be converted back to TFM format by the companion program TFMtoPL (see the next section).

The program accepts the following option, as well as the standard ‘-verbose’, ‘-help’ and ‘-version’ (see [Section 3.2 \[Common options\]](#), page 7):

‘-charcode-format=type’

Output character codes in the PL file according to *type*: either ‘octal’ or ‘ascii’. Default is ‘ascii’ for letters and digits, octal for all other characters. Exception: if the font’s coding scheme starts with ‘T_EX math sy’ or ‘T_EX math ex’, all character codes are output in octal.

In ‘ascii’ format, character codes that correspond to graphic characters, except for left and right parentheses, are output as a ‘C’ followed by the single character: ‘C K’, for example. In octal format, character codes are output as the letter ‘O’ followed by octal digits, as in ‘O 113’ for ‘K’.

‘octal’ format is useful for symbol and other non-alphabetic fonts, where using ASCII characters for the character codes is merely confusing.

As an example of the output, here is the (abridged) property list translation of ‘cmr10.tfm’:

```
(FAMILY CMR)
(FACE 0 352)
(CODINGScheme TEX TEXT)
(DESIGNSIZE R 10.0)
(COMMENT DESIGNSIZE IS IN POINTS)
(COMMENT OTHER SIZES ARE MULTIPLES OF DESIGNSIZE)
(CHECKSUM 0 11374260171)
(FONTDIMEN
  (SLANT R 0.0)
  (SPACE R 0.333334)
  (STRETCH R 0.166667)
  (SHRINK R 0.111112)
  (XHEIGHT R 0.430555)
  (QUAD R 1.000003)
  (EXTRASPACE R 0.111112)
)
(LIGTABLE
```

```

...
(LABEL C f)
(LIG C i 0 14)
(LIG C f 0 13)
(LIG C l 0 15)
(KRN 0 47 R 0.077779)
(KRN 0 77 R 0.077779)
(KRN 0 41 R 0.077779)
(KRN 0 51 R 0.077779)
(KRN 0 135 R 0.077779)
(STOP)
...
)
...
(Character C f
  (CHARWD R 0.305557)
  (CHARHT R 0.694445)
  (CHARIC R 0.077779)
  (COMMENT
    (LIG C i 0 14)
    (LIG C f 0 13)
    (LIG C l 0 15)
    (KRN 0 47 R 0.077779)
    (KRN 0 77 R 0.077779)
    ...
  )
)
...

```

As you can see, the general format is a list of parenthesized *properties*, nested where necessary.

- The first few items (FAMILY, FACE, and so on) are the so-called *headerbyte* information from Metafont, giving general information about the font.
- The FONTDIMEN property defines the T_EX `\fontdimen` values.
- The LIGTABLE property defines the ligature and kerning table. LIG properties define ligatures: in the example above, an ‘f’ (in the ‘LABEL’) followed by an ‘i’ is a ligature, i.e., a typesetting program like T_EX replaces those two consecutive characters by the character at position octal ‘014 in the current font—presumably the ‘fi’ ligature. KRN properties define kerns: if an ‘f’ is followed by character octal ‘047 (an apostrophe), T_EX inserts a small amount of space between them: 0.077779 times the design size the font was loaded at (about three-quarters of a printer’s point by default in this case, or .001 inches).
- The CHARACTER property defines the dimensions of a character: its width, height, depth, and italic correction, also in design-size units, as explained in the previous item. For our example ‘f’, the depth is zero, so that property is omitted. T_FtoP_L also inserts any kerns and ligatures for this character as a comment.

10.7 PLtoTF: Property list to T_EX font metric conversion

PLtoTF translates a property list file (as output by TFtoPL, for example) to T_EX font metric (TFM, see [Section “Metric files” in Dvips](#)) format. It’s much easier for both programs and humans to create the (plain text) property list files and let PLtoTF take care of creating the binary TFM equivalent than to output TFM files directly. Synopsis:

```
pltotf [option]... plfile[.pl] [tfmfile[.tfm]]
```

If *tfmfile* (extended with ‘.tfm’ if necessary) is not specified, the TFM file is written to the basename of ‘*plfile.tfm*’, e.g., ‘`pltotf /wherever/cmr10.pl`’ creates ‘`./cmr10.tfm`’. (Since TFM files are binary, writing to standard output by default is undesirable.)

The only options are ‘`-verbose`’, ‘`-help`’, and ‘`-version`’ (see [Section 3.2 \[Common options\]](#), page 7).

For an example of property list format, see the previous section.

