MAPS

NUMMER 48 • VOORJAAR 2018

NEDERLANDSTALIGE TEX GEBRUIKERSGROEP

Voorzitter
Hans Hagen
voorzitter@ntg.nl

Secretaris
Taco Hoekwater
secretaris@ntg.nl

Penningmeester
Ferdy Hanssen
penningmeester@ntg.nl

Bestuursleden
Frans Goddijn
Piet van Oostrum

Postadres
Nederlandstalige TEX Gebruikersgroep
Baarsjesweg 268-I
1058 AD Amsterdam

ING bankrekening
IBAN: NL53INGB0001306238
BIC: INGBNL2A

E-mail bestuur
ntg@ntg.nl

E-mail MAPS redactie
maps@ntg.nl

WWW
www.ntg.nl

De **Nederlandstalige TEX Gebruikersgroep (NTG)** is een vereniging die tot doel heeft de kennis en het gebruik van TEX te bevorderen. De NTG fungeert als een forum voor nieuwe ontwikkelingen met betrekking tot computergebaseerde documentopmaak in het algemeen en de ontwikkeling van 'TEX and friends' in het bijzonder. De doelstellingen probeert de NTG te realiseren door onder meer het uitwisselen van informatie, het organiseren van conferenties en symposia met betrekking tot TEX en daarmee verwante programmatuur.
De NTG biedt haar leden ondermeer:

- Tweemaal per jaar een NTG-bijeenkomst.
- Het NTG-tijdschrift MAPS.
- De 'TEX Live'-distributie op DVD/CDROM inclusief de complete CTAN software-archieven.
- Verschillende discussielijsten (mailing lists) over TEX-gerelateerde onderwerpen, zowel voor beginners als gevorderden, algemeen en specialistisch.
- De FTP server `ftp.ntg.nl` waarop vele honderden megabytes aan algemeen te gebruiken 'TEX-producten' staan.
- De WWW server `www.ntg.nl` waarop algemene informatie staat over de NTG, bijeenkomsten, publicaties en links naar andere TEX sites.
- Korting op (buitenlandse) TEX-conferenties en -cursussen en op het lidmaatschap van andere TEX-gebruikersgroepen.

Lid worden kan door overmaking van de verschuldigde contributie naar de NTG-giro (zie links); vermeld IBAN zowel als SWIFT/BIC en selecteer shared cost. Daarnaast dient via `www.ntg.nl` een informatieformulier te worden ingevuld. Zonodig kan ook een papieren formulier bij het secretariaat worden opgevraagd.
De contributie bedraagt € 40. Voor studenten geldt een tarief van € 20. Dit geeft alle lidmaatschapsvoordelen maar *geen stemrecht.* Een bewijs van inschrijving is vereist. Een gecombineerd NTG/TUG-lidmaatschap levert een korting van 10% op beide contributies op. De prijs in euro's wordt bepaald door de dollarkoers aan het begin van het jaar. De ongekorte TUG-contributie is momenteel $105.

Afmelding kan met ingang van het volgende kalenderjaar door opzegging per e-mail aan de penningmeester.

MAPS bijdragen kunt u opsturen naar `maps@ntg.nl`, bij voorkeur in LATEX- of ConTEXt formaat. Bijdragen op alle niveaus van expertise zijn welkom.

Productie. De Maps wordt gezet met behulp van een LATEX class file en een ConTEXt module. Het pdf bestand voor de drukker wordt aangemaakt met behulp van pdftex 1.40.19 en luatex 1.0.8 draaiend onder MacOS X 10.13. De gebruikte fonts zijn Linux Libertine, het niet-proportionele font Inconsolata, schreefloze fonts uit de Latin Modern collectie, en de Euler wiskunde fonts, alle vrij beschikbaar.

TEX is een door professor Donald E. Knuth ontwikkelde 'opmaaktaal' voor het letterzetten van documenten, een documentopmaaksysteem. Met TEX is het mogelijk om kwalitatief hoogstaand drukwerk te vervaardigen. Het is eveneens zeer geschikt voor formules in mathematische teksten.
Er is een aantal op TEX gebaseerde producten, waarmee ook de logische structuur van een document beschreven kan worden, met behoud van de letterzet-mogelijkheden van TEX. Voorbeelden zijn LATEX van Leslie Lamport, $\mathcal{AMS}$-TEX van Michael Spivak, en ConTEXt van Hans Hagen.

Contents

Redactioneel, *Maps redactie* **1**

The TEX Live Guide – 2018, *Karl Berry, editor* **3**

Executing TEX, *Hans Hagen* **46**

Variable fonts, *Hans Hagen* **51**

TEX Gyre text fonts revisited, *Bogusław Jackowski, Piotr Pianowski & Piotr Strzelczyk* **59**

TLaunch, the TEX Live Launcher, *Siep Kroonenberg* **66**

`updmap` and `fmtutil` — past and future changes, *Norbert Preining* **70**

Privacybeleid Nederlandstalige TEX Gebruikersgroep **77**

Redactioneel

Door middel van de Maps willen we u op de hoogte houden van ontwikkelingen, ook om daarmee onze leden te danken voor hun trouwe steun aan de TEX ontwikkelaars. Verder bieden we ruimte aan lezers die anderen laten delen in hun ervaringen met TEX, MetaPost, fonts en aanverwanten. Aarzel dus niet ons artikelen te sturen. Hoewel het internet tegenwoordig een belangrijke bron van informatie is, blijft papier een functie vervullen binnen de vereniging. Dat past immers bij TEX!

Daar doen we het voor!

Veel leesplezier,

Uw redactie

The TEX Live Guide—2018

Karl Berry, editor
http://tug.org/texlive/

April 2018

This year's TEX Live is dedicated to our sadly departed colleague Staszek Wawrykiewicz.

Contents

1 Introduction **2**
1.1 TEX Live and the TEX Collection . . . 2
1.2 Operating system support . . . 3
1.3 Basic installation of TEX Live . . . 3
1.4 Security considerations . . . 3
1.5 Getting help . . . 3

2 Overview of TEX Live **4**
2.1 The TEX Collection: TEX Live, proTEXt, MacTEX . . . 4
2.2 Top level TEX Live directories . . . 4
2.3 Overview of the predefined texmf trees . . . 5
2.4 Extensions to TEX . . . 6
2.5 Other notable programs in TEX Live . . . 6

3 Installation **7**
3.1 Starting the installer . . . 7
3.1.1 Unix . . . 8
3.1.2 Mac OS X . . . 8
3.1.3 Windows . . . 8
3.1.4 Cygwin . . . 9
3.1.5 The text installer . . . 9
3.1.6 The expert graphical installer . . . 9
3.1.7 The simple wizard installer . . . 9
3.2 Running the installer . . . 10
3.2.1 Binary systems menu (Unix only) . . . 10
3.2.2 Selecting what is to be installed . . . 10
3.2.3 Directories . . . 12
3.2.4 Options . . . 13
3.3 Command-line install-tl options . . . 14
3.3.1 The `-repository` option . . . 14
3.4 Post-install actions . . . 14
3.4.1 Environment variables for Unix . . . 15
3.4.2 Environment variables: Global configuration . . . 15
3.4.3 Internet updates after DVD installation . . . 15
3.4.4 System font configuration for XeTEX and LuaTEX . . . 15
3.4.5 ConTEXt Mark IV . . . 16
3.4.6 Integrating local and personal macros . . . 16
3.4.7 Integrating third-party fonts . . . 17
3.5 Testing the installation . . . 17
3.6 Links for additional downloadable software . . . 18

1 INTRODUCTION 2

4 Specialized installations 19
4.1 Shared-user (or cross-machine) installations . . . 19
4.2 Portable (USB) installations . . . 19

5 tlmgr: Managing your installation 20
5.1 tlmgr GUI mode . . . 20
5.2 Other GUI interfaces for tlmgr . . . 21
5.3 Sample tlmgr command-line invocations . . . 21

6 Notes on Windows 22
6.1 Windows-specific features . . . 22
6.2 Additional software included on Windows . . . 22
6.3 User Profile is Home . . . 23
6.4 The Windows registry . . . 23
6.5 Windows permissions . . . 23
6.6 Increasing maximum memory on Windows and Cygwin . . . 24

7 A user's guide to Web2C 24
7.1 Kpathsea path searching . . . 25
7.1.1 Path sources . . . 26
7.1.2 Config files . . . 26
7.1.3 Path expansion . . . 26
7.1.4 Default expansion . . . 27
7.1.5 Brace expansion . . . 27
7.1.6 Subdirectory expansion . . . 27
7.1.7 List of special characters and their meaning: a summary . . . 27
7.2 Filename databases . . . 28
7.2.1 The filename database . . . 28
7.2.2 kpsewhich: Standalone path searching . . . 28
7.2.3 Examples of use . . . 29
7.2.4 Debugging actions . . . 30
7.3 Runtime options . . . 32

8 Acknowledgements 32

9 Release history 34
9.1 Past . . . 34
9.1.1 2003 . . . 34
9.1.2 2004 . . . 35
9.1.3 2005 . . . 36
9.1.4 2006–2007 . . . 37
9.1.5 2008 . . . 37
9.1.6 2009 . . . 38
9.1.7 2010 . . . 38
9.1.8 2011 . . . 39
9.1.9 2012 . . . 39
9.1.10 2013 . . . 40
9.1.11 2014 . . . 40
9.1.12 2015 . . . 41
9.1.13 2016 . . . 41
9.1.14 2017 . . . 42
9.2 Present—2018 . . . 42
9.3 Future . . . 43

1 Introduction

1.1 TEX Live and the TEX Collection

This document describes the main features of the TEX Live software distribution — TEX and related programs for GNU/Linux and other Unix flavors, Mac OS X, and Windows systems.

You may have acquired TEX Live by downloading, or on the TEX Collection DVD, which TEX user groups distribute among their members, or in other ways. Section 2.1 briefly describes the contents of the DVD. Both TEX Live and the TEX Collection are cooperative efforts by the TEX user groups. This document mainly describes TEX Live itself.

TEX Live includes executables for TEX, LATEX 2ε, ConTEXt, METAFONT, MetaPost, BIBTEX and many other programs; an extensive collection of macros, fonts and documentation; and support for typesetting in many different scripts from around the world.

For a brief summary of the major changes in this edition of TEX Live, see the end of the document, section 9 (p. 34).

1.2 Operating system support

TEX Live contains binaries for many Unix-based platforms, including GNU/Linux, Mac OS X, and Cygwin. The included sources can be compiled on platforms for which we do not provide binaries.

As to Windows: Windows 7 and later are supported. Windows Vista may still mostly work, but TEX Live will no longer even install on Windows XP or earlier. There are no special 64-bit executables for Windows, but the 32-bit executables should run on 64-bit systems.

See section 2.1 for alternate solutions for Windows and Mac OS X.

1.3 Basic installation of TEX Live

You can install TEX Live either from DVD or over the Internet (http://tug.org/texlive/acquire.html). The net installer itself is small, and downloads everything requested from the Internet.

The DVD installer lets you install to a local disk. You cannot run TEX Live directly from the TEX Collection DVD (or its `.iso` image), but you can prepare a runnable installation on, e.g., a USB stick (see section 4.2). Installation is described in later sections (p. 7), but here is a quick start:

- The installation script is named `install-tl`. It can operate in a "wizard mode" given the option `-gui=wizard` (default for Windows), a text mode given `-gui=text` (default for everything else), and an expert GUI mode given `-gui=perltk`. But see section 3.1.3 for Windows.
- One of the installed items is the 'TEX Live Manager' program, named tlmgr. Like the installer, it can be used in both GUI mode and in text mode. You can use it to install and uninstall packages and do various configuration tasks.

1.4 Security considerations

To the best of our knowledge, the core TEX programs themselves are (and always have been) extremely robust. However, the contributed programs in TEX Live may not reach the same level, despite everyone's best efforts. As always, you should be careful when running programs on untrusted input; for maximum safety, use a new subdirectory.

This need for care is especially urgent on Windows, since in general Windows finds programs in the current directory before anything else, regardless of the search path. This opens up a wide variety of possible attacks. We have closed many holes, but undoubtedly some remain, especially with third-party programs. Thus, we recommend checking for suspicious files in the current directory, especially executables (binaries or scripts). Ordinarily they should not be present, and definitely should not normally be created by merely processing a document.

Finally, TEX (and its companion programs) are able to write files when processing documents, a feature that can also be abused in a wide variety of ways. Again, processing unknown documents in a new subdirectory is the safest bet.

1.5 Getting help

The TEX community is active and friendly, and most serious questions end up getting answered. However, the support is informal, done by volunteers and casual users, so it's especially important that you do your homework before asking. (If you prefer guaranteed commercial support, you can forgo TEX Live completely and purchase a vendor's system; http://tug.org/interest.html#vendors has a list.)

Here is a list of resources, approximately in the order we recommend using them:

Getting started If you are new to TEX, the web page http://tug.org/begin.html gives a brief introduction to the system.

TEX FAQ The TEX FAQ is a huge compendium of answers to all sorts of questions, from the most basic to the most arcane. It is included on TEX Live in `texmf-dist/doc/generic/FAQ-en/`, and is available on the Internet through http://www.tex.ac.uk/faq. Please check here first.

TEX Catalogue If you are looking for a particular package, font, program, etc., the TEX Catalogue is the place to look. It is a huge collection of all TEX-related items. See http://ctan.org/pkg/catalogue/.

TEX Web Resources The web page http://tug.org/interest.html has many TEX-related links, in particular for numerous books, manuals, and articles on all aspects of the system.

support archives Principal support forums include the LATEX community site at http://latex-community.org/, the q&a site http://tex.stackexchange.com, the Usenet newsgroup news:comp.text.tex, and the mailing list texhax@tug.org. Their archives have years of past questions and answers for your searching pleasure, via (for the latter two) http://groups.google.com/group/comp.text.tex/topics and http://tug.org/mail-archives/texhax. And a general web search, for example on http://google.com, never hurts.

asking questions If you cannot find an answer, you can post to http://latex-community.org/ and http://tex.stackexchange.com/ through their web interfaces, to `comp.text.tex` through Google or your newsreader, or to texhax@tug.org through email. But before you post anywhere, please read this FAQ entry, to maximize your chances of getting a useful answer: http://www.tex.ac.uk/cgi-bin/texfaq2html?label=askquestion.

TEX Live support If you want to report a bug or have suggestions or comments on the TEX Live distribution, installation, or documentation, the mailing list is tex-live@tug.org. However, if your question is about how to use a particular program included in TEX Live, please write to that program's maintainer or mailing list. Often running a program with the `--help` option will provide a bug reporting address.

The other side of the coin is helping others who have questions. All the above resources are open to anyone, so feel free to join, start reading, and help out where you can.

2 Overview of TEX Live

This section describes the contents of TEX Live and the TEX Collection of which it is a part.

2.1 The TEX Collection: TEX Live, proTEXt, MacTEX

The TEX Collection DVD comprises the following:

TEX Live A complete TEX system to be installed to disk. Home page: http://tug.org/texlive/.

MacTEX for Mac OS X, this adds a native Mac OS X installer and other Mac applications to TEX Live. Home page: http://tug.org/mactex/.

proTEXt An enhancement of the MiKTEX distribution for Windows, proTEXt adds a few extra tools to MiKTEX, and simplifies installation. It is entirely independent of TEX Live, and has its own installation instructions. Home page: http://tug.org/protext/.

CTAN A snapshot of the CTAN repository (http://www.ctan.org/).

CTAN and protext do not follow the same copying conditions as TEX Live, so be careful when redistributing or modifying.

2.2 Top level TEX Live directories

Here is a brief listing and description of the top level directories in a TEX Live installation.

`bin` The TEX system programs, arranged by platform.

`readme-*.dir` Quick overview and useful links for TEX Live, in various languages, in both HTML and plain text.

`source` The source to all included programs, including the main Web2C-based TEX distributions.

`texmf-dist` The principal tree; see `TEXMFDIST` below.

`tlpkg` Scripts, programs and data for managing the installation, and special support for Windows.

In addition to the directories above, the installation scripts and `README` files (in various languages) are at the top level of the distribution.

For documentation, the comprehensive links in the top-level file `doc.html` may be helpful. The documentation for nearly everything (packages, formats, fonts, program manuals, man pages, Info files) is in `texmf-dist/doc`. You can use the `texdoc` program to find documentation wherever it is located.

This TEX Live documentation itself is in `texmf-dist/doc/texlive`, available in several languages:

- Czech/Slovak: `texmf-dist/doc/texlive/texlive-cz`
- German: `texmf-dist/doc/texlive/texlive-de`
- English: `texmf-dist/doc/texlive/texlive-en`
- French: `texmf-dist/doc/texlive/texlive-fr`
- Italian: `texmf-dist/doc/texlive/texlive-it`
- Polish: `texmf-dist/doc/texlive/texlive-pl`
- Russian: `texmf-dist/doc/texlive/texlive-ru`
- Serbian: `texmf-dist/doc/texlive/texlive-sr`
- Simplified Chinese: `texmf-dist/doc/texlive/texlive-zh-cn`

2.3 Overview of the predefined texmf trees

This section lists the predefined variables specifying the texmf trees used by the system, and their intended purpose, and the default layout of TEX Live. The command `tlmgr conf` shows the values of these variables, so that you can easily find out how they map to particular directories in your installation.

All of the trees, including the personal ones, should follow the TEX Directory Structure (TDS, http://tug.org/tds), with all its myriad subdirectories, or files may not be found. Section 3.4.6 (p. 16) describes this in more detail. The order here is the reverse order in which the trees are searched, that is, later trees in the list override earlier ones.

`TEXMFDIST` The tree which holds nearly all of the files in the original distribution—configuration files, scripts, packages, fonts, etc. (The main exception are the per-platform executables, which are stored in a sibling directory `bin/`.)

`TEXMFSYSVAR` The (site-wide) tree used by `texconfig-sys`, `updmap-sys` and `fmtutil-sys`, and also by `tlmgr`, to store (cached) runtime data such as format files and generated map files.

`TEXMFSYSCONFIG` The (site-wide) tree used by the utilities `texconfig-sys`, `updmap-sys`, and `fmtutil-sys` to store modified configuration data.

`TEXMFLOCAL` The tree which administrators can use for system-wide installation of additional or updated macros, fonts, etc.

`TEXMFHOME` The tree which users can use for their own individual installations of additional or updated macros, fonts, etc. The expansion of this variable dynamically adjusts for each user to their own individual directory.

`TEXMFVAR` The (personal) tree used by `texconfig`, `updmap` and `fmtutil` to store (cached) runtime data such as format files and generated map files.

`TEXMFCONFIG` The (personal) tree used by the utilities `texconfig`, `updmap`, and `fmtutil` to store modified configuration data.

`TEXMFCACHE` The tree(s) used by ConTEXt MkIV and LuaLATEX to store (cached) runtime data; defaults to `TEXMFSYSVAR`, or (if that's not writable), `TEXMFVAR`.

The default layout is:

system-wide root can span multiple TEX Live releases (`/usr/local/texlive` by default on Unix):

2017 A previous release.

2018 The current release.

`bin`

`i386-linux` GNU/Linux binaries

```
        ...
        x86_64-darwin Mac OS X binaries
        win32 Windows binaries
    texmf-dist    TEXMFDIST and TEXMFMAIN
    texmf-var     TEXMFSYSVAR, TEXMFCACHE
    texmf-config TEXMFSYSCONFIG
texmf-local TEXMFLOCAL, intended to be retained from release to release.
```

user's home directory (`$HOME` or `%USERPROFILE%`)

```
.texlive2017 Privately generated and configuration data for a previous release.
.texlive2018 Privately generated and configuration data for the current release.
    texmf-var     TEXMFVAR, TEXMFCACHE
    texmf-config TEXMFCONFIG
texmf TEXMFHOME Personal macros, etc.
```

2.4 Extensions to TEX

Knuth's original TEX itself is frozen, apart from rare bug fixes. It is present in TEX Live as the program tex, and will remain so for the foreseeable future. TEX Live also contains several extended versions of TEX (also known as TEX engines):

ε-TEX adds a set of new primitives (related to macro expansion, character scanning, classes of marks, additional debugging features, and more) and the TEX--XET extensions for bidirectional typesetting. In default mode, ε-TEX is 100% compatible with ordinary TEX. See `texmf-dist/doc/etex/base/etex_man.pdf`.

pdfTEX builds on the ε-TEX extensions, adding support for writing PDF output as well as DVI, and many non-output-related extensions. This is the program invoked for most formats, e.g., etex, latex, pdflatex. Its web site is http://www.pdftex.org/. See `texmf-dist/doc/pdftex/manual/pdftex-a.pdf` for the manual, and `texmf-dist/doc/pdftex/manual/samplepdf/samplepdf.tex` for example usage of some of its features.

LuaTEX is the designated successor of pdfTEX, and is mostly (but not entirely) backward-compatible. It is also intended to be a functional superset of Aleph (see below), though technical compatibility is not intended. The incorporated Lua interpreter (http://www.lua.org/) enables elegant solutions for many thorny TEX problems. When called as `texlua`, it functions as a standalone Lua interpreter, and is already used as such within TEX Live. Its web site is http://www.luatex.org/, and the reference manual is `texmf-dist/doc/luatex/base/luatex.pdf`.

XeTEX adds support for Unicode input and OpenType- and system fonts, implemented using standard third-party libraries. See http://tug.org/xetex.

Ω **(Omega)** is based on Unicode (16-bit characters), thus supports working with almost all the world's scripts simultaneously. It also supports so-called 'Ω Translation Processes' (OTPs), for performing complex transformations on arbitrary input. Omega is no longer included in TEX Live as a separate program; only Aleph is provided:

Aleph combines the Ω and ε-TEX extensions. See `texmf-dist/doc/aleph/base`.

2.5 Other notable programs in TEX Live

Here are a few other commonly-used programs included in TEX Live:

bibtex, biber bibliography support.

makeindex, xindy index support.

dvips convert DVI to PostScript.

xdvi DVI previewer for the X Window System.

dviconcat, dviselect cut and paste pages from DVI files.

dvipdfmx convert DVI to PDF, an alternative approach to pdfTEX (mentioned above).

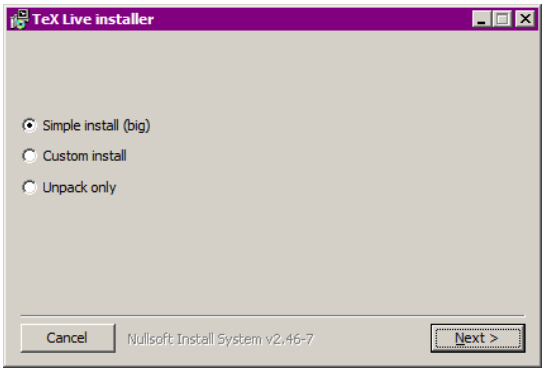


Figure 1: First stage of Windows .exe installer

psselect, psnup, ... PostScript utilities.

pdfjam, pdfjoin, ... PDF utilities.

context, mtxrun ConTEXt and PDF processor.

htlatex, ... tex4ht: (LA)TEX to HTML (and XML and more) converter.

3 Installation

3.1 Starting the installer

To begin, get the TEX Collection DVD or download the TEX Live net installer. See http://tug.org/texlive/acquire.html for more information and other methods of getting the software.

Net installer, .zip or .tar.gz: Download from CTAN, under `systems/texlive/tlnet`; the url http://mirror.ctan.org/systems/texlive/tlnet should redirect to a nearby, up-to-date, mirror. You can retrieve either `install-tl.zip` which can be used under Unix and Windows, or the considerably smaller `install-unx.tar.gz` for Unix only. After unpacking, `install-tl` and `install-tl-windows.bat` will be in the `install-tl` subdirectory.

Net installer, Windows .exe: Download from CTAN as above, and double-click. This starts up a first-stage installer and unpacker; see figure 1. It gives three choices: "Simple install" starts up the wizard installer, "Custom install" the expert GUI installer, as described in section 3.1.3. A third option is just unpacking.

TEX Collection DVD: go to the DVD's `texlive` subdirectory. Under Windows, the installer normally starts automatically when you insert the DVD. You can get the DVD by becoming a member of a TEX user group (highly recommended, http://tug.org/usergroups.html), or purchasing it separately (http://tug.org/store), or burning your own from the ISO image. You can also mount the ISO directly on most systems. After installing from DVD or ISO, if you want to get continuing updates from the Internet, please see 3.4.3.

The same installer program is run, whatever the source. The most visible difference between the two is that with the net installer, what you end up with is the packages that are currently available. This is in contrast to the DVD and ISO images, which are not updated between the major public releases.

If you need to download through proxies, use a `~/.wgetrc` file or environment variables with the proxy settings for Wget (http://www.gnu.org/software/wget/manual/html_node/Proxies.html). TEX Live always uses GNU Wget to download. Of course, this should not matter if you are installing from the DVD or ISO image.

The following sections explain installer start-up in more detail.

3.1.1 Unix

Below, `>` denotes the shell prompt; user input is **bold**. The script `install-tl` is a Perl script. The simplest way to start it on a Unix-compatible system is as follows:

```
> perl /path/to/installer/install-tl
```

(Or you can invoke `/path/to/installer/install-tl` if it stayed executable, or `cd` to the directory first, etc.; we won't repeat all the variations.) You may have to enlarge your terminal window so that it shows the full text installer screen (figure 2).

To install in expert GUI mode (figure 3), you'll need the `Perl::TK` module compiled with XFT support, which is generally the case with GNU/Linux, but often not so with other systems. Given that, you can run:

```
> perl install-tl -gui
```

For a complete listing of the various options:

```
> perl install-tl -help
```

Warning about Unix permissions: Your `umask` at the time of installation will be respected by the TEX Live installer. Therefore, if you want your installation to be usable by users other than you, make sure your setting is sufficiently permissive, for instance, `umask 002`. For more information about `umask`, consult your system documentation.

Special considerations for Cygwin: Unlike other Unix-compatible systems, Cygwin does not by default include all of the prerequisite programs needed by the TEX Live installer. See Section 3.1.4.

3.1.2 Mac OS X

As mentioned in section 2.1, a separate distribution is prepared for Mac OS X, named MacTEX (`http://tug.org/mactex`). We recommend using the native MacTEX installer instead of the TEX Live installer on Mac OS X, because the native installer makes a few Mac-specific adjustments, in particular to allow easily switching between the various TEX distributions for Mac OS X (MacTEX, Fink, MacPorts, ...) using the so-called TEXDist data structure.

MacTEX is firmly based on TEX Live, and the main TEX trees and binaries are precisely the same. It adds a few extra folders with Mac-specific documentation and applications.

3.1.3 Windows

If you are using the unpacked downloaded zip file, or the DVD installer failed to start automatically, double-click `install-tl-windows.bat`. If you want more customization options, e.g., selection of specific package collections, run `install-tl-advanced.bat` instead.

You can also start the installer from the command-prompt. Below, `>` denotes the prompt; user input is **bold**. If you are in the installer directory, run just:

```
> install-tl-windows
```

Or you can invoke it with an absolute location, such as:

```
> D:\texlive\install-tl-windows
```

for the TEX Collection DVD, supposing that `D:` is the optical drive. Figure 4 displays the wizard installer, which is the default for Windows.

To install in text mode, use:

```
> install-tl-windows -no-gui
```

For a complete listing of the various options:

```
> install-tl-windows -help
```

```
Installing TeX Live 2018 from: ...
Platform: x86_64-linux => 'GNU/Linux on x86_64'
Distribution: inst (compressed)
Directory for temporary files: /tmp
...
 Detected platform: GNU/Linux on Intel x86_64

 <B> binary platforms: 1 out of 19

 <S> set installation scheme (scheme-full)

 <C> customizing installation collections
     40 collections out of 41, disk space required: 5328 MB

 <D> directories:
   TEXDIR (the main TeX directory):
     /usr/local/texlive/2018
   ...

 <O> options:
   [ ] use letter size instead of A4 by default
   ...

 <V> set up for portable installation

Actions:
 <I> start installation to hard disk
 <P> save installation profile to 'texlive.profile' and exit
 <H> help
 <Q> quit
```

Figure 2: Main text installer screen (GNU/Linux)

3.1.4 Cygwin

Before beginning the installation, use Cygwin's `setup.exe` program to install the `perl` and `wget` packages if you have not already done so. The following additional packages are recommended:

- `fontconfig` [needed by XeTEX and LuaTEX]
- `ghostscript` [needed by various utilities]
- `libXaw7` [needed by `xdvi`]
- `ncurses` [provides the `clear` command used by the installer]

3.1.5 The text installer

Figure 2 displays the main text mode screen under Unix. The text installer is the default on Unix.

This is only a command-line installer; there is no cursor support at all. For instance, you cannot tab around checkboxes or input fields. You just type something (case-sensitive) at the prompt and press the Enter key, and then the entire terminal screen will be rewritten, with adjusted content.

The text installer interface is this primitive for a reason: it is designed to run on as many platforms as possible, even with a very barebones Perl.

3.1.6 The expert graphical installer

Figure 3 displays the expert graphical installer under GNU/Linux. Other than using buttons and menus, this does not differ much from the text installer.

This mode can be invoked explicitly with

```
> install-tl -gui=perltk
```

3.1.7 The simple wizard installer

Under Windows, the default is to run the simplest installation method we could devise, called the "wizard" installer (figure 4). It installs everything and asks almost no questions. If you want to customize your setup, you should run one of the other installers.

On other platforms, this mode can be invoked explicitly with

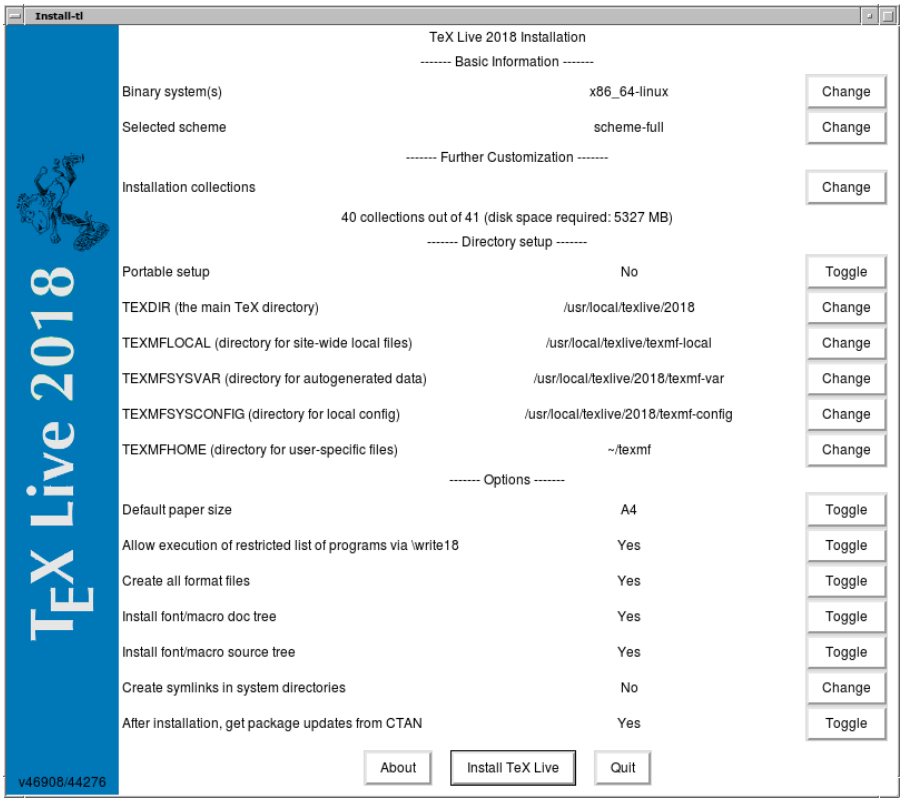


Figure 3: Expert GUI installer screen (GNU/Linux)

```
> install-tl -gui=wizard
```

3.2 Running the installer

The installer is intended to be mostly self-explanatory, but following are a few notes about the various options and submenus.

3.2.1 Binary systems menu (Unix only)

Figure 5 displays the text mode binaries menu. By default, only the binaries for your current platform will be installed. From this menu, you can select installation of binaries for other platforms as well. This can be useful if you are sharing a TEX tree across a network of heterogeneous machines, or for a dual-boot system.

3.2.2 Selecting what is to be installed

Figure 6 displays the TEX Live scheme menu; from here, you choose a “scheme”, which is an overall set of package collections. The default `full` scheme installs everything available. This is recommended, but you can also choose the `basic` scheme for a small system, `minimal` for testing purposes, and `medium` or `teTeX` to get something in between. There are also various specialized and country-specific schemes.

You can refine your scheme selection with the ‘collections’ menu (figure 7, shown in GUI mode for a change).

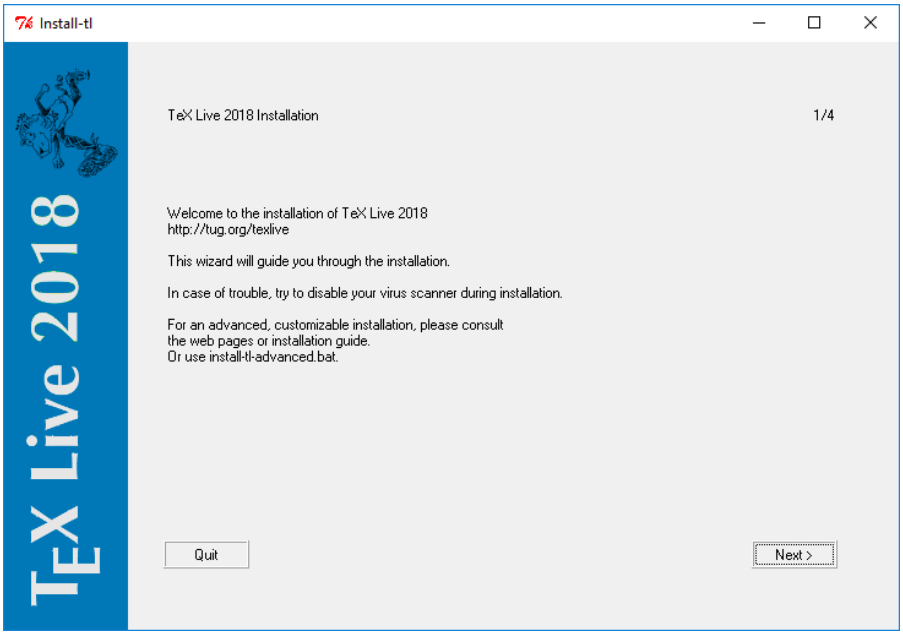


Figure 4: Wizard installer screen (Windows)

```
Available platforms:
================================================================================
   a [ ] Cygwin on Intel x86 (i386-cygwin)
   b [ ] Cygwin on x86_64 (x86_64-cygwin)
   c [ ] MacOSX current (10.10-10.13) on x86_64 (x86_64-darwin)
   d [ ] MacOSX legacy (10.6-10.10) on x86_64 (x86_64-darwinlegacy)
   e [ ] FreeBSD on x86_64 (amd64-freebsd)
   f [ ] FreeBSD on Intel x86 (i386-freebsd)
   g [ ] GNU/Linux on ARM64 (aarch64-linux)
   h [ ] GNU/Linux on ARM (armel-linux)
   i [ ] GNU/Linux on ARMhf (armhf-linux)
   j [ ] GNU/Linux on Intel x86 (i386-linux)
   k [ ] GNU/Linux on PowerPC (powerpc-linux)
   l [X] GNU/Linux on x86_64 (x86_64-linux)
   m [ ] GNU/Linux on x86_64 with musl (x86_64-linuxmusl)
   o [ ] NetBSD on x86_64 (amd64-netbsd)
   p [ ] NetBSD on Intel x86 (i386-netbsd)
   s [ ] Solaris on Intel x86 (i386-solaris)
   t [ ] Solaris on Sparc (sparc-solaris)
   u [ ] Solaris on x86_64 (x86_64-solaris)
   v [ ] Windows (win32)
```

Figure 5: Binaries menu

Collections are one level more detailed than schemes — in essence, a scheme consists of several collections, a collection consists of one or more packages, and a package (the lowest level grouping in TEX Live) contains the actual TEX macro files, font files, and so on.

If you want more control than the collection menus provide, you can use the TEX Live Manager (tlmgr) program after installation (see section 5); using that, you can control the installation at the package level.

```
Select scheme:
===============================================================================
 a [X] full scheme (everything)
 b [ ] medium scheme (small + more packages and languages)
 c [ ] small scheme (basic + xetex, metapost, a few languages)
 d [ ] basic scheme (plain and latex)
 e [ ] minimal scheme (plain only)
 f [ ] ConTeXt scheme
 g [ ] GUST TeX Live scheme
 h [ ] infrastructure-only scheme (no TeX at all)
 i [ ] teTeX scheme (more than medium, but nowhere near full)
 j [ ] custom selection of collections
```

Figure 6: Scheme menu

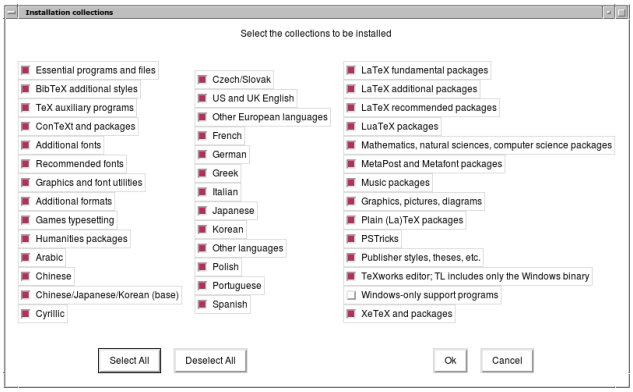


Figure 7: Collections menu

3.2.3 Directories

The default layout is described in section 2.3, p. 5. The default installation directory is `/usr/local/texlive/2018` on Unix and `%SystemDrive%\texlive\2018` on Windows. This arrangement enables having many parallel TEX Live installations, such as one for each release (typically by year, as here), and you can switch between them merely by altering your search path.

That installation directory can be overridden by setting the so-called `TEXDIR` in the installer. The GUI screen for this and other options is shown in figure 3. The most common reasons to change it are either lacking enough disk space in that partition (the full TEX Live needs several gigabytes), or lacking write permission for the default location (you don't have to be root or administrator to install TEX Live, but you do need write access to the target directory).

The installation directories can also be changed by setting a variety of environment variables before running the installer (most likely, `TEXLIVE_INSTALL_PREFIX` or `TEXLIVE_INSTALL_TEXDIR`); see the documentation from `install-tl --help` (available online at http://tug.org/texlive/doc/install-tl.html) for the full list and more details.

A reasonable alternative destination is a directory under your home, especially if you will be the sole user. Use '`~`' to indicate this, as in '`~/texlive/2018`'.

We recommend including the year in the name, to enable keeping different releases of TEX Live side by side. (You may wish to also maintain a version-independent name, such as `/usr/local/texlive-cur`, via a symbolic link, which you can then repoint after testing the new release.)

Changing `TEXDIR` in the installer will also change `TEXMFLOCAL`, `TEXMFSYSVAR` and `TEXMFSYSCONFIG`.

`TEXMFHOME` is the recommended location for personal macro files or packages. The default value is `~/texmf` (`~/Library/texmf` on Macs). In contrast to `TEXDIR`, here a `~` is preserved in the newly-written configuration files, since it usefully refers to the home directory of the user running TEX. It expands to `$HOME` on Unix and `%USERPROFILE%` on Windows. Special redundant note: `TEXMFHOME`, like all trees, must be organized according to the TDS, or files may not be found.

`TEXMFVAR` is the location for storing most cached runtime data specific to each user. `TEXMFCACHE` is

the variable name used for that purpose by LuaLaTeXand ConTeXt MkIV (see section 3.4.5, p. 16); its default value is `TEXMFSYSVAR`, or (if that's not writable), `TEXMFVAR`.

3.2.4 Options

```
Options customization:
================================================================================
 <P> use letter size instead of A4 by default: [ ]
 <E> execution of restricted list of programs: [X]
 <F> create all format files:                  [X]
 <D> install font/macro doc tree:              [X]
 <S> install font/macro source tree:           [X]
 <L> create symlinks in standard directories:  [ ]
           binaries to:
           manpages to:
               info to:
 <Y> after installation, get package updates from CTAN: [X]
```

Figure 8: Options menu (Unix)

Figure 8 shows the text mode options menu. More info on each:

use letter size instead of A4 by default: The default paper size selection. Of course, individual documents can and should specify a specific paper size, if desired.

execution of restricted list of programs: As of TeX Live 2010, execution of a few external programs is allowed by default. The (very short) list of allowed programs is given in the `texmf.cnf`. See the 2010 news (section 9.1.7) for more details.

create format files: Although unnecessary format files take time to generate and disk space to store, it is still recommended to leave this option checked: if you don't, then format files will be generated in people's private `TEXMFVAR` tree as they are needed. In that location, they will not be updated automatically if (say) binaries or hyphenation patterns are updated in the installation, and thus you could end up with incompatible format files.

install font/macro ... tree: Download/install the documentation and source files included in most packages. Unchecking is not recommended.

create symlinks in standard directories: This option (Unix only) bypasses the need to change environment variables. Without this option, TeX Live directories usually have to be added to `PATH`, `MANPATH` and `INFOPATH`. You will need write permissions to the target directories. This option is intended for accessing the TeX system through directories that are already known to users, such as `/usr/local/bin`, which don't already contain any TeX files. Do not overwrite existing files on your system with this option, e.g., by specifying system directories. The safest and recommended approach is to leave the option unchecked.

after installation ... CTAN: When installing from DVD, this option is enabled by default, since usually one wants to take any subsequent package updates from the CTAN area that is updated throughout the year. The only likely reason to disable it is if you install only a subset from the DVD and plan to augment the installation later. In any case, the package repository for the installer, and for updates after installation, can be set independently as needed; see section 3.3.1 and section 3.4.3.