10.8 VFtoVP: Virtual font to virtual property lists

VFtoVP translates a virtual font metric (VF, see [Section “Virtual fonts” in Dvips](#)) file and its accompanying T_EX font metric (TFM, see [Section “Metric files” in Dvips](#)) file (as output by VPtoVF, for example) to *virtual property list format* (a list of parenthesized items describing the virtual font) that humans can edit or read. This program is mostly used by people debugging virtual font utilities. Synopsis:

```
vftovp [option]... vfname[.vf] [tfmname[.tfm] [vplfile[.vpl]]]
```

The fonts *vfname* and *tfmname* (extended with ‘.vf’ and ‘.tfm’ if necessary) are searched for in the usual places (see [Section “Supported file formats” in Kpathsea](#)). To see all the relevant paths, set the environment variable `KPATHSEA_DEBUG` to ‘-1’ before running the program. If *tfmname* is not specified, *vfname* (without a trailing ‘.vf’) is used.

If *vplfile* (extended with ‘.vpl’ if necessary) is not specified, the property list file is written to standard output. The property list file can be converted back to VF and TFM format by the companion program VFtoVP (see the next section).

The program accepts the following option, as well as the standard ‘`-verbose`’, ‘`-help`’ and ‘`-version`’ (see [Section 3.2 \[Common options\]](#), page 7):

‘`-charcode-format=type`’

Output character codes in the PL file according to *type*: either ‘`octal`’ or ‘`ascii`’. Default is ‘`ascii`’ for letters and digits, octal for all other characters. Exception: if the font’s coding scheme starts with ‘`TeX math sy`’ or ‘`TeX math ex`’, all character codes are output in octal.

In ‘`ascii`’ format, character codes that correspond to graphic characters, except for left and right parentheses, are output as a ‘`C`’ followed by the single character: ‘`C K`’, for example. In octal format, character codes are output as the letter ‘`O`’ followed by octal digits, as in ‘`O 113`’ for ‘`K`’.

‘`octal`’ format is useful for symbol and other non-alphabetic fonts, where using ASCII characters for the character codes is merely confusing.

10.9 VPtoVF: Virtual property lists to virtual font

VPtoVF translates a virtual property list file (as output by VFtoVP, for example) to virtual font (VF, see [Section “Virtual fonts” in *Dvips*](#)) and \TeX font metric (TFM, see [Section “Metric files” in *Dvips*](#)) files. It’s much easier for both programs and humans to create the (plain text) property list files and let VPtoVF take care of creating the binary VF and TFM equivalents than to output them directly. Synopsis:

```
vptovf [option]... vplfile[.vpl] [vffile[.vf] [tfmfile[.tfm]]]
```

If *vffile* (extended with ‘.vf’ if necessary) is not specified, the VF output is written to the basename of ‘*vplfile.vf*’; similarly for *tfmfile*. For example, ‘vptovf /wherever/ptmr.vpl’ creates ‘./ptmr.vf’ and ‘./ptmr.tfm’.

The only options are ‘-verbose’, ‘-help’, and ‘-version’ (see [Section 3.2 \[Common options\]](#), page 7).

10.10 Font utilities available elsewhere

The Web2c complement of font utilities merely implements a few basic conversions. Many other more sophisticated font utilities exist; most are in ‘*CTAN:/fonts/utilities*’ (for CTAN info, see [Section “unixtex.ftp” in *Kpathsea*](#)). Here are some of the most commonly-requested items:

- AFM (Adobe font metric) to TFM conversion: see [Section “Invoking afm2tfm” in *Dvips*](#), and ‘*CTAN:/fonts/utilities/afm2tpl*’.
- BDF (the X bitmap format) conversion: <ftp://ftp.tug.org/tex/bdf.tar.gz>.
- Creating fonts using MetaPost: MetaType1. <ftp://bop.eps.gda.pl/pub/metatype1>. This is used to create the excellent Latin Modern font family (‘*CTAN:/fonts/lm*’), which extends Computer Modern to a vast repertoire of scripts.
- Editing of bitmap fonts: Xbfe from the GNU font utilities mentioned below; the X BDF-editing programs available from <ftp://ftp.x.org/R5contrib/xfed.tar.Z> and <ftp://ftp.x.org/R5contrib/xfedor.tar.Z>; and finally, if your fonts have only 128 characters, you can use the old gftopxl, pxtoch, and chtopx programs from <ftp://ftp.tug.org/tex/web>.
- Editing of outline fonts: FontForge, fontforge.sourceforge.net. This is a very elaborate program with support for many outline formats (Type 1, OpenType, TrueType, ...), and many advanced font editing features.
- PK bitmaps from PostScript outline fonts: gsftopk from the ‘xdvi’ distribution. Alternatively, ps2pk, from ‘*CTAN:/fonts/utilities/ps2pk*’.
- PostScript Type 1 font format conversion (i.e., between PFA and PFB formats): <http://www.lcdf.org/type>.
- Scanned image conversion: the (aging) GNU font utilities convert type specimen images to Metafont, PostScript, etc.: <http://www.gnu.org/software/fontutils/>.
- Tracing bitmaps to fitted outlines: Autotrace (<http://autotrace.sourceforge.net>), Potrace (<http://potrace.sourceforge.net>). For Metafont fonts, either of the two programs mftrace (<http://www.xs4all.nl/~hanwen/mftrace>) or textrace (<http://textrace.sourceforge.net>) make the job easier.
- Virtual font creation: ‘*CTAN:/fonts/utilities/fontinst*’.