Windows-specific options, as displayed in the advanced Perl/Tk interface:

adjust PATH setting in registry This ensures that all programs will see the TeX Live binary directory on their search path.

add menu shortcuts If set, there will be a TeX Live submenu of the Start menu. There is a third option 'Launcher entry' besides 'TeX Live menu' and 'No shortcuts'. This option is described in section 4.1.

change file associations The options are 'Only new' (create file associations, but do not overwrite existing ones), 'All' and 'None'.

install TeXworks front end

When all the settings are to your liking, you can type 'I' in the text interface, or press the 'Install TeX Live' button in the Perl/Tk GUI, to start the installation process. When it is done, skip to section 3.4 to read what else needs to be done, if anything.

3.3 Command-line install-tl options

Type

```
> install-tl -help
```

for a listing of command-line options. Either `-` or `--` can be used to introduce option names. These are the most common ones:

`-gui` If possible, use the GUI installer. This requires the Perl/Tk module (http://tug.org/texlive/distro.html#perltk) compiled with XFT support; if Perl/Tk is not available, installation continues in text mode.

`-no-gui` Force using the text mode installer, even under Windows.

`-lang` *LL* Specify the installer interface language as a standard (usually two-letter) code. The installer tries to automatically determine the right language but if it fails, or if the right language is not available, then it uses English as a fallback. Run `install-tl --help` to get the list of available languages.

`-portable` Install for portable use on, e.g., a USB stick. Also selectable from within the text installer with the `V` command, and from the GUI installer. See section 4.2.

`-profile` *file* Load the installation profile *file* and do the installation with no user interaction. The installer always writes a file `texlive.profile` to the `tlpkg` subdirectory of your installation. That file can be given as the argument to redo the exact same installation on a different system, for example. Alternatively, you can use a custom profile, most easily created by starting from a generated one and changing values, or an empty file, which will take all the defaults.

`-repository` *url-or-directory* Specify package repository from which to install; see following.

`-in-place` (Documented only for completeness: Do not use this unless you know what you are doing.) If you already have an rsync, svn, or other copy of TeX Live (see http://tug.org/texlive/acquire-mirror.html) then this option will use what you've got, as-is, and do only the necessary post-install. Be warned that the file `tlpkg/texlive.tlpdb` may be overwritten; saving it is your responsibility. Also, package removal has to be done manually. This option cannot be toggled via the installer interface.

3.3.1 The `-repository` option

The default network package repository is a CTAN mirror chosen automatically via http://mirror.ctan.org.

If you want to override that, the location value can be a url starting with `ftp:`, `http:`, or `file:/`, or a plain directory path. (When giving an `http:` or `ftp:` location, trailing '/' characters and/or a trailing '`/tlpkg`' component are ignored.)

For example, you could choose a particular CTAN mirror with something like: http://ctan.example.org/tex-archive/systems/texlive/tlnet/, substituting a real hostname and its particular top-level CTAN path for `ctan.example.org/tex-archive`. The list of CTAN mirrors is maintained at http://ctan.org/mirrors.

If the given argument is local (either a path or a `file:/` url), compressed files in an `archive` subdirectory of the repository path are used (even if uncompressed files are available as well).

3.4 Post-install actions

Some post-installation may be required.

3.4.1 Environment variables for Unix

If you elected to create symlinks in standard directories (described in section 3.2.4), then there is no need to edit environment variables. Otherwise, on Unix systems, the directory of the binaries for your platform must be added to the search path. (On Windows, the installer takes care of this.)

Each supported platform has its own subdirectory under `TEXDIR/bin`. See figure 5 for the list of subdirectories and corresponding platforms.

Optionally, you can also add the documentation man and Info directories to their respective search paths, if you want the system tools to find them. The man pages might be found automatically after the addition to `PATH`.

For Bourne-compatible shells such as `bash`, and using Intel x86 GNU/Linux and a default directory setup as an example, the file to edit might be `$HOME/.profile` (or another file sourced by `.profile`, and the lines to add would look like this:

```
PATH=/usr/local/texlive/2018/bin/i386-linux:$PATH; export PATH
MANPATH=/usr/local/texlive/2018/texmf-dist/doc/man:$MANPATH; export MANPATH
INFOPATH=/usr/local/texlive/2018/texmf-dist/doc/info:$INFOPATH; export INFOPATH
```

For csh or tcsh, the file to edit is typically `$HOME/.cshrc`, and the lines to add might look like:

```
setenv PATH /usr/local/texlive/2018/bin/i386-linux:$PATH
setenv MANPATH /usr/local/texlive/2018/texmf-dist/doc/man:$MANPATH
setenv INFOPATH /usr/local/texlive/2018/texmf-dist/doc/info:$INFOPATH
```

If you already have settings somewhere in your "dot" files, naturally the TEX Live directories should be merged in as appropriate.

3.4.2 Environment variables: Global configuration

If you want to make these changes globally, or for a user newly added to the system, then you are on your own; there is just too much variation between systems in how and where these things are configured.

Our two hints are: 1) you may want to check for a file `/etc/manpath.config` and, if present, add lines such as

```
MANPATH_MAP /usr/local/texlive/2018/bin/i386-linux \
            /usr/local/texlive/2018/texmf-dist/doc/man
```

And 2) check for a file `/etc/environment` which may define the search path and other default environment variables.

In each (Unix) binary directory, we also create a symbolic link named `man` to the directory `texmf-dist/doc/man`. Some `man` programs, such as the standard Mac OS X `man`, will automatically find that, obviating the need for any man page setup.

3.4.3 Internet updates after DVD installation

If you installed TEX Live from DVD and then wish to get updates from the Internet, you need to run this command—*after* you've updated your search path (as described in the previous section):

```
> tlmgr option repository http://mirror.ctan.org/systems/texlive/tlnet
```

This tells `tlmgr` to use a nearby CTAN mirror for future updates. This is done by default when installing from DVD, via the option described in section 3.2.4.

If there are problems with the automatic mirror selection, you can specify a particular CTAN mirror from the list at `http://ctan.org/mirrors`. Use the exact path to the `tlnet` subdir on that mirror, as shown above.

3.4.4 System font configuration for XeTEX and LuaTEX

XeTEX and LuaTEX can use any font installed on the system, not just those in the TEX trees. They do these via related but not identical methods.

On Windows, fonts shipped with TEX Live are automatically made available to XeTEX by font name. On Mac OS X, supporting font name lookups requires additional steps; please see the MacTEX

web pages (http://tug.org/mactex). For other Unix systems, the procedure to be able to find the fonts shipped with TEX Live via font name follows.

To facilitate this, when the xetex package is installed (either at initial installation or later), the necessary configuration file is created in `TEXMFSYSVAR/fonts/conf/texlive-fontconfig.conf`.

To set up the TEX Live fonts for system-wide use (assuming you have suitable privileges), proceed as follows:

1. Copy the `texlive-fontconfig.conf` file to `/etc/fonts/conf.d/09-texlive.conf`.
2. Run `fc-cache -fsv`.

If you do not have sufficient privileges to carry out the steps above, or if you want to make the TEX Live fonts available to only one user, you can do the following:

1. Copy the `texlive-fontconfig.conf` file to `~/.fonts.conf`, where `~` is your home directory.
2. Run `fc-cache -fv`.

You can run `fc-list` to see the names of the system fonts. The incantation `fc-list : family style file spacing` (all arguments are literal strings) shows some generally interesting information.

3.4.5 ConTEXt Mark IV

Both the 'old' ConTEXt (Mark II) and the 'new' ConTEXt (Mark IV) should run out of the box after TEX Live installation, and should need no special attention as long as you stick to using `tlmgr` for updates.

However, because ConTEXt MkIV does not use the kpathsea library, some setup will be required whenever you install new files manually (without using `tlmgr`). After each such installation, each MkIV user must run:

```
context --generate
```

to refresh the ConTEXt disk cache data. The resulting files are stored under `TEXMFCACHE`, whose default value in TEX Live is `TEXMFSYSVAR;TEXMFVAR`.

ConTEXt MkIV will read from all paths mentioned in `TEXMFCACHE`, and write to the first path that is writable. While reading, the last found match will take precedence in the case of duplicated cache data.

For more information, see http://wiki.contextgarden.net/Running_Mark_IV.

3.4.6 Integrating local and personal macros

This is already mentioned implicitly in section 2.3: `TEXMFLOCAL` (by default, `/usr/local/texlive/texmf-local` or `%SystemDrive%\texlive\texmf-local` on Windows) is intended for system-wide local fonts and macros; and `TEXMFHOME` (by default, `$HOME/texmf` or `%USERPROFILE%\texmf`), is for personal fonts and macros. These directories are intended to stick around from release to release, and have their content seen automatically by a new TEX Live release. Therefore, it is best to refrain from changing the definition of `TEXMFLOCAL` to be too far away from the main TEX Live directory, or you will need to manually change future releases.

For both trees, files should be placed in their proper TEX Directory Structure (TDS) subdirectories; see http://tug.org/tds or consult `texmf-dist/web2c/texmf.cnf`. For instance, a LATEX class file or package should be placed in `TEXMFLOCAL/tex/latex` or `TEXMFHOME/tex/latex`, or a subdirectory thereof.

`TEXMFLOCAL` requires an up-to-date filename database, or files will not be found. You can update it with the command mktexlsr or use the 'Reinit file database' button on the configuration tab of the TEX Live Manager GUI.

By default, each of these variables is defined to be a single directory, as shown. This is not a hard-and-fast requirement. If you need to easily switch back and forth between different versions of large packages, for example, you can maintain multiple trees for your own purposes. This is done by setting `TEXMFHOME` to the list of directories, within braces, separated by commas:

```
TEXMFHOME = {/my/dir1,/mydir2,/a/third/dir}
```

Section 7.1.5 describes brace expansion further.

3.4.7 Integrating third-party fonts

This is unfortunately a messy topic. Forget about it unless you want to delve into many details of the TEX installation. Many fonts are included in TEX Live already, so take a look if you don't already know that what you want isn't there.

A possible alternative is to use XeTEX or LuaTEX (see section 2.4), which let you use operating system fonts without any installation in TEX.

If you do need to do this, see http://tug.org/fonts/fontinstall.html for our best effort at describing the procedure.

3.5 Testing the installation

After installing TEX Live, you naturally want to test it out, so you can start creating beautiful documents and/or fonts.

One thing you may immediately be looking for is a front-end with which to edit files. TEX Live installs TEXworks (http://tug.org/texworks) on Windows (only), and MacTEX installs TeXShop (http://pages.uoregon.edu/koch/texshop. On other Unix systems, it's left up to you to choose an editor. There are many choices available, some of which are listed in the next section; see also http://tug.org/interest.html#editors. Any plain text editor will work; something TEX-specific is not required.

The rest of this section gives some basic procedures for testing that the new system is functional. We give Unix commands here; under Mac OS X and Windows, you're more likely to run the tests through a graphical interface, but the principles are the same.

1. Make sure that you can run the tex program in the first place:

   ```
   > tex --version
   TeX 3.14159265 (TeX Live ...)
   Copyright ... D.E. Knuth.
   ...
   ```

   If this comes back with 'command not found' instead of version and copyright information, or with an older version, most likely you don't have the correct bin subdirectory in your PATH. See the environment-setting information on p. 15.

2. Process a basic LATEX file:

   ```
   > latex sample2e.tex
   This is pdfTeX 3.14...
   ...
   Output written on sample2e.dvi (3 pages, 7484 bytes).
   Transcript written on sample2e.log.
   ```

   If this fails to find sample2e.tex or other files, most likely you have interference from old environment variables or configuration files; we recommend unsetting all TEX-related environment variables for a start. (For a deep analysis, you can ask TEX to report on exactly what it is searching for, and finding; see "Debugging actions" on page 30.)

3. Preview the result online:

   ```
   > xdvi sample2e.dvi    # Unix
   > dviout sample2e.dvi  # Windows
   ```

   You should see a new window with a nice document explaining some of the basics of LATEX. (Well worth reading, by the way, if you're new to TEX.) You do have to be running under X for xdvi to work; if you're not, or your DISPLAY environment variable is set incorrectly, you'll get an error 'Can't open display'.

4. Create a PostScript file for printing or display:

   ```
   > dvips sample2e.dvi -o sample2e.ps
   ```

5. Create a PDF file instead of DVI; this processes the `.tex` file and writes PDF directly:

   ```
   > pdflatex sample2e.tex
   ```

6. Preview the PDF file:

   ```
   > gv sample2e.pdf
   ```
   or:
   ```
   > xpdf sample2e.pdf
   ```

   Neither gv nor xpdf are included in TeX Live, so you must install them separately. See http://www.gnu.org/software/gv and http://www.foolabs.com/xpdf, respectively. There are plenty of other PDF viewers, too. For Windows, we recommend trying Sumatra PDF (https://www.sumatrapdfreader.org/free-pdf-reader.html).

7. Standard test files you may find useful in addition to `sample2e.tex`:

   `small2e.tex` A simpler document than `sample2e`, to reduce the input size if you're having troubles.

   `testpage.tex` Test if your printer introduces any offsets.

   `nfssfont.tex` For printing font tables and tests.

   `testfont.tex` Also for font tables, but using plain TeX.

   `story.tex` The most canonical (plain) TeX test file of all. You must type '`\bye`' to the `*` prompt after '`tex story.tex`'.

8. If you have installed the `xetex` package, you can test its access to system fonts as follows:

   ```
   > xetex opentype-info.tex
   This is XeTeX, Version 3.14...
   ...
   Output written on opentype-info.pdf (1 page).
   Transcript written on opentype-info.log.
   ```

   If you get an error message saying "Invalid fontname 'Latin Modern Roman/ICU'...", then you need to configure your system so that the fonts shipped with TeX Live can be found. See Section 3.4.4.

3.6 Links for additional downloadable software

If you are new to TeX, or otherwise need help with actually writing TeX or LaTeX documents, please visit http://tug.org/begin.html for some introductory resources.

Links for some other tools you may consider installing:

Ghostscript https://ghostscript.com/

Perl http://www.perl.org/ with supplementary packages from CPAN, http://www.cpan.org/

ImageMagick http://www.imagemagick.com, for graphics processing and conversion

NetPBM http://netpbm.sourceforge.net/, also for graphics.

TeX-oriented editors There is a wide choice, and it is a matter of the user's taste. Here is a selection in alphabetical order (a few here are for Windows only).

- GNU Emacs is also available natively under Windows, see http://www.gnu.org/software/emacs/emacs.html.
- Emacs with AucTeX for Windows is available from CTAN. The AuCTeX home page is http://www.gnu.org/software/auctex.
- SciTE is available from http://www.scintilla.org/SciTE.html.
- Texmaker is free software, available from http://www.xm1math.net/texmaker.
- TeXstudio started out as a fork of Texmaker with additional features; http://texstudio.org/.

- TeXnicCenter is free software, available from http://www.texniccenter.org and in the proTEXt distribution.
- TeXworks is free software, available from http://tug.org/texworks and installed as part of TEX Live for Windows (only).
- Vim is free software, available from http://www.vim.org.
- WinEdt is shareware available though http://tug.org/winedt or http://www.winedt.com.
- WinShell is available from http://www.winshell.de.

For a much longer list of packages and programs, see http://tug.org/interest.html.

4 Specialized installations

The previous sections described the basic installation process. Here we turn to some specialized cases.

4.1 Shared-user (or cross-machine) installations

TEX Live has been designed to be shared between different systems on a network. With a standard directory layout, no hard paths are configured: the locations for files needed by TEX Live programs are found relative to the programs. You can see this in the principal configuration file `$TEXMFDIST/web2c/texmf.cnf`, which contains lines such as

```
TEXMFROOT = $SELFAUTOPARENT
...
TEXMFDIST = $TEXMFROOT/texmf-dist
...
TEXMFLOCAL = $SELFAUTOGRANDPARENT/texmf-local
```

This means that adding the directory for TEX Live executables for their platform to their search path is sufficient to get a working setup.

By the same token, you can also install TEX Live locally and then move the entire hierarchy afterwards to a network location.

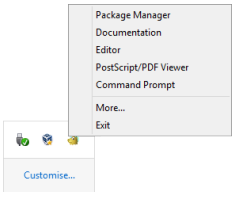
For Windows, TEX Live includes a launcher `tlaunch`. Its main window contains menu entries and buttons for various TEX-related programs and documentation, customizable via an `ini` file. On first use, it replicates the usual Windows-specific post-install, *i.e.*, search path modification and file associations, but only for the current user. Therefore, workstations with access to TEX Live on the local network only need a menu shortcut for the launcher. See the `tlaunch` manual (`texdoc tlaunch`, or https://ctan.org/pkg/tlaunch).

4.2 Portable (USB) installations

The `-portable` installer option (or `V` command in the text installer or corresponding GUI option) creates a completely self-contained TEX Live installation under a common root and forgoes system integration. You can create such an installation directly on a USB stick, or copy it to a USB stick afterwards.

To run TEX using this portable installation, you need to add the appropriate binary directory to the search path during your terminal session, as usual.

On Windows, you can double-click `tl-tray-menu` at the root of the installation and create a temporary 'tray menu' offering a choice of a few common tasks, as shown in this screenshot:



The 'More...' entry explains how you can customize this menu.

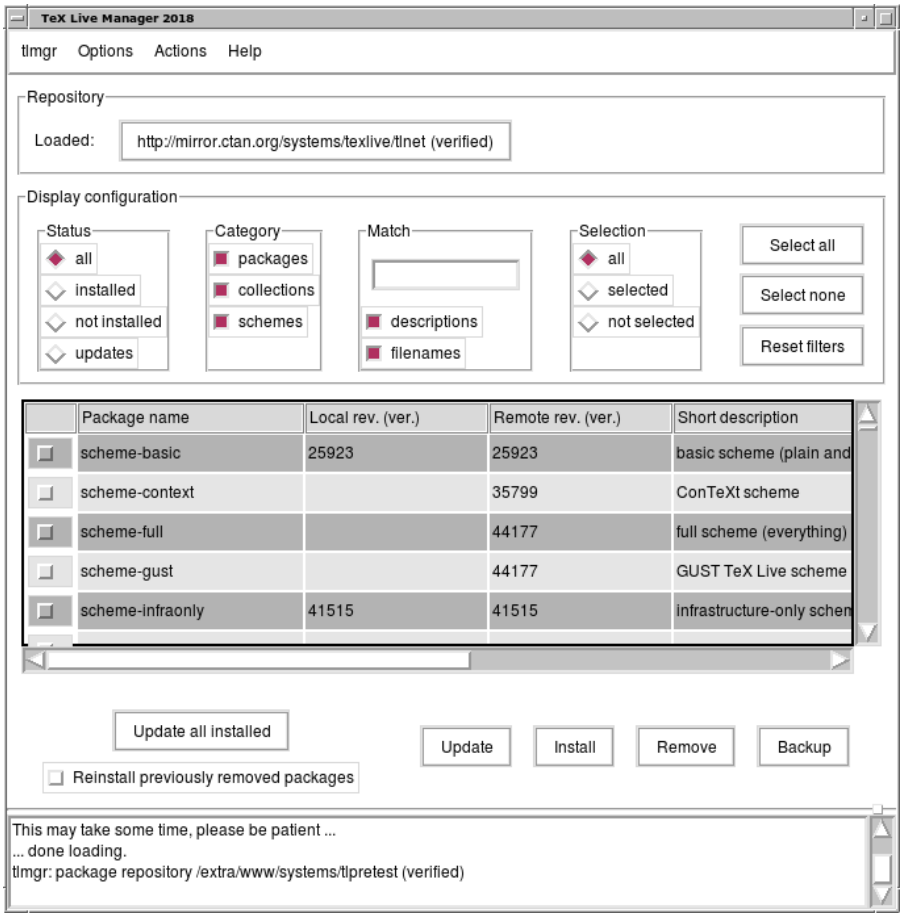


Figure 9: tlmgr in GUI mode: main window, after 'Load'.

5 tlmgr: Managing your installation

TEX Live includes a program named tlmgr for managing TEX Live after the initial installation. Its capabilities include:

- installing, updating, backing up, restoring, and uninstalling individual packages, optionally taking dependencies into account;
- searching for and listing packages and their descriptions;
- listing, adding, and removing platforms;
- changing installation options such as paper size and source location (see section 3.3.1).

tlmgr's functionality completely subsumes the texconfig program. We still distribute and maintain texconfig for the sake of anyone used to its interface, but we recommend using tlmgr nowadays.

5.1 tlmgr GUI mode

tlmgr can be started in GUI mode (figure 9) with

```
> tlmgr -gui
```

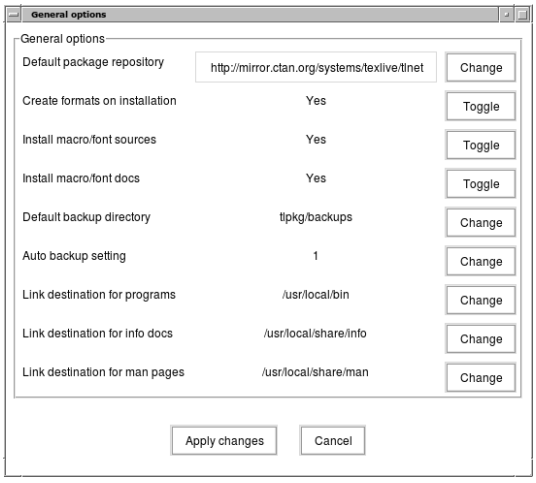


Figure 10: `tlmgr` in GUI mode: General options

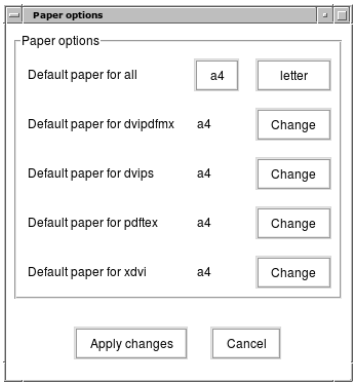


Figure 11: `tlmgr` in GUI mode: Paper size options

or in Windows via the Start menu: `Start`, `Programs`, `TeX Live ...`, `TeX Live Manager`. After clicking 'Load' it displays a list of available and installed packages. This assumes of course that the installation source is valid and reachable.