Appendix A Legalisms

In general, each file has its own copyright notice stating the copying permissions for that file. Following is a summary.

The Web2c system itself and most of the original WEB source files are public domain.

`'tex.web'`, the $\text{MLT}_{\text{E}}\text{X}$ code, `'mf.web'`, and `'bibtex.web'`, are copyrighted by their authors. They may be copied verbatim, but may be modified only through a `'ch'` file.

MetaPost-related files, including `'mp.web'` itself, are copyrighted under X-like terms; the precise notice is included below.

Finally, the Kpathsea library is covered by the GNU Lesser General Public License (see [Section “Introduction” in *Kpathsea*](#)). Therefore, the *binaries* resulting from a standard Web2c compilation are also covered by the LGPL; so if you (re)distribute the binaries, you must also (offer to) distribute the complete source that went into those binaries. See the file `'LGPL'` for complete details on the LGPL.

The following notice must be included by the terms of the MetaPost copyright.

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that the copyright notice and this permission notice and warranty disclaimer appear in supporting documentation, and that the names of AT&T Bell Laboratories or any of its entities not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

AT&T disclaims all warranties with regard to this software, including all implied warranties of merchantability and fitness. In no event shall AT&T be liable for any special, indirect or consequential damages or any damages whatsoever resulting from loss of use, data or profits, whether in an action of contract, negligence or other tortious action, arising out of or in connection with the use or performance of this software.

Appendix B References

1. Kpathsea: See [Section “Top” in *Kpathsea*](#).
2. Dvips and Afm2tfm: See [Section “Top” in *Dvips*](#).
3. The T_EX Users Group: <http://www.tug.org>. For an introduction to the T_EX system, see <http://tug.org/begin.html>.
4. TUGboat: <http://tug.org/TUGboat>.
5. T_EX and computer typesetting in general:
<ftp://ftp.math.utah.edu/pub/tex/bib/texbook1.bib>.
6. For a bibliography of formal articles and technical reports on the T_EX project, see the books *T_EX: The Program* or *Metafont: The Program* cited below.
7. [Bil87] Neenie Billawala. Write-white printing engines and tuning fonts with Metafont. *TUGboat*, 8(1):29–32, April 1987.
8. [GMS94] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The L_AT_EX Companion*. Addison-Wesley, Reading, MA, USA, 1994.
9. [Hob89] John D. Hobby. A Metafont-like system with PS output. *TUGboat*, 10(4):505–512, December 1989.
10. [Hob92] John D. Hobby. A User’s Manual for MetaPost. Technical Report CSTR-162, AT&T Bell Laboratories, 1992.
11. [Hob93] John D. Hobby. Drawing Graphs with MetaPost. Technical Report CSTR-164, AT&T Bell Laboratories, 1993.
12. [HS91] Samuel P. Harbison and Guy L. Steele Jr. *C—A Reference Manual*. Prentice-Hall, Upper Saddle River, NJ 07458, USA, third edition, 1991. An authoritative reference to the C programming language, and a good companion to Kernighan and Ritchie.
13. [KL93] Donald E. Knuth and Silvio Levy. *The CWEB System of Structured Documentation, Version 3.0*. Addison-Wesley, Reading, MA, USA, 1993.
14. [Knu84] Donald E. Knuth. A torture test for T_EX. Report No. STAN-CS-84-1027, Stanford University, Department of Computer Science, 1984.
15. [Knu86a] Donald E. Knuth. A Torture Test for METAFONT. Report No. STAN-CS-86-1095, Stanford University, Department of Computer Science, 1986.
16. [Knu86b] Donald E. Knuth. *The T_EXbook*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.
17. [Knu86c] Donald E. Knuth. *T_EX: The Program*, volume B of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.
18. [Knu86d] Donald E. Knuth. *The METAFONTbook*, volume C of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.
19. [Knu86e] Donald E. Knuth. *METAFONT: The Program*, volume D of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.
20. [Knu86f] Donald E. Knuth. *Computer Modern Typefaces*, volume E of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.
21. [Knu89] Donald E. Knuth. The errors of T_EX. *Software—Practice and Experience*, 19(7):607–681, July 1989. This is an updated version of Knuth:1988:ET.

- 22. [Knu90] Donald Knuth. Virtual Fonts: More Fun for Grand Wizards. *TUGboat*, 11(1):13–23, April 1990.
- 23. [Knu92] Donald E. Knuth. *Literate Programming*. CSLI Lecture Notes Number 27. Stanford University Center for the Study of Language and Information, Stanford, CA, USA, 1992.
- 24. [Lam94] Leslie Lamport. *LaTeX: A Document Preparation System: User's Guide and Reference Manual*. Addison-Wesley, Reading, MA, USA, second edition, 1994. Reprinted with corrections in 1996.
- 25. [Lia83] Franklin Mark Liang. Word hy-phen-a-tion by com-pu-ter. Technical Report STAN-CS-83-977, Stanford University, August 1983.
- 26. [Mac91] Pierre A. MacKay. Looking at the pixels: Quality control for 300 dpi laser printer fonts, especially Metafonts. In Robert A. Morris and Jacques Andre, editors, *Raster Imaging and Digital Typography II—Papers from the second RIDT meeting, held in Boston, Oct. 14–16, 1991*, pages 205–215, New York, 1991. Cambridge University Press.
- 27. [Spi89] Michael D. Spivak. *LAMST_EX, The Synthesis*. The T_EXplorators Corporation, 3701 W. Alabama, Suite 450-273, Houston, TX 77027, USA, 1989.
- 28. [Spi90] Michael D. Spivak. *The Joy of T_EX—A Gourmet Guide to Typesetting with the AMST_EX macro package*. American Mathematical Society, Providence, RI, USA, 2nd revised edition, 1990.

Index

#

'define' options 3

\$

\$ expansion in filenames 13

%

% magic number 11

-

- starting a filename 7
 - starts option names 7
 -- starts option names 7
 --disable-dump-share configure option 11
 --disable-mf-nowin 27
 --enable-epsfwin 27
 --enable-hp2627win 27
 --enable-ipc configure option 15
 --enable-mftalkwin 27
 --enable-next 27
 --enable-regiswin 27
 --enable-suntoolswin 27
 --enable-tektronixwin 27
 --enable-unitermwin 27
 --help common option 7
 --verbose common option 7
 --version common option 7
 --with-editor=*cmd* 12
 --with-mf-x-toolkit=*kit* 27
 --with-x 27
 -8bit 9
 -base=*base* 11
 -base=*dumpname* 8
 -change=*chfile* 29
 -charcode-format=*type* 49, 51
 -D compiler options 3
 -disable-write18 16
 -dpi=*real* 41
 -enable-write18 16
 -enc 15
 -file-line-error 7
 -file-line-error-style 7
 -fmt=*dumpname* 8
 -fmt=*fmt* 11
 -geometry, supported with X_t 27
 -halt-on-error 8
 -images 47
 -ini 8, 10
 -interaction=*string* 8
 -ipc 15
 -ipc-start 15

-jobname=*string* 8
 -kpathsea-debug=*number* 8
 -length=*number* 37
 -loose 38
 -lowercase 37
 -magnification=*integer* 40, 41
 -max-pages=*n* 40, 41
 -mem=*dumpname* 8
 -mem=*mem* 11
 -min-crossrefs=*n* 35
 -mixedcase 37
 -mktex=*filetype* 15, 25
 -mltex 15
 -mnemonics 47
 -no-file-line-error 7
 -no-mktex=*filetype* 15, 25
 -no-parse-first-line 8
 -no-shell-escape 16
 -oem 9
 -output-comment=*string* 16
 -output-directory 8, 9
 -output-level=*n* 41
 -overflow-label-offset=*points* 28
 -page-start=*page-spec* 40, 41
 -parse-first-line 8
 -progname=*string* 8, 11
 -recorder 9
 -shell-escape 16
 -shell-restricted 16
 -show-opcodes 41
 -strict 38
 -style=*mftfile* 30
 -T 33
 -terse 35
 -tex=*texprogram* 33
 -translate-file=*tcxfile* 9
 -troff 33
 -underline 38
 -uppercase 37
 -x 38

.