Figures 10 and 11 show the general and paper size option screens.

5.2 Other GUI interfaces for tlmgr

Besides the `tlmgr -gui` mode described just above, two other GUI programs use tlmgr as a backend: tlshell (written in Tcl/Tk) and tlcockpit (written in Java). They are included as separate packages.

5.3 Sample tlmgr command-line invocations

After the initial installation, you can update your system to the latest versions available with:

```
> tlmgr update -all
```

If this makes you nervous, first try

```
> tlmgr update -all -dry-run
```

or (less verbose):

```
> tlmgr update -list
```

This more complex example adds a collection, for the engine XeTEX, from a local directory:

```
> tlmgr -repository /local/mirror/tlnet install collection-xetex
```

It generates the following output (abridged):

```
install: collection-xetex
install: arabxetex
...
install: xetex
install: xetexconfig
install: xetex.i386-linux
running post install action for xetex
install: xetex-def
...
running mktexlsr
mktexlsr: Updating /usr/local/texlive/2018/texmf-dist/ls-R...
...
```

```
running fmtutil-sys --missing
...
Transcript written on xelatex.log.
fmtutil: /usr/local/texlive/2018/texmf-var/web2c/xetex/xelatex.fmt installed.
```

As you can see, tlmgr installs dependencies, and takes care of any necessary post-install actions, including updating the filename database and (re)generating formats. In the above, we generated new formats for XeTEX.

To describe a package (or collection or scheme):

```
> tlmgr show collection-latexextra
```

which produces output like this:

```
package:    collection-latexextra
category:   Collection
shortdesc:  LaTeX supplementary packages
longdesc:   A very large collection of add-on packages for LaTeX.
installed:  Yes
revision:   46963
sizes:      657941k
```

Last and most important, for full documentation see http://tug.org/texlive/tlmgr.html, or:

```
> tlmgr -help
```

6 Notes on Windows

6.1 Windows-specific features

Under Windows, the installer does some extra things:

Menus and shortcuts. A new 'TEX Live' submenu of the Start menu is installed, which contains entries for some GUI programs (tlmgr, texdoctk, the PS_View (psv) PostScript previewer) and some documentation.

File associations. If enabled, TeXworks, Dviout and PS_view become either the default program for their respective filetypes, or get an entry in the 'Open with' right-click menus of those filetypes.

Bitmap to eps converter. Various bitmapped formats get an entry bitmap2eps in their 'Open with' right-click menu. Bitmap2eps is a simple script which lets sam2p or bmeps do the real work.

Automatic path adjustment. No manual configuration steps are required.

Uninstaller. The installer creates an entry under 'Add/Remove Programs' for TEX Live. The uninstall tab of the TEX Live Manager GUI refers to this. For a single-user install, the installer also creates an uninstall entry under the Start menu.

Write-protect. For an admin install, the TEX Live directories are write-protected, at least if TEX Live is installed on a normal NTFS-formatted non-removable disk.

Also, have a look at `tlaunch`, described in section 4.1, for a different approach.

6.2 Additional software included on Windows

To be complete, a TEX Live installation needs support packages that are not commonly found on a Windows machine. TEX Live provides the missing pieces. These programs are all installed as part of TEX Live only on Windows.

Perl and Ghostscript. Because of the importance of Perl and Ghostscript, TEX Live includes 'hidden' copies of these programs. TEX Live programs that need them know where to find them, but they don't betray their presence through environment variables or registry settings. They aren't full-scale installations, and shouldn't interfere with any system installations of Perl or Ghostscript.

PS_View. Also installed is PS_View, a PostScript and PDF viewer; see figure 12.

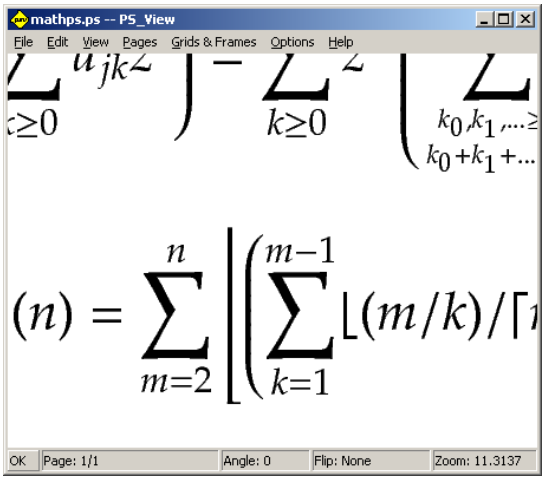


Figure 12: PS_View: very high magnifications available!

dviout. Also installed is dviout, a DVI viewer. At first, when you preview files with dviout, it will create fonts, because screen fonts were not installed. After a while, you will have created most of the fonts you use, and you will rarely see the font-creation window. More information can be found in the (highly recommended) on-line help.

TEXworks. TEXworks is a TEX-oriented editor with an integrated PDF viewer.

Command-line tools. A number of Windows ports of common Unix command-line programs are installed along with the usual TEX Live binaries. These include gzip, zip, unzip, and the utilities from the xpdf suite (pdfinfo, pdffonts, ...). The xpdf viewer itself is not available for Windows. One alternative among many is the Sumatra PDF viewer, available from https://sumatrapdfreader.org/.

fc-list, fc-cache, ... The tools from the fontconfig library allow XeTEX to handle system fonts on Windows. You can use fc-list to determine the font names to pass to XeTEX's extended `\font` command. If necessary, run fc-cache first to update font information.

6.3 User Profile is Home

The Windows counterpart of a Unix home directory is the `%USERPROFILE%` directory. Under Windows Vista and later it is `C:\Users\<username>`. In the `texmf.cnf` file, and Kpathsea in general, `~` will expand appropriately on both Windows and Unix.

6.4 The Windows registry

Windows stores nearly all configuration data in its registry. The registry contains a set of hierarchically organized keys, with several root keys. The most important ones for installation programs are `HKEY_CURRENT_USER` and `HKEY_LOCAL_MACHINE`, `HKCU` and `HKLM` in short. The `HKCU` part of the registry is in the user's home directory (see section 6.3). `HKLM` is normally in a subdirectory of the Windows directory.

In some cases, system information could be obtained from environment variables but for other information, for example the location of shortcuts, it is necessary to consult the registry. Setting environment variables permanently also requires registry access.

6.5 Windows permissions

In later versions of Windows, a distinction is made between regular users and administrators, where only the latter have free access to the entire operating system. We have made an effort to make TEX Live installable without administrative privileges.

If the installer is started with administrative permissions, there is an option to install for all users. If this option is chosen, shortcuts are created for all users, and the system search path is modified. Otherwise, shortcuts and menu entries are created for the current user, and the user search path is modified.

Regardless of administrator status, the default root of TeX Live proposed by the installer is always under `%SystemDrive%`. The installer always tests whether the root is writable for the current user.

A problem may arise if the user is not an administrator and TeX already exists in the search path. Since the effective search path consists of the system search path followed by the user search path, the new TeX Live would never get precedence. As a safeguard, the installer creates a shortcut to the command-prompt in which the new TeX Live binary directory is prepended to the local search path. The new TeX Live will be always usable from within such a command-prompt. The shortcut for TeXworks, if installed, also prepends TeX Live to the search path, so it should also be immune to this path problem.

You should be aware that even if you are logged in as administrator, you need to explicitly ask for administrator privileges. In fact, there is not much point in logging in as administrator. Instead, right-clicking on the program or shortcut that you want to run usually gives you a choice 'Run as administrator'.

6.6 Increasing maximum memory on Windows and Cygwin

Windows and Cygwin (see section 3.1.4 for Cygwin installation specifics) users may find that they run out of memory when running some of the programs shipped with TeX Live. For example, asy might run out of memory if you try to allocate an array of 25,000,000 reals, and LuaTeX might run out of memory if you try to process a document with a lot of big fonts.

For Cygwin, you can increase the amount of available memory by following the instructions in the Cygwin User's Guide (http://www.cygwin.com/cygwin-ug-net/setup-maxmem.html).

For Windows, you have to create a file, say `moremem.reg`, with these four lines:

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\Software\Cygwin]
"heap_chunk_in_mb"=dword:ffffff00
```

and then execute the command `regedit /s moremem.reg` as administrator. (If you want to change memory only for the current user instead of system-wide, use `HKEY_CURRENT_USER`.)

7 A user's guide to Web2C

Web2C is an integrated collection of TeX-related programs: TeX itself, METAFONT, MetaPost, BibTeX, etc. It is the heart of TeX Live. The home page for Web2C, with the current manual and more, is http://tug.org/web2c.

A bit of history: The original implementation was by Tomas Rokicki who, in 1987, developed a first TeX-to-C system based on change files under Unix, which were primarily the original work of Howard Trickey and Pavel Curtis. Tim Morgan became the maintainer of the system, and during this period the name changed to Web-to-C. In 1990, Karl Berry took over the work, assisted by dozens of additional contributors, and in 1997 he handed the baton to Olaf Weber, who returned it to Karl in 2006.

The Web2C system runs on Unix, 32-bit Windows systems, Mac OS X, and other operating systems. It uses Knuth's original sources for TeX and other basic programs written in the `WEB` literate programming system and translates them into C source code. The core TeX programs handled in this way are:

bibtex Maintaining bibliographies.

dvicopy Expands virtual font references in DVI files.

dvitomp DVI to MPX (MetaPost pictures).

dvitype DVI to human-readable text.

gftodvi Generic font proofsheets.

gftopk Generic to packed fonts.

gftype GF to human-readable text.

mf Creating typeface families.

mft Prettyprinting METAFONT source.

mpost Creating technical diagrams.

patgen Creating hyphenation patterns.

pktogf Packed to generic fonts.

pktype PK to human-readable text.

pltotf Plain text property list to TFM.

pooltype Display WEB pool files.

tangle WEB to Pascal.

tex Typesetting.

tftopl TFM to plain text property list.

vftovp Virtual font to virtual property list.

vptovf Virtual property list to virtual font.

weave WEB to TEX.

The precise functions and syntax of these programs are described in the documentation of the individual packages and of Web2C itself. However, knowing a few principles governing the whole family of programs will help you take advantage of your Web2C installation.

All programs honor these standard GNU options:

`--help` print basic usage summary.

`--verbose` print detailed progress report.

`--version` print version information, then exit.

For locating files the Web2C programs use the path searching library Kpathsea (http://tug.org/kpathsea). This library uses a combination of environment variables and configuration files to optimize searching the (huge) collection of TEX files. Web2C can look at many directory trees simultaneously, which is useful in maintaining TEX's standard distribution and local and personal extensions in distinct trees. To speed up file searches, the root of each tree has a file `ls-R`, containing an entry showing the name and relative pathname for all files under that root.

7.1 Kpathsea path searching

Let us first describe the generic path searching mechanism of the Kpathsea library.

We call a *search path* a colon- or semicolon-separated list of *path elements*, which are basically directory names. A search path can come from (a combination of) many sources. To look up a file '`my-file`' along a path '`.:/dir`', Kpathsea checks each element of the path in turn: first `./my-file`, then `/dir/my-file`, returning the first match (or possibly all matches).

In order to adapt optimally to all operating systems' conventions, on non-Unix systems Kpathsea can use filename separators different from colon ('`:`') and slash ('`/`').

To check a particular path element p, Kpathsea first checks if a prebuilt database (see "Filename database" on page 28) applies to p, i.e., if the database is in a directory that is a prefix of p. If so, the path specification is matched against the contents of the database.

Although the simplest and most common path element is a directory name, Kpathsea supports additional features in search paths: layered default values, environment variable names, config file values, users' home directories, and recursive subdirectory searching. Thus, we say that Kpathsea *expands* a path element, meaning it transforms all the specifications into basic directory name or names. This is described in the following sections in the same order as it takes place.

Note that if the filename being searched for is absolute or explicitly relative, i.e., starts with '`/`' or '`./`' or '`../`', Kpathsea simply checks if that file exists.

7.1.1 Path sources

A search path can come from many sources. In the order in which Kpathsea uses them:

1. A user-set environment variable, for instance, `TEXINPUTS`. Environment variables with a period and a program name appended override; e.g., if '`latex`' is the name of the program being run, then `TEXINPUTS.latex` will override `TEXINPUTS`.
2. A program-specific configuration file, for example, a line '`S /a:/b`' in `dvips`'s `config.ps`.
3. A Kpathsea configuration file `texmf.cnf`, containing a line like '`TEXINPUTS=/c:/d`' (see below).
4. The compile-time default.

You can see each of these values for a given search path by using the debugging options (see "Debugging actions" on page 30).

7.1.2 Config files

Kpathsea reads *runtime configuration files* named `texmf.cnf` for search path and other definitions. The search path used to look for these files is named `TEXMFCNF`, but we do not recommend setting this (or any) environment variable.

Instead, normal installation results in a file `.../2018/texmf.cnf`. If you must make changes to the defaults (not normally necessary), this is the place to put them. The main configuration file is in `.../2018/texmf-dist/web2c/texmf.cnf`. You should not edit this latter file, as your changes will be lost when the distributed version is updated.

All `texmf.cnf` files in the search path will be read and definitions in earlier files override those in later files. For example, with a search path of `.:$TEXMF`, values from `./texmf.cnf` override those from `$TEXMF/texmf.cnf`.

- Comments start with `%` and continue to the end of the line.
- Blank lines are ignored.
- A \ at the end of a line acts as a continuation character, i.e., the next line is appended. Whitespace at the beginning of continuation lines is not ignored.
- Each remaining line has the form:

  variable`[.`*progname*`] [=]` *value*

  where the '`=`' and surrounding whitespace are optional.
- The *variable* name may contain any character other than whitespace, '`=`', or '`.`', but sticking to '`A-Za-z_`' is safest.
- If '`.`*progname*' is present, the definition only applies if the program that is running is named *progname* or *progname*`.exe`. This allows different flavors of TEX to have different search paths, for example.
- *value* may contain any characters except `%` and '`@`'. The `$`*var*`.`*prog* feature is not available on the right-hand side; instead, you must use an additional variable. A '`;`' in *value* is translated to '`:`' if running under Unix; this is useful to be able to have a single `texmf.cnf` for Unix, MS-DOS and Windows systems.
- All definitions are read before anything is expanded, so variables can be referenced before they are defined.

A configuration file fragment illustrating most of these points is shown below:

```
TEXMF              = {$TEXMFLOCAL,!!$TEXMFMAIN}
TEXINPUTS.latex    = .;$TEXMF/tex/{latex,generic;}//
TEXINPUTS.fontinst = .;$TEXMF/tex//;$TEXMF/fonts/afm//
% e-TeX related files
TEXINPUTS.elatex   = .;$TEXMF/{etex,tex}/{latex,generic;}//
TEXINPUTS.etex     = .;$TEXMF/{etex,tex}/{eplain,plain,generic;}//
```

7.1.3 Path expansion

Kpathsea recognizes certain special characters and constructions in search paths, similar to those available in Unix shells. As a general example, the complex path, `~$USER/{foo,bar}//baz`, expands to all subdirectories under directories `foo` and `bar` in `$USER`'s home directory that contain a directory or file `baz`. These expansions are explained in the sections below.

7.1.4 Default expansion

If the highest-priority search path (see "Path sources" on page 26) contains an *extra colon* (i.e., leading, trailing, or doubled), Kpathsea inserts at that point the next-highest-priority search path that is defined. If that inserted path has an extra colon, the same happens with the next highest. For example, given an environment variable setting

```
> setenv TEXINPUTS /home/karl:
```

and a `TEXINPUTS` value from `texmf.cnf` of

```
.:$TEXMF//tex
```

then the final value used for searching will be:

```
/home/karl:.:$TEXMF//tex
```

Since it would be useless to insert the default value in more than one place, Kpathsea changes only one extra '`:`' and leaves any others in place. It checks first for a leading '`:`', then a trailing '`:`', then a doubled '`:`'.

7.1.5 Brace expansion

A useful feature is brace expansion, which means that, for instance, `v{a,b}w` expands to `vaw:vbw`. Nesting is allowed. This is used to implement multiple TEX hierarchies, by assigning a brace list to `$TEXMF`. For example, in `texmf.cnf`, a definition like this (simplified for this example) is made:

```
TEXMF = {$TEXMFVAR,$TEXMFHOME,!!$TEXMFLOCAL,!!$TEXMFDIST}
```

We can then use this to define, for example, the TEX input path:

```
TEXINPUTS = .;$TEXMF/tex//
```

which means that, after looking in the current directory, the `$TEXMFVAR/tex`, `$TEXMFHOME/tex`, `$TEXMFLOCAL/tex` and `$TEXMFDIST/tex` trees *only*) will be searched (the last two using `ls-R` data base files). It is a convenient way for running two parallel TEX structures, one "frozen" (on a CD, for instance) and the other being continuously updated with new versions as they become available. By using the `$TEXMF` variable in all definitions, one is sure to always search the up-to-date tree first.

7.1.6 Subdirectory expansion

Two or more consecutive slashes in a path element following a directory *d* is replaced by all subdirectories of *d*: first those subdirectories directly under *d*, then the subsubdirectories under those, and so on. At each level, the order in which the directories are searched is *unspecified*.

If you specify any filename components after the '`//`', only subdirectories with matching components are included. For example, '`/a//b`' expands into directories `/a/1/b`, `/a/2/b`, `/a/1/1/b`, and so on, but not `/a/b/c` or `/a/1`.

Multiple '`//`' constructs in a path are possible, but '`//`' at the beginning of a path is ignored.

7.1.7 List of special characters and their meaning: a summary

The following list summarizes the special characters in Kpathsea configuration files.

`:` Separator in path specification; at the beginning or the end of a path it substitutes the default path expansion.

`;` Separator on non-Unix systems (acts like `:`).

`$` Variable expansion.

`~` Represents the user's home directory.

`{...}` Brace expansion.

`//` Subdirectory expansion (can occur anywhere in a path, except at its beginning).

`%` Start of comment.

`\` Continuation character (allows multi-line entries).

`!!` Search *only* database to locate file, *do not* search the disk.

7.2 Filename databases

Kpathsea goes to some lengths to minimize disk accesses for searches. Nevertheless, at installations with enough directories, searching each possible directory for a given file can take an excessively long time (this is especially true if many hundreds of font directories have to be traversed.) Therefore, Kpathsea can use an externally-built plain text "database" file named `ls-R` that maps files to directories, thus avoiding the need to exhaustively search the disk.

A second database file `aliases` allows you to give additional names to the files listed in `ls-R`. This can be helpful to confirm to DOS 8.3 filename conventions in source files.

7.2.1 The filename database

As explained above, the name of the main filename database must be `ls-R`. You can put one at the root of each TEX hierarchy in your installation that you wish to be searched (`$TEXMF` by default). Kpathsea looks for `ls-R` files along the `TEXMFDBS` path.

The recommended way to create and maintain '`ls-R`' is to run the `mktexlsr` script included with the distribution. It is invoked by the various '`mktex`'... scripts. In principle, this script just runs the command

`cd` */your/texmf/root* `&& \ls -1LAR ./ >ls-R`

presuming your system's `ls` produces the right output format (GNU `ls` is all right). To ensure that the database is always up-to-date, it is easiest to rebuild it regularly via `cron`, so that it is automatically updated when the installed files change, such as after installing or updating a LATEX package.

If a file is not found in the database, by default Kpathsea goes ahead and searches the disk. If a particular path element begins with '`!!`', however, *only* the database will be searched for that element, never the disk.

7.2.2 kpsewhich: Standalone path searching

The `kpsewhich` program exercises path searching independent of any particular application. This can be useful as a sort of `find` program to locate files in TEX hierarchies (this is used heavily in the distributed '`mktex`'... scripts).

`> kpsewhich` *option*`...` *filename*`...`

The options specified in *option* start with either '`-`' or '`--`', and any unambiguous abbreviation is accepted.

Kpathsea looks up each non-option argument on the command line as a filename, and returns the first file found. There is no option to return all the files with a particular name (you can run the Unix '`find`' utility for that).

The most common options are described next.

`--dpi=`*num* Set the resolution to *num*; this only affects '`gf`' and '`pk`' lookups. '`-D`' is a synonym, for compatibility with `dvips`. Default is 600.

`--format=`*name*
Set the format for lookup to *name*. By default, the format is guessed from the filename. For formats which do not have an associated unambiguous suffix, such as MetaPost support files and `dvips` configuration files, you have to specify the name as known to Kpathsea, such as `tex` or `enc files`. Run `kpsewhich --help` for a list.

`--mode=`*string*
Set the mode name to *string*; this only affects '`gf`' and '`pk`' lookups. No default: any mode will be found.

`--must-exist`
Do everything possible to find the files, notably including searching the disk. By default, only the `ls-R` database is checked, in the interest of efficiency.

`--path=`*string*
Search along the path *string* (colon-separated as usual), instead of guessing the search path from the filename. '`//`' and all the usual expansions are supported. The options '`--path`' and '`--format`' are mutually exclusive.

`--progname=`*name*
: Set the program name to *name*. This can affect the search paths via the *.progname* feature. The default is kpsewhich.

`--show-path=`*name*
: shows the path used for file lookups of file type *name*. Either a filename extension (`.pk`, `.vf`, etc.) or a name can be used, just as with '`--format`' option.

`--debug=`*num*
: sets the debugging options to *num*.

7.2.3 Examples of use

Let us now have a look at Kpathsea in action. Here's a straightforward search:

```
> kpsewhich article.cls
   /usr/local/texmf-dist/tex/latex/base/article.cls
```

We are looking for the file `article.cls`. Since the '`.cls`' suffix is unambiguous we do not need to specify that we want to look for a file of type `tex` (TEX source file directories). We find it in the subdirectory `tex/latex/base` below the '`texmf-dist`' TEX Live directory. Similarly, all of the following are found without problems thanks to their unambiguous suffix.