., used for output 9
 .2602gf 24
 .aux cross-reference files 35
 .base 25
 .bbl bibliography files 35
 .bib bibliography databases 35
 .blg Bib_T_EX log file 35
 .fmt 16
 .mem 33
 .mf 24
 .mp 31
 .mps files and PDF 32

`.nnn` PostScript figures 31
`.nnngf` generic fonts 24
`.tcx` character translation files 9, 19
`.tex` 14
`.tfm` output 24, 31
`.Xdefaults` 27
`.Xresources` 27

^

^^ notation, avoiding 20

\

`\bibliography` 35
`\bibliographystyle` 35
`\charsubdef` and `MLTeX` 18
`\countn` 40, 41
`\font` and dynamic generation 14
`\fontdimen` 50
`\immediate\write18` 21
`\input` filenames 12
`\input`, and pipes 22
`\mag` 40, 41
`\openin`, and pipes 22
`\openout` and security 14
`\openout`, and pipes 22
`\output` routine, and `\write` 22
`\pdfshellescape` 22
`\string` 13
`\tracingcharsubdef` and `MLTeX` 19
`\tracinglostchars` and `MLTeX` 19
`\write18` shell escape extension 21

~

~ expansion in filenames 13

2

2602gf 24

8

8 bit clean 9
 8 bit clean output, specifying 9
 8-bit characters 19

A

`abbrv.bst` 36
 accented character 19
 accents, hyphenating words with 18
 acknowledgements 1
`acm.bst` 36
 Ada, WEB for 37
 additional Make targets 4
 AFM to TFM conversion 52

`afm2tfm` 52
`afmtopl` 52
 Aleph 23
 aliases for fonts 9
`alpha.bst` 36
 American Mathematical Society, typesetting
 system 17
`AMSTeX` 17
`apalike.bst` 36
 architecture dependencies 11
 array limit, fixed 5
 array sizes 5
 assembly language routines 4
`autotrace` 52
 Awk, WEB for 37

B

base file, determining 11
 base files 25
 base files, need mode definitions 26
 base files, plain only 26
 base files, sharing 11
`bases` Make target 4
 basic BibTeX style files 36
 basic fonts and macros 2
 batch languages 14
 BDF and GF conversion 52
`beginfig` 31
 Berry, Karl 1
 BIBINPUTS, search path for bib files 35
 bibliographies, creating 35
 bibliography 54
 bibliography items, cross-referenced 35
`bibtex` 35
 BibTeX 35
 BibTeX collection 35
 BibTeX style files 36
 BigEndian machines 11
 binaries, linking 8
 blank lines, in TCX files 20
 boxes, memory for 5
 breakpoints, memory for 5
 Breitenlohner, Peter 1
 BSTINPUTS, search path for bst files 35
`btex` for MetaPost labels 31
`btxdoc.bib` 35
`btxdoc.tex` 35
`btXHak.tex` 35
 byte position 46, 48
 byte swapping 11

C

`c-sources` Makefile target 4
 change files, and MFT 29
 change files, and Tangle 37
 change files, and Weave 38

changing error messages style	7
character codes, in GFtype output	48
character codes, in PKtype output	46
character codes, in TCX files	20
character proofs of fonts	28
CHARACTER property	50
character translation files	19
CHARDP property	50
CHARHT property	50
CHARIC property	50
CHARWD property	50
chtopx	52
class name for Metafont	27
cm.base	26
cmbase.mf	26
cmbase.mft	30
cmmf.base not recommended	26
color, in DVItMP	34
comments, in TCX files	20
comments, MFT control	29
common options	7
commonalities	7
compilation	2
compile-time options	2, 3
Computer Modern fonts, and Troff	32
Computer Modern macros	26
<i>Computer Modern Typefaces</i> , production of	30
configuration	2
configuration file reading	9
configuration file values	5
configuration, compile-time	2
configure --with/--enable options	2
CONTENTS.tex	38
control sequence names, space for	5
conventions for options,	7
conversion, DVI to plain text	40
conversion, GF to PK	45
conversion, GF to plain text	47
conversion, PK to GF	45
conversion, PK to plain text	45
conversion, property list to TFM	51
conversion, property list to VF	52
conversion, TFM to property list	49
conversion, VF to VPL	51
copyright notices	53
Cork encoding and ISO input	20
creating memory dumps	10
cross-referenced bibliography items	35
cross-references, omitting	38
current directory, used for output	9
Curtis, Pavel	1
Cweb	37
CWEB	37

D

date and time, in memory dumps	11
debugging DVI utilities	41

debugging flags, specifying	8
decimal character codes, in TCX files	20
dependencies, hardware	11
design-size units	50
device definitions, for Metafont	26
device-independent width	46, 48
directory structure	2
DISPLAY	27
dot files, written by T _E X programs	14
downloading of fonts for MetaPost labels	32
DrawingServant	27
dump file	8
dumping memory	10
DVI comment, specifying	16
DVI files, converting to MPX	34
DVI files, explained	44
DVI format definition	40
DVI opcodes, showing	41
DVI utilities	40
dvicopy	40
dvitomp	34
dvitype DVI validation	40
dvitype output example	41
dvitype.web	40
dx horizontal escapement	46, 48
dy vertical escapement	48
dynamic array allocation	5
dynamic Metafont mode definitions with smode	26
dynamic packing variable	46

E

e response at error prompt	12
e-circumflex	19
e-T _E X	23
e.mft	30
EC fonts	14, 24
editing of bitmap fonts	52
editor invoked at error	12
eight-bit characters in filenames	13
empty.tcx	20
endian dependencies	11
eoc GF command	48
Eplain	18
epsf	27
errors, editor invoked at	12
escapement, horizontal	46, 48
escapement, vertical	48
etex for MetaPost labels	31
executables, shared initial and virgin	10
exit status, of shell escape	22
expanded plain format	18
extensions to T _E X	23
extra_mem_bot	5

F

FACE property	50
FAMILY property	50
Ferguson, Michael	18
file formats for fonts	44
file recorder	9
filename conventions, in input files	12
filenames starting with ‘-’	7
first line of the main input file	9
fixed-point arithmetic	4
FIXPT	4
flag byte	46
floating-point arithmetic	4
floating-point values	11
fnt file, determining	11
fnt files	16
fnt files, sharing	11
fnts Make target	4
font aliases	9
font character code, translating	21
font design	24
font downloading for MetaPost labels	32
font file formats	44
font proofs	28
font utilities	44
font utilities, non-Web2c	52
font_mem_size	5
fontforge	52
fontinst, for creating virtual fonts	52
fonts, basic	2
fontutils	52
format files	17
formats for T _E X	17
formats Make target	4
fraction routines	4
Free Software Foundation documentation system	18
freedom of Web2c	1
ftp.math.utah.edu	35

G

generating source specials	16
geometric designs	24
geometric font scaling	44
geometry for Metafont	27
getopt_long_only	7
GF files, explained	44
GF files, output by Metafont	24
GF format definition	44
GF output	24
GF, converting PK to	45
GF, converting to PK	45
gftodvi	28
gftopk	45
gftopxl	52
gftype GF validation	47
gftype.web	44

glue ratio representations	11
glue, memory for	5
glyph substitutions	18
gray font	28
Gruff, Billy Goat	4
gsftopk	52

H

Harbison, Samuel P.	12
hardware and memory dumps	11
hash table, increasing size of	5
hash_extra	5
headerbyte information	50
height, in pixels	46
help, online	7
Henry, Patrick	1
Herberts, Mathias	27
hex character codes, in TCX files	20
history	1
Hobby, John	1
horizontal escapement	46, 48
hp2627	27
human languages, supported in T _E X	18
human-readable text, converting DVI to	40
human-readable text, converting GF to	47
human-readable text, converting PK to	45
human-readable text, converting TFM to	49
human-readable text, converting VF to	51
hypertext	23
hyphenation and languages	18
hyphenation patterns, creating	21

I

ice cream	1
identifier case	37
identifier collisions	38
identifier length	37
identifiers with underlines	38
ieeeetr.bst	36
il1-t1.tcx	20
il2-t1.tcx	20
Info format	18
initial form, enabling	8
initial Metafont	25
initial MetaPost	33
initial programs	10
initial T _E X	16
initializations, lengthy	10
input filenames	12
install-bases Make target	4
install-fnts Make target	4
install-formats Make target	4
install-mems Make target	4
installation	2
interaction between TCX files and ‘-8bit’	21
interaction mode	8

international characters 19
 introduction 1
 IPC 22
 IPC_DEBUG 4, 22

J

job name 8

K

kerning table, in TFM files 50
 keyboard character code, translating 21
 Knuth, Donald E. 1, 29
 KPATHSEA_DEBUG 8
 KRN property 50

L

label font 28
 LABEL property 50
 language support in T_EX 18
 languages, hyphenation rules for 21
 L^AT_EX 17
 Latin Modern 52
 left side bearing 46, 48
 legalisms 53
 licensing terms 1
 LIG property 50
 ligature table, in TFM files 50
 LIGTABLE property 50
 linking binaries 8
 links to binaries 11
 literate programming 37
 LittleEndian machines 11
 log file, BibT_EX 35
 Lua 23
 luaT_EX 23

M

machine dependencies 11
 machine-readable, converting property lists to
 51, 52
 MacKay, Pierre 45
 macro packages, major T_EX 17
 macros, basic 2
 macros, predefining in memory dumps 10
 magnification 40, 41
 main_memory 5
 Make targets, additional 4
 Martin, Rick 1
 Mathematical Reviews 17
 mathematical typesetting 14
 mem file, determining 11
 mem files 33
 mem files, plain only 33
 mem files, sharing 11

memory dump to use, determining 11
 memory dumps 10
 memory dumps and hardware 11
 memory dumps, contain date and time 11
 memory dumps, creating 10
 mems Make target 4
 meta characters in filenames 13
 Metafont 24
 Metafont geometry 27
 Metafont graphics 27
 Metafont input files 24
 Metafont invocation 24
 Metafont meets PostScript 31
 Metafont online support, new devices 28
 Metafont source, prettyprinting 29
 Metafont, compatibility in MetaPost 33
 Metafont, initial 25
 Metafont, MetaPost, and T_EX 10
 MetaFun, enhanced MetaPost 33
 metafun.mp 33
 MetaPost 31
 MetaPost and plain Metafont compatibility 33
 MetaPost input files 31
 MetaPost invocation 31
 MetaPost source, prettyprinting 30
 MetaPost, initial 33
 MetaPost, T_EX, and Metafont 10
 metatype1 52
 mf 24
 mf.base 25
 MFEDIT 12
 mfplain.mem 33
 mfput 24
 mft 29
 mftalk 27
 MFTERM 27
 mftmac.tex 29
 micro-typography 23
 mktexmf, disabling 24
 mktexfm, disabling 14
 mltex 18
 MLT_EX, enabling 15
 mode needed to run Metafont 24
 mode_def 26
 mode_setup 26
 modes file needed for Metafont 26
 modes.mf recommended modes file 26
 Morgan, Tim 1
 Morris, Bob 1
 MPEDIT 12
 mpgraph.ps 31
 mpman.ps 31
 mpost 31
 mpost, reason for name change 2
 mpost.mem 33
 mpout 31
 mproof.tex 32
 mptrap Make target 4

mptrap test.....	4
mptrap.readme	4
MPX files, converting from DVI files	34
Multi-lingual T _E X	18

N

N tilde	19
new graphics support for Metafont	28
newrow GF command	48
next	27
NO_X11WIN	27
non-Unix system, compiling on	4
non-windows-capable Metafont	27
NUL, not allowed in filenames	13

O

octal character codes, in TCX files	20
offset for overflow labels	28
online Metafont graphics	27
opcodes, showing DVI	41
optical font scaling	44
option conventions	7
origin	46
output directory, specifying	8, 9
output file location	9
output files, written by T _E X programs	14
output_comment for DVI files	16
overflow label offset	28

P

packet length	46
page, starting	40, 41
parsing the first line	8
Pascal, creating from WEB	37
patgen	21
path searching	9
path searching debugging	8
PDF	23
PDF, and .mps files	32
pdfT _E X	23
permissions, legal	53
PFA and PFB conversion	52
pfaedit	52
PiCT _E X, increasing memory for	5
pipes, reading and writing	22
pixel height	46
pixel width	46
PK bitmaps from PostScript	52
PK files, explained	44
PK files, not output by Metafont	24
PK format definition	44
PK, converting GF to	45
PK, converting to GF	45
pktogf	45
pktype PK validation	45

pktype.web	44
PL files, explained	44
plain Metafont, compatibility in MetaPost	33
plain text, converting DVI to	40
plain text, converting GF to	47
plain text, converting PK to	45
plain text, converting TFM to	49
plain text, converting VF to	51
plain.base	25
plain.bst	36
plain.fmt	17
plain.mem	33
plain.mft	30
pltotf	51
pool file, writing	37
Poole, Simon	27
pooltype	38
portable filenames	13
PostScript fonts, and Troff	32
PostScript meets Metafont	31
PostScript output	31
PostScript to PK bitmaps	52
PostScript Type 1 font conversion	52
PostScript, and font scaling	44
potrace	52
Pragma ADE	33
predefined macros and memory dumps	10
prettyprinting Metafont source	29
prettyprinting WEB programs	38
primitives, new	23
printable characters, specifying	20
printer characteristics, for Metafont	26
production use	10
program name, determines memory dump	11
program names, special	8, 15
prologues	33
prologues , and EPSF output	32
proof mode	24
proof sheets, of fonts	28
property list format	49
property list, converting TFM to	49
property list, converting VF to virtual	51
ps2pk	52
psfonts.map , read by MetaPost	32
PXL files, explained	44
pxtoch	52

R

Raichle, Bernd	18
reading, additional	1
readonly directory, running T _E X in	9
reallocation of arrays	5
redefined character substitutions	19
reference pixel	46
references	54
regis	27
Regis graphics support	27

regression testing 16
 repeated rows 47
 representation of strings 39
 restricted shell escapes 21
 right side bearing 46, 48
 Rokicki, Tomas 1
 run length encoded bitmaps 47, 48
 runtime options 5

S

scaled pixels 46, 48
 scaling of fonts 44
 scanned images of fonts 52
 security, and `\openout` 14
 security, and output files 14
 security, and shell escapes 21
 security, and `write` 32
 shapes 24
 sharing memory dumps 11
 shell commands in `TEX` 21
`shell_escape` enabling in `TEX` 21
`shell_escape_commands` 22
`siam.bst` 36
 side bearings 46, 48
 slant font 28
 slides, producing 18
`SliTEX` 18
 small Metafont memory and modes 26
`smode` and dynamic Metafont mode definition .. 26
 sockets 22
 space-terminated filenames 12
 Spiderweb 37
 Stallman, Richard 1
 starting page 40, 41
 Steele Jr., Guy L. 12
 stopping at the first error 8
 strategy, overall 1
 string numbers, displaying 38
 string pool, writing 37
 string representation 39
 style design, for Bib`TEX` 35
 style files 30
 substitutions of font glyphs 18
`sun` 27
`sun-gfx.c` 27
 Suntools 27
 SunView 27
 swap space, as array limit 5
 swapping bytes 11
 syntax of TCX files 20
`system` C library function 21
 system command 21

T

T1 encoding and ISO input 20
 Tachikawa, Elizabeth 45

`tangle` 37
 targets, additional Make 4
 TCX character translation files 19
`tek` 27
 Tektronix 27
 Tektronix 4014 27
`TERM` 27
 terminator for filenames 12
 terse output 35
`tex` 14
`tex.fmt` 17
`TEX`, bibliographies for 35
`TEX`, creating from Metafont 29
`TEX`, creating from WEB 38
`TEX`, description of 14
`TEX`, extensions to 23
`TEX`, format packages for 17
`TEX`, initial 16
`TEX`, input files found 14
`TEX`, invocation 14
`TEX`, Metafont, and MetaPost 10
`TEX`, Web2c implementation of 1
`TEXBIB`, search path for bib files 35
`TEXEDIT` 12
`texfonts.map` 9
`Texinfo` 18
`texmf.cnf` 9
`texmfmp.c` 28
`TEXMFOUTPUT`, used for reading 9
`TEXMFOUTPUT`, used if ‘.’ unwritable 9
`texput` 14
 TFM files, converting property lists to 51
 TFM files, explained 44
 TFM files, memory for 5
 TFM files, output by Metafont 24
 TFM files, output by MetaPost 31
 TFM width of characters 46, 48
`tftopl` 49
 three programs 10
 time and date, in memory dumps 11
 title font 28
 toolkits, X 27
 torture tests 4
 translation file for `TEX`, specifying 9
 translation from WEB to C 1
`trap` Make target 4
 trap test 4
`trapman.tex` 4
 Trickey, Howard 1
`trip` Make target 4
 trip test 4
`tripman.tex` 4
`triptrap` Make target 4
 Troff, and MetaPost 32
 Troff, WEB for 37
 Trojan horses and `TEX` programs 14
 TUGboat bibliography 35
 Type 1 conversion 52

type design, personal	26
type programs, DVI	40
type programs, GF	47
type programs, PK	45
type programs, pool	38
typeface families	24
typeface specimen sheets	52
typesetting	14

U

Unicode	23
Unicode input	15
uniterm	27
unsrt.bst	36
using local codepage to display messages	9
UTF-8 input	15

V

validation, of DVI files	40
validation, of GF files	47
validation, of PK files	45
validation, of TFM files	49
validation, of VF files	51
verbose BibTeX output, suppressing	35
verbosity, enabling	7
version number, finding	7
vertical escapement	48
VF files, converting property lists to	52
vftovp	51
virgin programs	10
virtual font creation	52
virtual fonts, expanding	40
virtual-fonts.knuth	44
virtualfonts.txt	44
vptovf	52

W

weave	38
WEB	37
web environments, and security	22
WEB pool files, displaying	38
WEB programs, compiling	37
WEB programs, typesetting	38
WEB2C , search path for TCX files	20
Weber, Olaf	1
webmac.tex	38
webman.tex	37
whitespace, in TCX files	20
whitespace-terminated filenames	12
width, device-independent	46, 48
width, in pixels	46
word processor, not	14
writing memory dumps	10

X

X bitmap fonts	52
X class name for Metafont	27
x offset	46
X resources	27
X toolkits and Metafont	27
xampl.bib	35
xbfe , bitmap font editor	52
XeTeX	23
xfed , bitmap font editor	52
xfedor , bitmap font editor	52
Xlib	27
Xlib support	27
Xt	27
Xt support	27
xterm	27

Y

y offset	46
----------------	----