```
> kpsewhich array.sty
   /usr/local/texmf-dist/tex/latex/tools/array.sty
> kpsewhich latin1.def
   /usr/local/texmf-dist/tex/latex/base/latin1.def
> kpsewhich size10.clo
   /usr/local/texmf-dist/tex/latex/base/size10.clo
> kpsewhich small2e.tex
   /usr/local/texmf-dist/tex/latex/base/small2e.tex
> kpsewhich tugboat.bib
   /usr/local/texmf-dist/bibtex/bib/beebe/tugboat.bib
```

By the way, that last is a BIBTEX bibliography database for *TUGboat* articles.

```
> kpsewhich cmr10.pk
```

Font bitmap glyph files of type `.pk` are used by display programs like dvips and xdvi. Nothing is returned in this case since there are no pre-generated Computer Modern '`.pk`' files in TEX Live — the Type 1 variants are used by default.

```
> kpsewhich wsuipa10.pk
   /usr/local/texmf-var/fonts/pk/ljfour/public/wsuipa/wsuipa10.600pk
```

For these fonts (a phonetic alphabet from the University of Washington) we had to generate '`.pk`' files, and since the default METAFONT mode on our installation is `ljfour` with a base resolution of 600 dpi (dots per inch), this instantiation is returned.

```
> kpsewhich -dpi=300 wsuipa10.pk
```

In this case, when specifying that we are interested in a resolution of 300 dpi (`-dpi=300`) we see that no such font is available on the system. A program like dvips or xdvi would go off and actually build the required `.pk` files using the script mktexpk.

Next we turn our attention to dvips's header and configuration files. We first look at one of the commonly used files, the general prologue `tex.pro` for TEX support, before turning our attention to the generic configuration file (`config.ps`) and the PostScript font map `psfonts.map` — as of 2004, map and encoding files have their own search paths and new location in `texmf` trees. As the '`.ps`' suffix is ambiguous we have to specify explicitly which type we are considering (`dvips config`) for the file `config.ps`.

```
> kpsewhich tex.pro
   /usr/local/texmf/dvips/base/tex.pro
> kpsewhich --format="dvips config" config.ps
   /usr/local/texmf/dvips/config/config.ps
> kpsewhich psfonts.map
   /usr/local/texmf/fonts/map/dvips/updmap/psfonts.map
```

We now take a closer look at the URW Times PostScript support files. The prefix for these in the standard font naming scheme is 'utm'. The first file we look at is the configuration file, which contains the name of the map file:

```
> kpsewhich --format="dvips config" config.utm
   /usr/local/texmf-dist/dvips/psnfss/config.utm
```

The contents of that file is

```
p +utm.map
```

which points to the file utm.map, which we want to locate next.

```
> kpsewhich utm.map
   /usr/local/texmf-dist/fonts/map/dvips/times/utm.map
```

This map file defines the file names of the Type 1 PostScript fonts in the URW collection. Its contents look like (we only show part of the lines):

```
utmb8r  NimbusRomNo9L-Medi    ... <utmb8a.pfb
utmbi8r NimbusRomNo9L-MediItal... <utmbi8a.pfb
utmr8r  NimbusRomNo9L-Regu    ... <utmr8a.pfb
utmri8r NimbusRomNo9L-ReguItal... <utmri8a.pfb
utmbo8r NimbusRomNo9L-Medi    ... <utmb8a.pfb
utmro8r NimbusRomNo9L-Regu    ... <utmr8a.pfb
```

Let us, for instance, take the Times Roman instance utmr8a.pfb and find its position in the texmf directory tree with a search for Type 1 font files:

```
> kpsewhich utmr8a.pfb
   /usr/local/texmf-dist/fonts/type1/urw/times/utmr8a.pfb
```

It should be evident from these examples how you can easily locate the whereabouts of a given file. This is especially important if you suspect that the wrong version of a file is picked up somehow, since kpsewhich will show you the first file encountered.

7.2.4 Debugging actions

Sometimes it is necessary to investigate how a program resolves file references. To make this practical, Kpathsea offers various levels of debugging output:

- 1 stat calls (disk lookups). When running with an up-to-date ls-R database this should almost give no output.
- 2 References to hash tables (such as ls-R databases, map files, configuration files).
- 4 File open and close operations.
- 8 General path information for file types searched by Kpathsea. This is useful to find out where a particular path for the file was defined.
- 16 Directory list for each path element (only relevant for searches on disk).
- 32 File searches.
- 64 Variable values.

A value of -1 will set all the above options; in practice, this is usually the most convenient.

Similarly, with the dvips program, by setting a combination of debug switches, one can follow in detail where files are being picked up from. Alternatively, when a file is not found, the debug trace shows in which directories the program looks for the given file, so that one can get an indication what the problem is.

Generally speaking, as most programs call the Kpathsea library internally, one can select a debug option by using the KPATHSEA_DEBUG environment variable, and setting it to (a combination of) values as described in the above list.

(Note for Windows users: it is not easy to redirect all messages to a file in this system. For diagnostic purposes you can temporarily SET KPATHSEA_DEBUG_OUTPUT=err.log).

Let us consider, as an example, a small LaTeX source file, hello-world.tex, which contains the following input.

```
debug:start search(file=texmf.cnf, must_exist=1, find_all=1,
  path=.:/usr/local/bin/texlive:/usr/local/bin:
       /usr/local/bin/texmf/web2c:/usr/local:
       /usr/local/texmf/web2c:/.:/./teTeX/TeX/texmf/web2c:).
kdebug:start search(file=ls-R, must_exist=1, find_all=1,
  path=~/tex:/usr/local/texmf).
kdebug:search(ls-R) =>/usr/local/texmf/ls-R
kdebug:start search(file=aliases, must_exist=1, find_all=1,
  path=~/tex:/usr/local/texmf).
kdebug:search(aliases) => /usr/local/texmf/aliases
kdebug:start search(file=config.ps, must_exist=0, find_all=0,
  path=.:~/tex:!!/usr/local/texmf/dvips//).
kdebug:search(config.ps) => /usr/local/texmf/dvips/config/config.ps
kdebug:start search(file=/root/.dvipsrc, must_exist=0, find_all=0,
  path=.:~/tex:!!/usr/local/texmf/dvips//).
search(file=/home/goossens/.dvipsrc, must_exist=1, find_all=0,
  path=.:~/tex/dvips//:!!/usr/local/texmf/dvips//).
kdebug:search($HOME/.dvipsrc) =>
kdebug:start search(file=config.cms, must_exist=0, find_all=0,
  path=.:~/tex/dvips//:!!/usr/local/texmf/dvips//).
kdebug:search(config.cms)
=>/usr/local/texmf/dvips/cms/config.cms
```

Figure 13: Finding configuration files

```
\documentclass{article}
\begin{document}
Hello World!
\end{document}
```

This little file only uses the font `cmr10`, so let us look at how dvips prepares the PostScript file (we want to use the Type 1 version of the Computer Modern fonts, hence the option `-Pcms`).

```
> dvips -d4100 hello-world -Pcms -o
```

In this case we have combined dvips's debug class 4 (font paths) with Kpathsea's path element expansion (see the dvips reference manual). The output (slightly rearranged) appears in Figure 13.

dvips starts by locating its working files. First, `texmf.cnf` is found, which gives the definitions of the search paths for the other files, then the file database `ls-R` (to optimize file searching) and the file `aliases`, which makes it possible to declare several names (e.g., a short DOS-like 8.3 and a more natural longer version) for the same file. Then dvips goes on to find the generic configuration file `config.ps` before looking for the customization file `.dvipsrc` (which, in this case is *not found*). Finally, dvips locates the config file for the Computer Modern PostScript fonts `config.cms` (this was initiated with the `-Pcms` option on the dvips command). This file contains the list of the map files which define the relation between the TEX, PostScript and file system names of the fonts.

```
> more /usr/local/texmf/dvips/cms/config.cms
   p +ams.map
   p +cms.map
   p +cmbkm.map
   p +amsbkm.map
```

dvips thus goes on to find all these files, plus the generic map file `psfonts.map`, which is always loaded (it contains declarations for commonly used PostScript fonts; see the last part of Section 7.2.3 for more details about PostScript map file handling).

At this point dvips identifies itself to the user:

```
This is dvips(k) 5.92b Copyright 2002 Radical Eye Software (www.radicaleye.com)
```

Then it goes on to look for the prolog file `texc.pro`:

```
kdebug:start search(file=texc.pro, must_exist=0, find_all=0,
  path=.:~/tex/dvips//:!!/usr/local/texmf/dvips//:
       ~/tex/fonts/type1//:!!/usr/local/texmf/fonts/type1//).
kdebug:search(texc.pro) => /usr/local/texmf/dvips/base/texc.pro
```

After having found the file in question, dvips outputs the date and time, and informs us that it will generate the file `hello-world.ps`, then that it needs the font file `cmr10`, and that the latter is declared as "resident" (no bitmaps needed):

```
TeX output 1998.02.26:1204' -> hello-world.ps
Defining font () cmr10 at 10.0pt
Font cmr10 <CMR10> is resident.
```

Now the search is on for the file `cmr10.tfm`, which is found, then a few more prolog files (not shown) are referenced, and finally the Type 1 instance `cmr10.pfb` of the font is located and included in the output file (see last line).

```
kdebug:start search(file=cmr10.tfm, must_exist=1, find_all=0,
  path=.:~/tex/fonts/tfm//:!!/usr/local/texmf/fonts/tfm//:
       /var/tex/fonts/tfm//).
kdebug:search(cmr10.tfm) => /usr/local/texmf/fonts/tfm/public/cm/cmr10.tfm
kdebug:start search(file=texps.pro, must_exist=0, find_all=0,
   ...
<texps.pro>
kdebug:start search(file=cmr10.pfb, must_exist=0, find_all=0,
  path=.:~/tex/dvips//:!!/usr/local/texmf/dvips//:
       ~/tex/fonts/type1//:!!/usr/local/texmf/fonts/type1//).
kdebug:search(cmr10.pfb) => /usr/local/texmf/fonts/type1/public/cm/cmr10.pfb
<cmr10.pfb>[1]
```

7.3 Runtime options

Another useful feature of Web2C is its possibility to control a number of memory parameters (in particular, array sizes) via the runtime file `texmf.cnf` read by Kpathsea. The memory settings can be found in Part 3 of that file in the TEX Live distribution. The more important are:

`main_memory` Total words of memory available, for TEX, METAFONT and MetaPost. You must make a new format file for each different setting. For instance, you could generate a "huge" version of TEX, and call the format file `hugetex.fmt`. Using the standard way of specifying the program name used by Kpathsea, the particular value of the `main_memory` variable will then be read from `texmf.cnf`.

`extra_mem_bot` Extra space for "large" TEX data structures: boxes, glue, breakpoints, etc. Especially useful if you use PICTEX.

`font_mem_size` Number of words for font information available for TEX. This is more or less the total size of all TFM files read.

`hash_extra` Additional space for the hash table of control sequence names. Only ≈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. The default value of `hash_extra` is 50000.

Of course, this facility is no substitute for truly dynamic arrays and memory allocation, but since these are extremely difficult to implement in the present TEX source, these runtime parameters provide a practical compromise allowing some flexibility.

8 Acknowledgements

TEX Live is a joint effort by virtually all of the TEX user groups. This edition of TEX Live was overseen by Karl Berry. The other principal contributors, past and present, are listed below.

- The English, German, Dutch, and Polish TEX user groups (TUG, DANTE e.V., NTG, and GUST, respectively), which provide the necessary technical and administrative infrastructure. Please join the TEX user group near you! (See http://tug.org/usergroups.html.)
- The CTAN team (http://ctan.org), which distributes the TEX Live images and provides the common infrastructure for package updates, upon which TEX Live depends.
- Nelson Beebe, for making many platforms available to TEX Live developers, and his own comprehensive testing and unparalleled bibliographic efforts.
- John Bowman, for making many changes to his advanced graphics program Asymptote to make it work in TEX Live.

- Peter Breitenlohner and the ε-TEX team for the stable foundation of future TEX's, and Peter specifically for years of stellar help with GNU autotools and keeping sources up to date. Peter passed away in October 2015, and we dedicate the continuing work to his memory.
- Jin-Hwan Cho and all of the DVIPDFM*x* team, for their excellent driver and responsiveness to configuration issues.
- Thomas Esser, without whose marvelous teTEX package TEX Live would have never existed.
- Michel Goossens, who co-authored the original documentation.
- Eitan Gurari, whose TEX4ht is used to create the HTML version of this documentation, and who worked tirelessly to improve it at short notice, every year. Eitan prematurely passed away in June 2009, and we dedicate this documentation to his memory.
- Hans Hagen, for much testing and making his ConTEXt package (http://pragma-ade.com) work within TEX Live's framework.
- Hàn Thế Thành, Martin Schröder, and the pdfTEX team (http://pdftex.org), for continuing enhancements of TEX's abilities.
- Hartmut Henkel, for significant development contributions to pdfTEX LuaTEX, and more.
- Taco Hoekwater, for major renewed development efforts on MetaPost and (Lua)TEX (http://luatex.org) itself, incorporating ConTEXt into TEX Live, giving Kpathsea multi-threaded functionality, and much more.
- Khaled Hosny, for substantial work on XeTEX, DVIPDFM*x*, and efforts with Arabic and other fonts.
- Paweł Jackowski, for the Windows installer tlpm, and Tomasz Łuczak, for tlpmgui, used in past releases.
- Akira Kakuto, for providing the Windows binaries from his W32TEX distribution for Japanese TEX (http://w32tex.org), and many other development contributions.
- Jonathan Kew, for developing the remarkable XeTEX engine and taking the time and trouble to integrate it in TEX Live, as well as the initial version of the MacTEX installer, and also for our recommended front-end TEXworks.
- Dick Koch, for maintaining MacTEX (http://tug.org/mactex) in very close tandem with TEX Live, and for his great good cheer in doing so.
- Reinhard Kotucha, for major contributions to the TEX Live 2008 infrastructure and installer, as well as Windows research efforts, the `getnonfreefonts` script, and more.
- Siep Kroonenberg, also for major contributions to the TEX Live 2008 infrastructure and installer, especially on Windows, and for the bulk of work updating this manual describing those features.
- Mojca Miklavec, for much help with ConTEXt, building many binary sets, and plenty more.
- Heiko Oberdiek, for the epstopdf package and many others, compressing the huge pst-geo data files so we could include them, and most of all, for his remarkable work on hyperref.
- Petr Olšak, who coordinated and checked all the Czech and Slovak material very carefully.
- Toshio Oshima, for his dviout previewer for Windows.
- Manuel Pégourié-Gonnard, for helping with package updates, documentation improvements, and texdoc development.
- Fabrice Popineau, for the original Windows support in TEX Live and work on the French documentation.
- Norbert Preining, the principal architect of the current TEX Live infrastructure and installer, and also for coordinating the Debian version of TEX Live (together with Frank Küster), and doing so much work along the way.
- Sebastian Rahtz, for originally creating TEX Live and maintaining it for many years. Sebastian passed away in March 2016, and we dedicate the continuing work to his memory.
- Luigi Scarso, for continuing development of MetaPost, LuaTEX, and much more.
- Tomasz Trzeciak, for wide-ranging help with Windows.
- Vladimir Volovich, for substantial help with porting and other maintenance issues, and especially for making it feasible to include xindy.
- Staszek Wawrykiewicz, a principal tester for all of TEX Live, and coordinator of the many major Polish contributions: fonts, Windows installation, and more. Staszek passed away in February 2018, and we dedicate the continuing work to his memory.
- Olaf Weber, for his patient maintenance of Web2C in past years.
- Gerben Wierda, for creating and maintaining the original Mac OS X support.
- Graham Williams, the originator of the TEX Catalogue.

Builders of the binaries: Marc Baudoin (amd64-netbsd, i386-netbsd), Ken Brown (i386-cygwin, x86_64-cygwin), Simon Dales (armhf-linux), Johannes Hielschier (aarch64-linux), Akira Kakuto (win32),

Dick Koch (x86_64-darwin), Nikola Lečić (amd64-freebsd, i386-freebsd), Henri Menke (x86_64-linuxmusl), Mojca Miklavec (i386-linux, x86_64-darwinlegacy, i386-solaris, x86_64-solaris, sparc-solaris), Norbert Preining (x86_64-linux), Thomas Schmitz (powerpc-linux), Boris Veytsman (armel-linux). For information on the TEX Live build process, see http://tug.org/texlive/build.html.

Translators of this manual: Denis Bitouzé & Patrick Bideault (French), Carlos Enriquez Figueras (Spanish), Jjgod Jiang, Jinsong Zhao, Yue Wang, & Helin Gai (Chinese), Nikola Lečić (Serbian), Marco Pallante & Carla Maggi (Italian), Petr Sojka & Jan Busa (Czech/Slovak), Boris Veytsman (Russian), Zofia Walczak (Polish), Uwe Ziegenhagen (German). The TEX Live documentation web page is http://tug.org/texlive/doc.html.

Of course the most important acknowledgement must go to Donald Knuth, first for inventing TEX, and then for giving it to the world.

9 Release history

9.1 Past

Discussion began in late 1993 when the Dutch TEX Users Group was starting work on its 4AllTEX CD for MS-DOS users, and it was hoped at that time to issue a single, rational, CD for all systems. This was too ambitious a target for the time, but it did spawn not only the very successful 4AllTEX CD, but also the TUG Technical Council working group on a *TEX Directory Structure* (http://tug.org/tds), which specified how to create consistent and manageable collections of TEX support files. A complete draft of the TDS was published in the December 1995 issue of *TUGboat*, and it was clear from an early stage that one desirable product would be a model structure on CD. The distribution you now have is a very direct result of the working group's deliberations. It was also clear that the success of the 4AllTEX CD showed that Unix users would benefit from a similarly easy system, and this is the other main strand of TEX Live.

We first undertook to make a new Unix-based TDS CD in the autumn of 1995, and quickly identified Thomas Esser's teTEX as the ideal setup, as it already had multi-platform support and was built with portability across file systems in mind. Thomas agreed to help, and work began seriously at the start of 1996. The first edition was released in May 1996. At the start of 1997, Karl Berry completed a major new release of Web2c, which included nearly all the features which Thomas Esser had added in teTEX, and we decided to base the 2nd edition of the CD on the standard Web2C, with the addition of teTEX's `texconfig` script. The 3rd edition of the CD was based on a major revision of Web2C, 7.2, by Olaf Weber; at the same time, a new revision of teTEX was being made, and TEX Live included almost all of its features. The 4th edition followed the same pattern, using a new version of teTEX, and a new release of Web2C (7.3). The system now included a complete Windows setup, thanks to Fabrice Popineau.

For the 5th edition (March 2000) many parts of the CD were revised and checked, updating hundreds of packages. Package details were stored in XML files. But the major change for TEX Live 5 was that all non-free software was removed. Everything in TEX Live is now intended to be compatible with the Debian Free Software Guidelines (http://www.debian.org/intro/free); we have done our best to check the license conditions of all packages, but we would very much appreciate hearing of any mistakes.

The 6th edition (July 2001) had much more material updated. The major change was a new install concept: the user could select a more exact set of needed collections. Language-related collections were completely reorganized, so selecting any of them installs not only macros, fonts, etc., but also prepares an appropriate `language.dat`.

The 7th edition of 2002 had the notable addition of Mac OS X support, and the usual myriad of updates to all sorts of packages and programs. An important goal was integration of the source back with teTEX, to correct the drift apart in versions 5 and 6.

9.1.1 2003

In 2003, with the continuing flood of updates and additions, we found that TEX Live had grown so large it could no longer be contained on a single CD, so we split it into three different distributions (see section 2.1, p. 4). In addition:

- At the request of the LATEX team, we changed the standard latex and pdflatex commands to now use ε-TEX (see p. 6).
- The new Latin Modern fonts were included (and are recommended).

- Support for Alpha OSF was removed (HPUX support was removed previously), since no one had (or volunteered) hardware available on which to compile new binaries.
- Windows setup was substantially changed; for the first time an integrated environment based on XEmacs was introduced.
- Important supplementary programs for Windows (Perl, Ghostscript, ImageMagick, Ispell) are now installed in the TEX Live installation directory.
- Font map files used by dvips, dvipdfm and pdftex are now generated by the new program updmap and installed into texmf/fonts/map.
- TEX, METAFONT, and MetaPost now, by default, output most input characters (32 and above) as themselves in output (e.g., \write) files, log files, and the terminal, i.e., *not* translated using the ^^ notation. In TEX Live 7, this translation was dependent on the system locale settings; now, locale settings do not influence the TEX programs' behavior. If for some reason you need the ^^ output, rename the file texmf/web2c/cp8bit.tcx. (Future releases will have cleaner ways to control this.)
- This documentation was substantially revised.
- Finally, since the edition numbers had grown unwieldy, the version is now simply identified by the year: TEX Live 2003.

9.1.2 2004

2004 saw many changes:

- If you have locally-installed fonts which use their own .map or (much less likely) .enc support files, you may need to move those support files.

  .map files are now searched for in subdirectories of fonts/map only (in each texmf tree), along the TEXFONTMAPS path. Similarly, .enc files are now searched for in subdirectories of fonts/enc only, along the ENCFONTS path. updmap will attempt to warn about problematic files.

  For methods of handling this and other information, please see http://tug.org/texlive/mapenc.html.

- The TEX Collection has been expanded with the addition of a MiKTEX-based installable CD, for those who prefer that implementation to Web2C. See section 2 (p. 4).

- Within TEX Live, the single large texmf tree of previous releases has been replaced by three: texmf, texmf-dist, and texmf-doc. See section 2.2 (p. 4), and the README files for each.

- All TEX-related input files are now collected in the tex subdirectory of texmf* trees, rather than having separate sibling directories tex, etex, pdftex, pdfetex, etc. See texmf-dist/doc/generic/tds/tds.html#Extensions.

- Helper scripts (not meant to be invoked by users) are now located in a new scripts subdirectory of texmf* trees, and can be searched for via kpsewhich -format=texmfscripts. So if you have programs which call such scripts, they'll need to be adjusted. See texmf-dist/doc/generic/tds/tds.html#Scripts.

- Almost all formats leave most characters printable as themselves via the "translation file" cp227.tcx, instead of translating them with the ^^ notation. Specifically, characters at positions 32–256, plus tab, vertical tab, and form feed are considered printable and not translated. The exceptions are plain TEX (only 32–126 printable), ConTEXt (0–255 printable), and the Ω-related formats. This default behavior is almost the same as in TEX Live 2003, but it's implemented more cleanly, with more possibilities for customization. See texmf-dist/doc/web2c/web2c.html#TCX-files. (By the way, with Unicode input, TEX may output partial character sequences when showing error contexts, since it is byte-oriented.)

- pdfetex is now the default engine for all formats except (plain) tex itself. (Of course it generates DVI when run as latex, etc.) This means, among other things, that the microtypographic features of pdftex are available in LATEX, ConTEXt, etc., as well as the ε-TEX features (texmf-dist/doc/etex/base/).

  It also means it's *more important than ever* to use the ifpdf package (works with both plain and LATEX) or equivalent code, because simply testing whether \pdfoutput or some other primitive is defined is not a reliable way to determine if PDF output is being generated. We made this

backward compatible as best we could this year, but next year, `\pdfoutput` may be defined even when DVI is being written.

- pdfTeX (http://pdftex.org) has many new features:
  - `\pdfmapfile` and `\pdfmapline` provide font map support from within a document.
  - Microtypographic font expansion can be used more easily. http://www.ntg.nl/pipermail/ntg-pdftex/2004-May/000504.html
  - All parameters previously set through the special configuration file `pdftex.cfg` must now be set through primitives, typically in `pdftexconfig.tex`; `pdftex.cfg` is no longer supported. Any extant `.fmt` files must be redumped when `pdftexconfig.tex` is changed.
  - See the pdfTeX manual for more: `texmf-dist/doc/pdftex/manual/pdftex-a.pdf`.
- The `\input` primitive in tex (and mf and mpost) now accepts double quotes containing spaces and other special characters. Typical examples:

  ```
  \input "filename with spaces"   % plain
  \input{"filename with spaces"}  % latex
  ```

  See the Web2C manual for more: `texmf-dist/doc/web2c`.
- encTeX support is now included within Web2C and consequently all TeX programs, via the `-enc` option — *only when formats are built*. encTeX supports general re-encoding of input and output, enabling full support of Unicode (in UTF-8). See `texmf-dist/doc/generic/enctex/` and http://www.olsak.net/enctex.html.
- Aleph, a new engine combining ε-TeX and Ω, is available. A little information is available in `texmf-dist/doc/aleph/base` and http://www.tex.ac.uk/cgi-bin/texfaq2html?label=aleph. The LaTeX-based format for Aleph is named lamed.
- The latest LaTeX release has a new version of the LPPL — now officially a Debian-approved license. Assorted other updates, see the `ltnews` files in `texmf-dist/doc/latex/base`.
- dvipng, a new program for converting DVI to PNG image files, is included. See http://www.ctan.org/pkg/dvipng.
- We reduced the cbgreek package to a "medium" sized set of fonts, with the assent and advice of the author (Claudio Beccari). The excised fonts are the invisible, outline, and transparency ones, which are relatively rarely used, and we needed the space. The full set is of course available from CTAN (http://mirror.ctan.org/tex-archive/fonts/greek/cbfonts).
- oxdvi has been removed; just use xdvi.
- The ini and vir commands (links) for tex, mf, and mpost are no longer created, such as initex. The ini functionality has been available through the command-line option `-ini` for years now.
- i386-openbsd platform support was removed. Since the tetex package in the BSD Ports system is available, and GNU/Linux and FreeBSD binaries were available, it seemed volunteer time could be better spent elsewhere.
- On sparc-solaris (at least), you may have to set the `LD_LIBRARY_PATH` environment variable to run the t1utils programs. This is because they are compiled with C++, and there is no standard location for the runtime libraries. (This is not new in 2004, but wasn't previously documented.) Similarly, on mips-irix, the MIPSpro 7.4 runtimes are required.

9.1.3 2005

2005 saw the usual huge number of updates to packages and programs. The infrastructure stayed relatively stable from 2004, but inevitably there were some changes there as well:

- New scripts texconfig-sys, updmap-sys, and fmtutil-sys were introduced, which modify the configuration in the system trees. The texconfig, updmap, and fmtutil scripts now modify user-specific files, under `$HOME/.texlive2005`.

- Corresponding new variables `TEXMFCONFIG` and `TEXMFSYSCONFIG` to specify the trees where configuration files (user or system, respectively) are found. Thus, you may need to move personal versions of `fmtutil.cnf` and `updmap.cfg` to these places; another option is to redefine `TEXMFCONFIG` or `TEXMFSYSCONFIG` in `texmf.cnf`. In any case the real location of these files and the values of `TEXMFCONFIG` and `TEXMFSYSCONFIG` must agree. See section 2.3, p. 5.
- Last year, we kept `\pdfoutput` and other primitives undefined for DVI output, even though the pdfetex program was being used. This year, as promised, we undid that compatibility measure. So if your document uses `\ifx\pdfoutput\undefined` to test if PDF is being output, it will need to be changed. You can use the package ifpdf.sty (which works under both plain TEX and LATEX) to do this, or steal its logic.
- Last year, we changed most formats to output (8-bit) characters as themselves (see previous section). The new TCX file `empty.tcx` now provides an easier way to get the original `^^` notation if you so desire, as in:

  ```
  latex --translate-file=empty.tcx yourfile.tex
  ```
- The new program dvipdfmx is included for translation of DVI to PDF; this is an actively maintained update of dvipdfm (which is also still available for now, though no longer recommended).
- The new programs pdfopen and pdfclose are included to allow reloading of PDF files in the Adobe Acrobat Reader without restarting the program. (Other PDF readers, notably xpdf, gv, and gsview, have never suffered from this problem.)
- For consistency, the variables `HOMETEXMF` and `VARTEXMF` have been renamed to `TEXMFHOME` and `TEXMFSYSVAR`, respectively. There is also `TEXMFVAR`, which is by default user-specific. See the first point above.

9.1.4 2006–2007

In 2006–2007, the major new addition to TEX Live was the XeTEX program, available as the `xetex` and `xelatex` programs; see http://scripts.sil.org/xetex.

MetaPost also received a notable update, with more planned for the future (http://tug.org/metapost/articles), likewise pdfTEX (http://tug.org/applications/pdftex).

The TEX `.fmt` (high-speed format) and the similar files for MetaPost and METAFONT are now stored in subdirectories of `texmf/web2c`, instead of in the directory itself (although the directory is still searched, for the sake of existing `.fmt`'s). The subdirectories are named for the 'engine' in use, such as `tex` or `pdftex` or `xetex`. This change should be invisible in normal use.

The (plain) `tex` program no longer reads `%&` first lines to determine what format to run; it is the pure Knuthian TEX. (LATEX and everything else do still read `%&` lines).

Of course the year also saw (the usual) hundreds of other updates to packages and programs. As usual, please check CTAN (http://mirror.ctan.org) for updates.

Internally, the source tree is now stored in Subversion, with a standard web interface for viewing the tree, as linked from our home page. Although not visible in the final distribution, we expect this will provide a stable development foundation for future years.

Finally, in May 2006 Thomas Esser announced that he would no longer be updating teTEX (http://tug.org/tetex). As a result, there was a surge of interest in TEX Live, especially among GNU/Linux distributors. (There is a new `tetex` installation scheme in TEX Live, which provides an approximate equivalent.) We hope this will eventually translate to improvements in the TEX environment for everyone.

9.1.5 2008

In 2008, the entire TEX Live infrastructure was redesigned and reimplemented. Complete information about an installation is now stored in a plain text file `tlpkg/texlive.tlpdb`.

Among other things, this finally makes possible upgrading a TEX Live installation over the Internet after the initial installation, a feature MiKTEX has provided for many years. We expect to regularly update new packages as they are released to CTAN.

The major new engine LuaTEX (http://luatex.org) is included; besides a new level of flexibility in typesetting, this provides an excellent scripting language for use both inside and outside of TEX documents.

Support among Windows and the Unix-based platforms is now much more uniform. In particular, most Perl and Lua scripts are now available on Windows, using the Perl internally distributed with TEX Live.

The new tlmgr script (section 5) is the general interface for managing TEX Live after the initial installation. It handles package updates and consequent regeneration of formats, map files, and language files, optionally including local additions.

With the advent of tlmgr, the texconfig actions to edit the format and hyphenation configuration files are now disabled.

The xindy indexing program (http://xindy.sourceforge.net/) is now included on most platforms.

The kpsewhich tool can now report all matches for a given file (option -all) and limit matches to a given subdirectory (option -subdir).

The dvipdfmx program now includes functionality to extract bounding box information, via the command name extractbb; this was one of the last features provided by dvipdfm not in dvipdfmx.

The font aliases Times-Roman, Helvetica, and so on have been removed. Different packages expected them to behave differently (in particular, to have different encodings), and there was no good way to resolve this.

The platex format has been removed, to resolve a name conflict with a completely different Japanese platex; the polski package is now the main Polish support.

Internally, the WEB string pool files are now compiled into the binaries, to ease upgrades.

Finally, the changes made by Donald Knuth in his 'TEX tuneup of 2008' are included in this release. See http://tug.org/TUGboat/Articles/tb29-2/tb92knut.pdf.

9.1.6 2009

In 2009, the default output format for Lua(LA)TEX is now PDF, to take advantage of LuaTEX's OpenType support, et al. New executables named dviluatex and dvilualatex run LuaTEX with DVI output. The LuaTEX home page is http://luatex.org.

The original Omega engine and Lambda format have been excised, after discussions with the Omega authors. The updated Aleph and Lamed remain, as do the Omega utilities.

A new release of the AMS Type 1 fonts is included, including Computer Modern: a few shape changes made over the years by Knuth in the Metafont sources have been integrated, and the hinting has been updated. The Euler fonts have been thoroughly reshaped by Hermann Zapf (see http://tug.org/TUGboat/Articles/tb29-2/tb92hagen-euler.pdf). In all cases, the metrics remain unchanged. The AMS fonts home page is http://www.ams.org/tex/amsfonts.html.

The new GUI front end TEXworks is included for Windows, and also in MacTEX. For other platforms, and more information, see the TEXworks home page, http://tug.org/texworks. It is a cross-platform front end inspired by the Mac OS X TeXShop editor, aiming at ease-of-use.

The graphics program Asymptote is included for several platforms. This implements a text-based graphics description language vaguely akin to MetaPost, but with advanced 3D support and other features. Its home page is http://asymptote.sourceforge.net.

The separate dvipdfm program has been replaced by dvipdfmx, which operates in a special compatibility mode under that name. dvipdfmx includes CJK support and has accumulated many other fixes over the years since the last dvipdfm release.

Executables for the cygwin and i386-netbsd platforms are now included, while we were advised that OpenBSD users get TEX through their package systems, plus there were difficulties in making binaries that have a chance of working on more than one version.

A miscellany of smaller changes: we now use xz compression, the stable replacement for lzma (http://tukaani.org/xz/); a literal $ is allowed in filenames when it does not introduce a known variable name; the Kpathsea library is now multi-threaded (made use of in MetaPost); the entire TEX Live build is now based on Automake.

Final note on the past: all releases of TEX Live, along with ancillary material such as CD labels, are available at ftp://tug.org/historic/systems/texlive.

9.1.7 2010

In 2010, the default version for PDF output is now 1.5, enabling more compression. This applies to all the TEX engines when used to produce PDF and to dvipdfmx. Loading the pdf14 LATEX package changes back to PDF 1.4, or set \pdfminorversion=4.

pdf(LA)TEX now *automatically* converts a requested Encapsulated PostScript (EPS) file to PDF, via the epstopdf package, when and if the LATEX graphics.cfg configuration file is loaded, and

PDF is being output. The default options are intended to eliminate any chance of hand-created PDF files being overwritten, but you can also prevent `epstopdf` from being loaded at all by putting `\newcommand{\DoNotLoadEpstopdf}{}` (or `\def...`) before the `\documentclass` declaration. It is also not loaded if the pst-pdf package is used. For more details, see the `epstopdf` package documentation (http://ctan.org/pkg/epstopdf-pkg).

A related change is that execution of a very few external commands from TEX, via the `\write18` feature, is now enabled by default. These commands are `repstopdf`, `makeindex`, `kpsewhich`, `bibtex`, and `bibtex8`; the list is defined in `texmf.cnf`. Environments which must disallow all such external commands can deselect this option in the installer (see section 3.2.4), or override the value after installation by running `tlmgr conf texmf shell_escape 0`.

Yet another related change is that BIBTEX and Makeindex now refuse to write their output files to an arbitrary directory (like TEX itself), by default. This is so they can now be enabled for use by the restricted `\write18`. To change this, the `TEXMFOUTPUT` environment variable can be set, or the `openout_any` setting changed.

XeTEX now supports margin kerning along the same lines as pdfTEX. (Font expansion is not presently supported.)

By default, tlmgr now saves one backup of each package updated (`tlmgr option autobackup 1`), so broken package updates can be easily reverted with `tlmgr restore`. If you do post-install updates, and don't have the disk space for the backups, run `tlmgr option autobackup 0`.

New programs included: the pTEX engine and related utilities for typesetting Japanese; the BIBTEXU program for Unicode-enabled BIBTEX; the chktex utility (http://baruch.ev-en.org/proj/chktex) for checking (LA)TEX documents; the dvisvgm (http://dvisvgm.sourceforge.net) DVI-to-SVG translator.

Executables for these new platforms are now included: `amd64-freebsd`, `amd64-kfreebsd`, `i386-freebsd`, `i386-kfreebsd`, `x86_64-darwin`, `x86_64-solaris`.

A change in TEX Live 2009 that we failed to note: numerous TEX4ht-related executables (http://tug.org/tex4ht) were removed from the binary directories. The generic `mk4ht` program can be used to run any of the various `tex4ht` combinations.

Finally, the TEX Live release on the TEX Collection DVD can no longer be run live (oddly enough). A single DVD no longer has enough room. One beneficial side effect is that installation from the physical DVD is much faster.

9.1.8 2011

The Mac OS X binaries (`universal-darwin` and `x86_64-darwin`) now work only on Leopard or later; Panther and Tiger are no longer supported.

The `biber` program for bibliography processing is included on common platforms. Its development is closely coupled with the `biblatex` package, which completely reimplements the bibliographical facilities provided by LaTeX.

The MetaPost (`mpost`) program no longer creates or uses `.mem` files. The needed files, such as `plain.mp`, are simply read on every run. This is related to supporting MetaPost as a library, which is another significant though not user-visible change.

The `updmap` implementation in Perl, previously used only on Windows, has been revamped and is now used on all platforms. There shouldn't be any user-visible changes as a result, except that it runs much faster.

The initex and inimf programs were restored (but no other ini* variants).

9.1.9 2012

`tlmgr` supports updates from multiple network repositories. The section on multiple repositories in the `tlmgr help` output has more.

The parameter `\XeTeXdashbreakstate` is set to 1 by default, for both `xetex` and `xelatex`. This allows line breaks after em-dashes and en-dashes, which has always been the behavior of plain TEX, LATEX, LuaTEX, etc. Existing XeTEX documents which must retain perfect line-break compatibility will need to set `\XeTeXdashbreakstate` to 0 explicitly.

The output files generated by `pdftex` and `dvips`, among others, can now exceed 2 gigabytes.

The 35 standard PostScript fonts are included in the output of `dvips` by default, since so many different versions of them are extant.

In the restricted `\write18` execution mode, set by default, `mpost` is now an allowed program.

A `texmf.cnf` file is also found in `../texmf-local`, e.g., `/usr/local/texlive/texmf-local/web2c/texmf.cnf`, if it exists.

The `updmap` script reads a per-tree `updmap.cfg` instead of one global config. This change should be invisible, unless you edited your updmap.cfg's directly. The `updmap --help` output has more.

Platforms: armel-linux and mipsel-linux added; sparc-linux and i386-netbsd are no longer in the main distribution.

9.1.10 2013

Distribution layout: the top-level `texmf/` directory has been merged into `texmf-dist/`, for simplicity. Both the `TEXMFMAIN` and `TEXMFDIST` Kpathsea variables now point to `texmf-dist`.

Many small language collections have been merged together, to simplify installation.

MetaPost: native support for PNG output and floating-point (IEEE double) has been added.

LuaTeX: updated to Lua 5.2, and includes a new library (`pdfscanner`) to process external PDF page content, among much else (see its web pages).

XeTeX (also see its web pages for more):

- The HarfBuzz library is now used for font layout instead of ICU. (ICU is still used to support input encodings, bidirectionality, and the optional Unicode line breaking.)
- Graphite2 and HarfBuzz are used instead of SilGraphite for Graphite layout.
- On Macs, Core Text is used instead of the (deprecated) ATSUI.
- Prefer TrueType/OpenType fonts to Type1 when the names are the same.
- Fix occasional mismatch in font finding between XeTeX and `xdvipdfmx`.
- Support OpenType math cut-ins.

xdvi: now uses FreeType instead of `t1lib` for rendering.

microtype.sty: some support for XeTeX (protrusion) and LuaTeX (protrusion, font expansion, tracking), among other enhancements.

tlmgr: new `pinning` action to ease configuring multiple repositories; that section in `tlmgr --help` has more, online at http://tug.org/texlive/doc/tlmgr.html#MULTIPLE-REPOSITORIES.

Platforms: armhf-linux, mips-irix, i386-netbsd, and amd64-netbsd added or revived; powerpc-aix removed.

9.1.11 2014

2014 saw another TeX tune-up from Knuth; this affected all engines, but the only visible change likely is the restoration of the `preloaded format` string on the banner line. Per Knuth, this now reflects the format that *would* be loaded by default, rather than an undumped format that is actually preloaded in the binary; it may be overridden in various ways.

pdfTeX: new warning-suppression parameter `\pdfsuppresswarningpagegroup`; new primitives for fake interword spaces to help with PDF text reflowing: `\pdfinterwordspaceon`, `\pdfinterwordspaceoff`, `\pdffakespace`.

LuaTeX: Notable changes and fixes were made to font loading and hyphenation. The biggest addition is a new engine variant, `luajittex` (http://foundry.supelec.fr/projects/luajittex) and its siblings `texluajit` and `texluajitc`. This uses a just-in-time Lua compiler (detailed *TUGboat* article at http://tug.org/TUGboat/tb34-1/tb106scarso.pdf). `luajittex` is still in development, is not available on all platforms, and is considerably less stable than `luatex`. Neither we nor its developers recommend using it except for the specific purpose of experimenting with jit on Lua code.

XeTeX: The same image formats are now supported on all platforms (including Mac); avoid Unicode compatibility decomposition fallback (but not other variants); prefer OpenType to Graphite fonts, for compatibility with previous XeTeX versions.

MetaPost: A new numbersystem `decimal` is supported, along with a companion internal `numberprecision`; a new definition of `drawdot` in `plain.mp`, per Knuth; bug fixes in SVG and PNG output, among others.

The pstopdf ConTeXt utility will be removed as a standalone command at some point after the release, due to conflicts with OS utilities of the same name. It can still (and now) be invoked as `mtxrun -script pstopdf`.

psutils has been substantially revised by a new maintainer. As a result, several seldom-used utilities (`fix*`, `getafm`, `psmerge`, `showchar`) are now only in the `scripts/` directory rather than being user-level executables (this can be reversed if it turns out to be problematic). A new script, `psjoin`, has been added.

The MacTEX redistribution of TEX Live (section 3.1.2) no longer includes the optional Mac-only packages for the Latin Modern and TEX Gyre fonts, since it is easy enough for individual users to make them available to the system. The convert program from ImageMagick has also been excised, since TEX4ht (specifically `tex4ht.env`) now uses Ghostscript directly.

The langcjk collection for Chinese, Japanese, and Korean support has been split into individual language collections for the sake of more moderate sizes.

Platforms: x86_64-cygwin added, mips-irix removed; Microsoft no longer supports Windows XP, so our programs may start failing there at any time.

9.1.12 2015

LATEX 2_ε now incorporates, by default, changes previously included only by explicitly loading the fixltx2e package, which is now a no-op. A new latexrelease package and other mechanisms allow for controlling what is done. The included LATEX News #22 and "LATEX changes" documents have details. Incidentally, the babel and psnfss packages, while core parts of LATEX, are maintained separately and are not affected by these changes (and should still work).

Internally, LATEX 2_ε now includes Unicode-related engine configuration (what characters are letters, naming of primitives, etc.) which was previously part of TEX Live. This change is intended to be invisible to users; a few low-level internal control sequences have been renamed or removed, but the behavior should be just the same.

pdfTEX: Support JPEG Exif as well as JFIF; do not emit a warning if `\pdfinclusionerrorlevel` is negative; sync with xpdf 3.04.

LuaTEX: New library newtokenlib for scanning tokens; bug fixes in the `normal` random number generator and other places.

XeTEX: Image handling fixes; xdvipdfmx binary looked for first as a sibling to xetex; internal `XDV` opcodes changed.

MetaPost: New numbersystem `binary`; new Japanese-enabled upmpost and updvitomp programs, analogous to up*tex.

MacTEX: Updates to the included Ghostscript package for CJK support. The TEX Distribution Preference Pane now works in Yosemite (Mac OS X 10.10). Resource-fork font suitcases (generally without an extension) are no longer supported by XeTEX; data-fork suitcases (`.dfont`) remain supported.

Infrastructure: The fmtutil script has been reimplemented to read `fmtutil.cnf` on a per-tree basis, analogous to updmap. Web2C mktex* scripts (including mktexlsr, mktextfm, mktexpk) now prefer programs in their own directory, instead of always using the existing `PATH`.

Platforms: *-kfreebsd removed, since TEX Live is now easily available through the system platform mechanisms. Support for some additional platforms is available as custom binaries (http://tug.org/texlive/custom-bin.html). In addition, some platforms are now omitted from the DVD (simply to save space), but can be installed normally over the net.

9.1.13 2016

LuaTEX: Sweeping changes to primitives, both renames and removals, along with some node structure rearrangements. The changes are summarized in an article by Hans Hagen, "LuaTEX 0.90 backend changes for PDF and more" (http://tug.org/TUGboat/tb37-1/tb115hagen-pdf.pdf); for all the details, see the LuaTEX manual, `texmf-dist/doc/luatex/base/luatex.pdf`.

Metafont: New highly experimental sibling programs MFlua and MFluajit, integrating Lua with METAFONT, for trial testing purposes.

MetaPost: Bug fixes and internal preparations for MetaPost 2.0.

`SOURCE_DATE_EPOCH` support in all engines except LuaTEX (which will come in the next release) and original `tex` (intentionally omitted): if the environment variable `SOURCE_DATE_EPOCH` is set, its value is used for timestamps in the PDF output. If `SOURCE_DATE_EPOCH_TEX_PRIMITIVES` is also set, the `SOURCE_DATE_EPOCH` value is used to initialize the TEX primitives `\year`, `\month`, `\day`, `\time`. The pdfTEX manual has examples and details.

pdfTEX: new primitives `\pdfinfoomitdate`, `\pdftrailerid`, `\pdfsuppressptexinfo`, to control values appearing in the output which normally change with each run. These features are for PDF output only, not DVI.

XeTEX: New primitives `\XeTeXhyphenatablelength`, `\XeTeXgenerateactualtext`, `\XeTeXinterwordspaceshaping`, `\mdfivesum`; character class limit increased to 4096; DVI id byte incremented.

Other utilities:

- `gregorio` is a new program, part of the `gregoriotex` package for typesetting Gregorian chant scores; it is included in `shell_escape_commands` by default.
- `upmendex` is an index creation program, mostly compatible with `makeindex`, with support for Unicode sorting, among other changes.
- `afm2tfm` now makes only accent-based height adjustments upward; a new option `-a` omits all adjustments.
- `ps2pk` can handle extended PK/GF fonts.

MacTEX: The TEX Distribution Preference Pane is gone; its functionality is now in TeX Live Utility; bundled GUI applications upgraded; new script `cjk-gs-integrate` to be run by users who wish to incorporate various CJK fonts into Ghostscript.

Infrastructure: System-level `tlmgr` configuration file supported; verify package checksums; if GPG is available, verify signature of network updates. These checks happen with both the installer and `tlmgr`. (If GPG is not available, updates proceed as usual.)

Platforms: `alpha-linux` and `mipsel-linux` removed.

9.1.14 2017

LuaTEX: More callbacks, more typesetting control, more access to internals; `ffi` library for dynamic code loading added on some platforms.

pdfTEX: Environment variable `SOURCE_DATE_EPOCH_TEX_PRIMITIVES` from last year renamed to `FORCE_SOURCE_DATE`, with no changes in functionality; if the `\pdfpageattr` token list contains the string `/MediaBox`, omit output of the default `/MediaBox`.

XeTEX: Unicode/OpenType math now based on HarfBuzz's MATH table support; some bug fixes.

Dvips: Make the last papersize special win, for consistency with `dvipdfmx` and package expectations; the `-L0` option (`L0` config setting) restores the previous behavior of the first special winning.

epTEX, eupTEX: New primitives `\pdfuniformdeviate`, `\pdfnormaldeviate`, `\pdfrandomseed`, `\pdfsetrandomseed`, `\pdfelapsedtime`, `\pdfresettimer`, from pdfTEX.

MacTEX: As of this year, only Mac OS X releases for which Apple still releases security patches will be supported in MacTEX, under the platform name `x86_64-darwin`; currently this means Yosemite, El Capitan, and Sierra (10.10 and newer). Binaries for older Mac OS X versions are not included in MacTEX, but are still available in TEX Live (`x86_64-darwinlegacy`, `i386-darwin`, `powerpc-darwin`).

Infrastructure: The `TEXMFLOCAL` tree is now searched before `TEXMFSYSCONFIG` and `TEXMFSYSVAR` (by default); the hope is that this will better match expectations of local files overriding system files. Also, `tlmgr` has a new mode `shell` for interactive and scripted use, and a new action `conf auxtrees` to easily add and remove extra trees.

`updmap` and `fmtutil`: These scripts now give a warning when invoked without explicitly specifying either so-called system mode (`updmap-sys`, `fmtutil-sys`, or option `-sys`), or user mode (`updmap-user`, `fmtutil-user`, or option `-user`). The hope is that this will reduce the perennial problem of invoking user mode by accident and thus losing future system updates. See http://tug.org/texlive/scripts-sys-user.html for details.

`install-tl`: Personal paths such as `TEXMFHOME` are now set to MacTEX values (`~/Library/...`) by default on Macs. New option `-init-from-profile` to start an installation with the values from a given profile; new command `P` to explicitly save a profile; new profile variable names (but previous ones are still accepted).

SyncTEX: the temporary file name now looks like `foo.synctex(busy)`, instead of `foo.synctex.gz(busy)` (no `.gz`). Front ends and build systems that want to remove temp files may need adjusting.

Other utilities: `texosquery-jre8` is a new cross-platform program for retrieving locale and other OS information from a TEX document; it is included in `shell_escape_commands` by default for restricted shell execution. (Older JRE versions are supported by texosquery, but cannot be enabled in restricted mode, as they are no longer supported by Oracle, even for security issues.)

Platforms: See MacTEX entry above; no other changes.

9.2 Present—2018

Kpathsea: Case-insensitive filename matching now done by default in non-system directories; set `texmf.cnf` or environment variable `texmf_casefold_search` to `0` to disable. Full details in the Kpathsea manual (http://tug.org/kpathsea).

epTEX, eupTEX: New primitive `\epTeXversion`.

LuaTEX: Preparation for moving to Lua 5.3 in 2019: a binary `luatex53` is available on most platforms, but must be renamed to `luatex` to be effective. Or use the ConTEXt Garden (http://wiki.contextgarden.net) files; more information there.

MetaPost: Fixes for wrong path directions, TFM and PNG output.

pdfTEX: Allow encoding vectors for bitmap fonts; current directory not hashed into PDF ID; bug fixes for `\pdfprimitive` and related.

XeTEX: Support `/Rotate` in PDF image inclusion; exit nonzero if the output driver fails; various obscure UTF-8 and other primitive fixes.

MacTEX: See version support changes below. In addition, the files installed in `/Applications/TeX/` by MacTEX have been reorganized for greater clarity; now this location contains four GUI programs (BibDesk, LaTeXiT, TeX Live Utility, and TeXShop) at the top level and folders with additional utilities and documentation.

`tlmgr`: new front-ends `tlshell` (Tcl/Tk) and `tlcockpit` (Java); JSON output; `uninstall` now a synonym for `remove`; new action/option `print-platform-info`.

Platforms:

- New: `x86_64-linuxmusl` and `aarch64-linux`. Removed: `armel-linux`, `powerpc-linux`.
- `x86_64-darwin` supports 10.10–10.13 (Yosemite, El Capitan, Sierra, and High Sierra).
- `x86_64-darwinlegacy` supports 10.6–10.10 (though `x86_64-darwin` is preferred for 10.10). All support for 10.5 (Leopard) is gone, that is, both the `powerpc-darwin` and `i386-darwin` `platforms` have been removed.
- Windows: XP is no longer supported.

9.3 Future

TEX Live is not perfect, and never will be. We intend to continue to release new versions, and would like to provide more documentation, more programs, an ever-improved and better-checked tree of macros and fonts, and anything else TEX. This work is all done by volunteers in their spare time, and so there is always more to do. Please see http://tug.org/texlive/contribute.html.

Please send corrections, suggestions, and offers of help to:

tex-live@tug.org
http://tug.org/texlive

Happy TEXing!

Executing TEX

Abstract
Much of the LUA code in CONTEXT originates from experiments. When it survives in the source code it is probably used, waiting to be used or kept for educational purposes. The functionality that we describe here has already been present for a while in CONTEXT, but improved a little starting with LUATEX 1.08 due to an extra helper. The code shown here is generic and not used in CONTEXT as such.

Say that we have this code:

```
for i=1,10000 do
    tex.sprint("1")
    tex.sprint("2")
    for i=1,3 do
        tex.sprint("3")
        tex.sprint("4")
        tex.sprint("5")
    end
    tex.sprint("\\space")
end
```

When we call `\directlua` with this snippet we get some 30 pages of `12345345345`. The printed text is saved till the end of the LUA call, so basically we pipe some 170.000 characters to TEX that get interpreted as one paragraph.

Now imagine this:

```
\setbox0\hbox{xxxxxxxxxxx} \number\wd0
```

which gives 3595950. If we check the box in LUA, with:

```
tex.sprint(tex.box[0].width)
tex.sprint("\\enspace")
tex.sprint("\\setbox0\\hbox{!}")
tex.sprint(tex.box[0].width)
```

the result is `3595950 3595950`, which is not what you would expect at first sight. However, if you consider that we just pipe to a TEX buffer that gets parsed after the LUA call, it will be clear that the reported width is the width that we started with. It will work all right if we say:

```
tex.sprint(tex.box[0].width)
tex.sprint("\\enspace")
tex.sprint("\\setbox0\\hbox{!}")
tex.sprint("\\directlua{tex.sprint(tex.box[0].width)}")
```

because now we get: `3595950 301500`. It's not that complex to write some support code that makes this more convenient. This can work out quite well but there is a drawback. If we use this code:

```
print(status.input_ptr)
tex.sprint(tex.box[0].width)
tex.sprint("\\enspace")
tex.sprint("\\setbox0\\hbox{!}")
```

```
tex.sprint("\\directlua{print(status.input_ptr)\
    tex.sprint(tex.box[0].width)}")
```

Here we get 6 and 7 reported. You can imagine that when a lot of nested \directlua calls happen, we can get an overflow of the input level or (depending on what we do) the input stack size. Ideally we want to do a LUA call, temporarily go to TEX, return to LUA, etc. without needing to worry about nesting and possible crashes due to LUA itself running into problems. One charming solution is to use so-called coroutines: independent LUA threads that one can switch between — you jump out from the current routine to another and from there back to the current one. However, when we use \directlua for that, we still have this nesting issue and what is worse, we keep nesting function calls too. This can be compared to:

```
\def\whatever{\ifdone\whatever\fi}
```

where at some point \ifdone is false so we quit. But we keep nesting when the condition is met, so eventually we can end up with some nesting related overflow. The following:

```
\def\whatever{\ifdone\expandafter\whatever\fi}
```

is less likely to overflow because there we have tail recursion which basically boils down to not nesting but continuing. Do we have something similar in LUATEX for LUA? Yes, we do. We can register a function, for instance:

```
lua.get_functions_table()[1] = function() print("Hi there!") end
```

and call that one with:

```
\luafunction 1
```

This is a bit faster than calling a function like:

```
\directlua{HiThere()}
```

which can also be achieved by

```
\directlua{print("Hi there!")}
```

which sometimes can be more convenient. Anyway, a function call is what we can use for our purpose as it doesn't involve interpretation and effectively behaves like a tail call. The following snippet shows what we have in mind:

```
tex.routine(function()
    tex.sprint(tex.box[0].width)
    tex.sprint("\\enspace")
    tex.sprint("\\setbox0\\hbox{!}")
    tex.yield()
    tex.sprint(tex.box[0].width)
end)
```

We start a routine, jump out to TEX in the middle, come back when we're done and continue. This gives us: 3595950 188640, which is what we expect.

3595950 188640

This mechanism permits efficient (nested) loops like:

```
tex.routine(function()
    for i=1,10000 do
        tex.sprint("1")
        tex.yield()
        tex.sprint("2")
        tex.routine(function()
            for i=1,3 do
```

```
                tex.sprint("3")
                tex.yield()
                tex.sprint("4")
                tex.yield()
                tex.sprint("5")
            end
        end)
        tex.sprint("\\space")
        tex.yield()
    end
end)
```

We do create coroutines, go back and forwards between LUA and TEX, but avoid memory being filled up with printed content. If we flush paragraphs (instead of e.g. the space) then the main difference is that instead of a small delay due to the loop unfolding in a large set of prints and accumulated content, we now get a steady flushing and processing.

However, we can still have an overflow of input buffers because we still nest them: the limitation at the TEX end has moved to a limitation at the LUA end. How come? Here is the code that we use:

```
local stepper = nil
local stack   = { }
local fid     = 0xFFFFFF
local goback  = "\\luafunction" .. fid .. "\\relax"

function tex.resume()
    if coroutine.status(stepper) == "dead" then
        stepper = table.remove(stack)
    end
    if stepper then
        coroutine.resume(stepper)
    end
end

lua.get_functions_table()[fid] = tex.resume

function tex.yield()
    tex.sprint(goback)
    coroutine.yield()
    texio.closeinput()
end

function tex.routine(f)
    table.insert(stack,stepper)
    stepper = coroutine.create(f)
    tex.sprint(goback)
end
```

The `routine` creates a coroutine, and `yield` gives control to TEX. The `resume` is done at the TEX end when we're finished there. In practice this works fine and when you permit enough nesting and levels in TEX then you will not easily overflow.

When I picked up this side project and wondered how to get around it, it suddenly struck me that if we could just quit the current input level then nesting would not be a problem. Adding a simple helper to the engine made that possible (of course figuring it out took a while):

```
local stepper = nil
local stack   = { }
```

```
local fid     = 0xFFFFFF
local goback  = "\\luafunction" .. fid .. "\\relax"

function tex.resume()
    if coroutine.status(stepper) == "dead" then
        stepper = table.remove(stack)
    end
    if stepper then
        coroutine.resume(stepper)
    end
end

lua.get_functions_table()[fid] = tex.resume

if texio.closeinput then
    function tex.yield()
        tex.sprint(goback)
        coroutine.yield()
        texio.closeinput()
    end
else
    function tex.yield()
        tex.sprint(goback)
        coroutine.yield()
    end
end

function tex.routine(f)
    table.insert(stack,stepper)
    stepper = coroutine.create(f)
    tex.sprint(goback)
end
```

The trick is in `texio.closeinput`, a recent helper and one that should be used with care. We assume that the user knows what she or he is doing. On an old laptop with a i7-3840 processor running WINDOWS 10 the following snippet takes less than 0.35 seconds with LUATEX and 0.26 seconds with LUAJITTEX.

```
tex.routine(function()
    for i=1,10000 do
        tex.sprint("\\setbox0\\hpack{x}")
        tex.yield()
        tex.sprint(tex.box[0].width)
        tex.routine(function()
            for i=1,3 do
                tex.sprint("\\setbox0\\hpack{xx}")
                tex.yield()
                tex.sprint(tex.box[0].width)
            end
        end)
    end
end)
```

Say that we run the bad snippet:

```
for i=1,10000 do
    tex.sprint("\\setbox0\\hpack{x}")
    tex.sprint(tex.box[0].width)
    for i=1,3 do
```

```
        tex.sprint("\\setbox0\\hpack{xx}")
        tex.sprint(tex.box[0].width)
    end
end
```

This time we need 0.12 seconds in both engines. So what if we run this:

```
\dorecurse{10000}{%
    \setbox0\hpack{x}
    \number\wd0
    \dorecurse{3}{%
        \setbox0\hpack{xx}
        \number\wd0
    }%
}
```

Pure TEX needs 0.30 seconds for both engines but there we lose 0.13 seconds on the loop code. In the LUA example where we yield, the loop code takes hardly any time. As we need only 0.05 seconds more it demonstrates that when we use the power of LUA the performance hit of the switch is quite small: we yield 40.000 times! In general, such differences are far exceeded by the overhead: the time needed to typeset the content (which `\hpack` doesn't do), breaking paragraphs into lines, constructing pages and other overhead involved in the run. In CONTEXT we use a slightly different variant which has 0.30 seconds more overhead, but that is probably true for all LUA usage in CONTEXT, but again, it disappears in other runtime.

Here is another example:

```
\def\TestWord#1%
  {\directlua{
     tex.routine(function()
       tex.sprint("\\setbox0\\hbox{\\tttf #1}")
       tex.yield()
       tex.sprint(math.round(100 * tex.box[0].width/tex.hsize))
       tex.sprint(" percent of the hsize: ")
       tex.sprint("\\box0")
     end)
  }}

The width of next word is \TestWord {inline}!
```

The width of next word is 8 percent of the hsize: `inline`!

Now, in order to stay realistic, this macro can also be defined as:

```
\def\TestWord#1%
  {\setbox0\hbox{\tttf #1}%
   \directlua{
      tex.sprint(math.round(100 * tex.box[0].width/tex.hsize))
   } %
   percent of the hsize: \box0\relax}
```

We get the same result: "The width of next word is 8 percent of the hsize: `inline`!".

We have been using a LUA-TEX mix for over a decade now in CONTEXT, and have never really needed this mixed model. There are a few places where we could (have) benefited from it and we might use it in a few places, but so far we have done fine without it. In fact, in most cases typesetting can be done fine at the TEX end. It's all a matter of imagination.

Hans Hagen

Variable fonts

Introduction

History shows the tendency to recycle ideas. Often quite some effort is made by historians to figure out what really happened, not just long ago, when nothing was written down and we had to do with stories or pictures at most, but also in recent times. Descriptions can be conflicting, puzzling, incomplete, partially lost, biased, ...

Just as language was invented (or evolved) several times, so were scripts. The same might be true for rendering scripts on a medium. Semaphores came and went within decades and how many people know now that they existed and that encryption was involved? Are the old printing presses truly the old ones, or are older examples simply gone? One of the nice aspects of the internet is that one can now more easily discover similar solutions for the same problem, but with a different (and independent) origin.

So, how about this "new big thing" in font technology: variable fonts. In this case, history shows that it's not that new. For most TEX users the names METAFONT and METAPOST will ring bells. They have a very well documented history so there is not much left to speculation. There are articles, books, pictures, examples, sources, and more around for decades. So, the ability to change the appearance of a glyph in a font depending on some parameters is not new. What probably *is* new is that creating variable fonts is done in the natural environment where fonts are designed: an interactive program. The METAFONT toolkit demands quite some insight in programming shapes in such a way that one can change look and feel depending on parameters. There are not that many meta fonts made and one reason is that making them requires a certain mind- and skill set. On the other hand, faster computers, interactive programs, evolving web technologies, where real-time rendering and therefore more or less real-time tweaking of fonts is a realistic option, all play a role in acceptance.

But do interactive font design programs make this easier? You still need to be able to translate ideas into usable beautiful fonts. Taking the common shapes of glyphs, defining extremes and letting a program calculate some interpolations will not always bring good results. It's like morphing a picture of your baby's face into yours of old age (or that of your grandparent): not all intermediate results will look great. It's good to notice that variable fonts are a revival of existing techniques and ideas used in, for instance, multiple master fonts. The details might matter even more as they can now be exaggerated when some transformation is applied.

There is currently (March 2017) not much information about these fonts so what I say next may be partially wrong or at least different from what is intended. The perspective will be one from a TEX user and coder. Whatever you think of them, these fonts will be out there and for sure there will be nice examples circulating soon. And so, when I ran into a few experimental fonts, with POSTSCRIPT and TRUETYPE outlines, I decided to have a look at what is inside. After all, because it's visual, it's also fun to play with. Let's stress that at the moment of this writing I only have a few simple fonts available, fonts that are designed for testing and not usage. Some recommended tables were missing and no complex OPENTYPE features are used in these fonts.

The specification

I'm not that good at reading specifications, first of all because I quickly fall asleep with such documents, but most of all because I prefer reading other stuff (I do have lots of books waiting to be read). I'm also someone who has to play with something in order to understand it: trial and error is my modus operandi. Eventually it's my intended usage that drives the interface and that is when everything comes together.

Exploring this technology comes down to: locate a font, get the OpenType 1.8 specification from the Microsoft website, and try to figure out what is in the font. When I had a rough idea the next step was to get to the shapes and see if I could manipulate them. Of course it helped that in ConTeXt we already can load fonts and play with shapes (using MetaPost). I didn't have to install and learn other programs. Once I could render them, in this case by creating a virtual font with inline PDF literals, a next step was to apply variation. Then came the first experiments with a possible user interface. Seeing more variation then drove the exploration of additional properties needed for typesetting, like features.

The main extension to the data packaged in a font file concerns the (to be discussed) axis along which variable fonts operate and deltas to be applied to coordinates. The `gdef` table has been extended and contains information that is used in `gpos` features. There are new `hvar`, `vvar` and `mvar` tables that influence the horizontal, vertical and general font dimensions. The `gvar` table is used for TrueType variants, while the `cff2` table replaces the `cff` table for OpenType PostScript outlines. The `avar` and `stat` tables contain some meta-information about the axes of variations.

It must be said that because this is new technology the information in the standard is not always easy to understand. The fact that we have two rendering techniques, PostScript `cff` and TrueType `ttf`, also means that we have different information and perspectives. But this situation is not much different from OpenType standards a few years ago: it takes time but in the end I will get there. And, after all, users also complain about the lack of documentation for ConTeXt, so who am I to complain? In fact, it will be those ConTeXt users who will provide feedback and make the implementation better in the end.

Loading

Before we discuss some details, it will be useful to summarize what the font loader does when a user requests a font at a certain size and with specific features enabled. When a font is used the first time, its binary format is converted into a form that makes it suitable for use within ConTeXt and therefore LuaTeX. This conversion involves collecting properties of the font as a whole (official names, general dimensions like x-height and em-width, etc.), of glyphs (dimensions, Unicode properties, optional math properties), and all kinds of information that relates to (contextual) replacements of glyphs (small caps, oldstyle, scripts like Arabic) and positioning (kerning, anchoring marks, etc.). In the ConTeXt font loader this conversion is done in Lua.

The result is stored in a condensed format in a cache and the next time the font is needed it loads in an instant. In the cached version the dimensions are untouched, so a font at different sizes has just one copy in the cache. Often a font is needed at several sizes and for each size we create a copy with scaled glyph dimensions. The feature-related dimensions (kerning, anchoring, etc.) are shared and scaled when needed. This happens when sequences of characters in the node list get converted into sequences of glyphs. We could do the same with glyph dimensions but one reason for having a scaled copy is that this copy can also contain virtual glyphs and

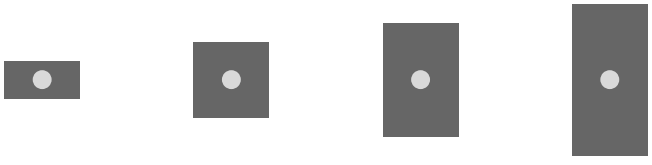
these have to be scaled beforehand. In practice there are several layers of caching in order to keep the memory footprint within reasonable bounds.[1]

When the font is actually used, interaction between characters is resolved using the feature-related information. When for instance two characters need to be kerned, a lookup results in the injection of a kern, scaled from general dimensions to the current size of the font.

When the outlines of glyphs are needed in MetaFun the font is also converted from its binary form to something in Lua, but this time we filter the shapes. For a `cff` this comes down to interpreting the `charstrings` and reducing the complexity to `moveto`, `lineto` and `curveto` operators. In the process subroutines are inlined. The result is something that MetaPost is happy with but that also can be turned into a piece of a PDF.

We now come to what a variable font actually is: a basic design which is transformed along one or more axes. A simple example is wider shapes:

We can also go taller and retain the width:



Here we have a linear scaling but glyphs are not normally done that way. There are font collections out there with lots of intermediate variants (say from light to heavy) and it's more profitable to sell each variant independently. However, there is often some logic behind it, probably supported by programs that designers use, so why not build that logic into the font and have one file that represents many intermediate forms. In fact, once we have multiple axes, even when the designer has clear ideas of the intended usage, nothing will prevent users from tinkering with the axis properties in ways that will fulfil their demands but hurt the designers eyes. We will not discuss that dilemma here.

When a variable font follows the route described above, we face a problem. When you load a TrueType font it will just work. The glyphs are packaged in the same format as static fonts. However, a variable font has axes and on each axis a value can be set. Each axis has a minimum, maximum and default. It can be that the default instance also assumes some transformations are applied. The standard recommends adding tables to describe these things but the fonts that I played with each lacked such tables. So that leaves some guesswork. But still, just loading a TrueType font gives some sort of outcome, although the dimensions (widths) might be weird due to lack of a (default) axis being applied.

An OpenType font with PostScript outlines is different: the internal `cff` format has been upgraded to `cff2` which on the one hand is less complicated but on the other hand has a few new operators — which results in programs that have not been adapted complaining or simply quitting on them.

One could argue that a font is just a resource and that one only has to pass it along but that's not what works well in practice. Take LuaTeX. We can of course load the font and apply axis vales so that we can process the document as we normally do. But

1. In retrospect one can wonder if that makes sense; just look at how much memory a browser uses when it has been open for some time. In the beginning of LuaTeX users wondered about caching fonts, but again, just look at what amounts browsers cache: it gets pretty close to the average amount of writes that a SSD can handle per day within its guaranteed life span.

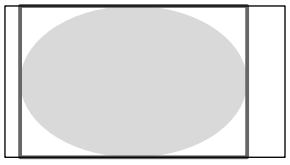
at some point we have to create a PDF. We can simply embed the TrueType files but no axis values are applied. This is because, even if we add the relevant information, there is no way in current PDF formats to deal with it. For that, we should be able to pass all relevant axis-related information as well as specify what values to use along these axes. And for TrueType fonts this information is not part of the shape description so then we in fact need to filter and pass more. An OpenType PostScript font is much cleaner because there we have the information needed to transform the shape mostly in the glyph description. There we only need to carry some extra information on how to apply these so-called blend values. The region/axis model used there only demands passing a relatively simple table (stripped down to what we need). But, as said above, `cff2` is not backward-compatible so a viewer will (currently) simply not show anything.

Recalling how we load fonts, how does that translate with variable changes? If we have two characters with glyphs that get transformed and that have a kern between them, the kern may or may not transform. So, when we choose values on an axis, then not only glyph properties change but also relations. We can no longer share positional information and scale afterwards because each instance can have different values to start with. We could carry all that information around and apply it at runtime but because we're typesetting documents with a static design it's more convenient to just apply it once and create an instance. We can use the same caching as mentioned before but each chosen instance (provided by the font or made up by user specifications) is kept in the cache. As a consequence, using a variable font has no overhead, apart from initial caching.

So, having dealt with that, how do we proceed? Processing a font is not different from what we already had. However, I would not be surprised if users are not always satisfied with, for instance, kerning, because in such fonts a lot of care has to be given to this by the designer. Of course I can imagine that programs used to create fonts deal with this, but even then, there is a visual aspect to it too. The good news is that in ConTeXt we can manipulate features so in theory one can create a so-called font goodie file for a specific instance.

Shapes

For OpenType PostScript shapes we always have to do a dummy rendering in order to get the right bounding box information. For TrueType this information is already present but not when we use a variable instance, so I had to do a bit of coding for that. Here we face a problem. For TeX we need the width, height and depth of a glyph. Consider the following case:



The shape has a bounding box that fits the shape. However, its left corner is not at the origin. So, when we calculate a tight bounding box, we cannot use it for actually positioning the glyph. We do use it (for horizontal scripts) to get the height and depth but for the width we depend on an explicit value. In OpenType PostScript we have the width available and how the shape is positioned relative to the origin doesn't much matter. In a TrueType shape a bounding box is part of the specification, as is the width, but for a variable font one has to use so-called phantom points to recalculate the width and the test fonts I had were not suitable for investigating this.

At any rate, once I could generate documents with typeset text using variable fonts it became time to start thinking about a user interface. A variable font can

have predefined instances but of course a user also wants to mess with axis values. Take one of the test fonts: Adobe Variable Font Prototype. It has several instances:

extralight	It looks like this!	`weight=0.0 contrast=0.0`
light	It looks like this!	`weight=150.0 contrast=0.0`
regular	It looks like this!	`weight=394.0 contrast=0.0`
semibold	**It looks like this!**	`weight=600.0 contrast=0.0`
bold	**It looks like this!**	`weight=824.0 contrast=0.0`
black high contrast	**It looks like this!**	`weight=1000.0 contrast=100.0`
black medium contrast	**It looks like this!**	`weight=1000.0 contrast=50.0`
black	**It looks like this!**	`weight=1000.0 contrast=0.0`

Such an instance is accessed with:

```
\definefont
  [MyLightFont]
  [name:adobevariablefontprototypelight*default]
```

The Avenir Next variable demo font (currently) provides:

regular	It looks like this!	`weight=400.0 width=100.0`
medium	It looks like this!	`weight=500.0 width=100.0`
bold	**It looks like this!**	`weight=700.0 width=100.0`
heavy	**It looks like this!**	`weight=900.0 width=100.0`
condensed	It looks like this!	`weight=400.0 width=75.0`
medium condensed	It looks like this!	`weight=500.0 width=75.0`
bold condensed	**It looks like this!**	`weight=700.0 width=75.0`
heavy condensed	**It looks like this!**	`weight=900.0 width=75.0`

Before we continue I will show a few examples of variable shapes. Here we use some MetaFun magic. Just take these definitions for granted.

```
\startMPcode
  draw outlinetext.b
    ("\definedfont[name:adobevariablefontprototypeextralight]foo@bar")
    (withcolor "gray")
    (withcolor red withpen pencircle scaled 1/10)
    xsized .45TextWidth ;
\stopMPcode

\startMPcode
  draw outlinetext.b
    ("\definedfont[name:adobevariablefontprototypelight]foo@bar")
    (withcolor "gray")
    (withcolor red withpen pencircle scaled 1/10)
    xsized .45TextWidth ;
\stopMPcode

\startMPcode
  draw outlinetext.b
    ("\definedfont[name:adobevariablefontprototypebold]foo@bar")
    (withcolor "gray")
    (withcolor red withpen pencircle scaled 1/10)
    xsized .45TextWidth ;
\stopMPcode

\startMPcode
  draw outlinetext.b
    ("\definefontfeature[whatever][axis={weight:350}]%
      \definedfont[name:adobevariablefontprototype*whatever]foo@bar")
```

```
    (withcolor "gray")
    (withcolor red withpen pencircle scaled 1/10)
    xsized .45TextWidth ;
\stopMPcode
```

The results are shown in figure 1. What we see here is that as long as we fill the shape everything will look as expected but using only an outline won't. The crucial (control) points are moved to different locations and as a result they can end up inside the shape. Giving up outlines is the price we evidently need to pay. Of course this is not unique for variable fonts although in practice static fonts behave better. To some extent we're back to where we were with METAFONT and (for instance) Computer Modern: because these originate in bitmaps (and probably use similar design logic) we also can have overlap and bits and pieces pasted together and no one will notice that. The first outline variants of Computer Modern also had such artifacts while in the static Latin Modern successors, outlines were cleaned up.

Figure 1. Four variants

The fact that we need to preprocess an instance but only know how to do that when we have gotten the information about axis values from the font means that the font handler has to be adapted to keep caching correct. Another definition is:

```
\definefontfeature
  [lightdefault]
  [default]
  [axis={weight:230,contrast:50}]

\definefont
  [MyLightFont]
  [name:adobevariablefontprototype*lightdefault]
```

Here the complication is that where normally features are dealt with after loading, the axis feature is part of the preparation (and caching). If you want the virtual font solution you can do this:

```
\definefontfeature
  [inlinelightdefault]
  [default]
  [axis={weight:230,contrast:50},
   variableshapes=yes]

\definefont
  [MyLightFont]
  [name:adobevariablefontprototype*inlinelightdefault]
```

When playing with these fonts it was hard to see if loading was done right. For instance not all values make sense. It is beyond the scope of this article, but axes like weight, width, contrast and italic values get applied differently to so-called regions

(subspaces). So say that we have an x coordinate with value 50. This value can be adapted in, for instance, four subspaces (regions), so we actually get:

$$x' = x + s_1 \times x_1 + s_2 \times x_2 + s_3 \times x_3 + s_4 \times x_4$$

The (here) four scale factors s_n are determined by the axis value. Each axis has some rules about how to map the values 230 for weight and 50 for contrast to such a factor. And each region has its own translation from axis values to these factors. The deltas $x_1, ..., x_4$ are provided by the font. For a PostScript-based font we find sequences like:

```
1 <setvstore>
120 [10 -30 40 -60] 1 <blend> ... <operator>
100 120 [10 -30 40 -60] [30 -10 -30 20] 2 <blend> .. <operator>
```

A store refers to a region specification. From there the factors are calculated using the chosen values on the axis. The deltas are part of the glyphs specification. Officially there can be multiple region specifications, but how likely it is that they will be used in real fonts is an open question.

For TrueType fonts the deltas are not in the glyph specification but in a dedicated `gvar` table.

```
apply x deltas [10 -30 40 -60] to x 120
apply y deltas [30 -10 -30 20] to y 100
```

Here the deltas come from tables outside the glyph specification and their application is triggered by a combination of axis values and regions.

The following two examples use Avenir Next Variable and demonstrate that kerning is adapted to the variant.

```
\definefontfeature
  [default:shaped]
  [default]
  [axis={width:10}]

\definefont
  [SomeFont]
  [file:avenirnextvariable*default:shaped]
```

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely-praised program, called up on the screen, will make everything automatic from now on.

Hermann Zapf

```
\definefontfeature
  [default:shaped]
  [default]
  [axis={width:100}]

\definefont
  [SomeFont]
  [file:avenirnextvariable*default:shaped]
```

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the

rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely-praised program, called up on the screen, will make everything automatic from now on.

Hermann Zapf

Embedding

Once we're done typesetting and a PDF file has to be created there are three possible routes:

- ☐ We can embed the shapes as PDF images (inline literal) using virtual font technology. We cannot use so-called xforms here because we want to support color selectively in text.
- ☐ We can wait till the PDF format supports such fonts, which might happen but even then we might be stuck for years with viewers getting there. Also documents need to get printed, and when printer support might arrive is another unknown.
- ☐ We can embed a regular font with shapes that match the chosen values on the axis. This solution is way more efficient than the first.

Once I could interpret the right information in the font, the first route was the way to go. A side effect of having a converter for both outline types meant that it was trivial to create a virtual font at runtime. This option will stay in ConTeXt as pseudo-feature `variableshapes`.

When trying to support variable fonts I tried to limit the impact on the backend code. Also, processing features and such was not touched. The inclusion of the right shapes is done via a callback that requests the blob to be injected in the `cff` or `glyf` table. When implementing this I actually found out that the LuaTeX backend also does some juggling of charstrings, to serve the purpose of inlining subroutines. In retrospect I could have learned a few tricks faster by looking at that code but I never realized that it was there. Looking at the code again, it strikes me that the whole inclusion could be done with Lua code and some day I will give that a try.

Conclusion

When I first heard about variable fonts I was confident that when they showed up they could be supported. Of course a specimen was needed to prove this. A first implementation demonstrates that indeed it's no big deal to let ConTeXt with LuaTeX handle such fonts. Of course we need to fill in some gaps which can be done once we have complete fonts. And then of course users will demand more control. In the meantime the helper script that deals with identifying fonts by name has been extended and the relevant code has been added to the distribution. At some point the ConTeXt Garden will provide the LuaTeX binary that has the callback.

I end with a warning. On the one hand this technology looks promising but on the other hand one can easily get lost. Probably most such fonts operate over a well-defined domain of values but even then one should be aware of complex interactions with features like positioning or replacements. Not all combinations can be tested. It's probably best to stick to fonts that have all the relevant tables and don't depend on properties of a specific rendering technology.

Hans Hagen

TEX Gyre text fonts revisited

Introduction

The collection of the TEX Gyre (TG in short) family of text fonts was released by the GUST e-foundry in 2006 – 2009 [1, 2]. Having finished this task, the GUST e-foundry team started to work on the math companion (in the OpenType, OTF, format – see [5]) for the TG text fonts [3]. Work on the math companion was finished two years ago. It resulted in the broadening of the repertoire of glyphs that could be used not only in math mode but also in text mode in technical documents. Hans Hagen, indefatigably coming up with interesting ideas, proposed to migrate the relevant glyphs to the text TG fonts. Needless to say, we seized on Hans's suggestion.

The first step was to decide which glyphs are to be migrated (and/or improved). Obviously, the list of candidates grew and grew. All in all, about 1000 glyphs were designated to be added, mostly geometrical and math symbols. A math companion, so far, was provided only for serif fonts, thus the consistent enhancement of the repertoire of the sans-serif fonts was a working test for our fonts generator – cf. Section "The MetaType 1 engine" below.

We started with two fonts – the serif TG Pagella and the sans-serif TG Adventor. The results were satisfying. Now we are ready for the next step: to enhance similarly the rest of the TG family (TG Chorus which is hardly suitable for technical texts, needs individual approach). We believe, however, that we're over the hump. Below, we describe the most difficult and thus most interesting (to us) aspects of this stage of the TG project.

The MetaType 1 engine

The scheme of the new MetaType 1 workflow is depicted in Figure 1.

The main change in the engine consists of the replacement of several components (AWK plus perl plus T1utils) by Python code with the FontForge library (finally, the library is available both under Linux and Windows®). However, the FontForge library does not allow for a sufficiently detailed control over the contents of the AFM and PFM files being generated which necessitates additional steps for fine tuning these files (dashed arrows in Figure 1).

The converter from Type 1 fonts to MetaType 1 sources, actually written in AWK plus T1utils, has not been rewritten so far. We plan to rewrite it in Python with FontForge library and enhance it to process also TrueType and OpenType files.

Of course, MetaPost still is the main module for generating glyph shapes. However, instead of spreading the auxiliary information into several output files (including EPS files), a single auxiliary output file, containing all the information needed for further processing, is generated. We will refer to this file as *Olio Typographic Information* file, OTI. *Olio* is a traditional name for a potpourri (it appears, e.g., in Robert Burns's *Address to a Haggis* – "French ragout or olio"). An OTI file is a container of "assorted bites and fragrances", indeed. Below is a fragment of an OTI file for TG Pagella regular.

```
FNT FAMILY_NAME TeX Gyre Pagella
FNT HEADER_BYTE49 TeX Gyre Pagella
FNT GROUP_NAME TeX Gyre Pagella
FNT STYLE_NAME Regular
. . .
FNT WEIGHT Regular
FNT ITALIC_ANGLE 0
. . .
GLY A CODE 65
GLY A EPS 165
GLY A ANCHOR INBAS ALT.ogonek 623 -143
GLY A ANCHOR INBAS BOT_MAIN 392 -143
GLY A ANCHOR INBAS TOP_MAIN 392 819
GLY A WD 778 HT 692 DP 0 IC 6 GA 392
GLY A HSBW 778
GLY A BBX 15 -3 756 700
. . .
FNT FONT_DIMEN7 0.83
FNT DIMEN_NAME7 (extra space)
FNT FONT_DIMEN22 2.5
FNT DIMEN_NAME22 (math axis)
FNT HEADER_BYTE72 234
```

Each line of the OTI file contains either global information, concerning the whole font (prefix 'FNT'), or local, concerning a given glyph (prefix 'GLY' followed by the glyph name). We will not dwell too much on the details of the structure of OTI files as it will be documented elsewhere.

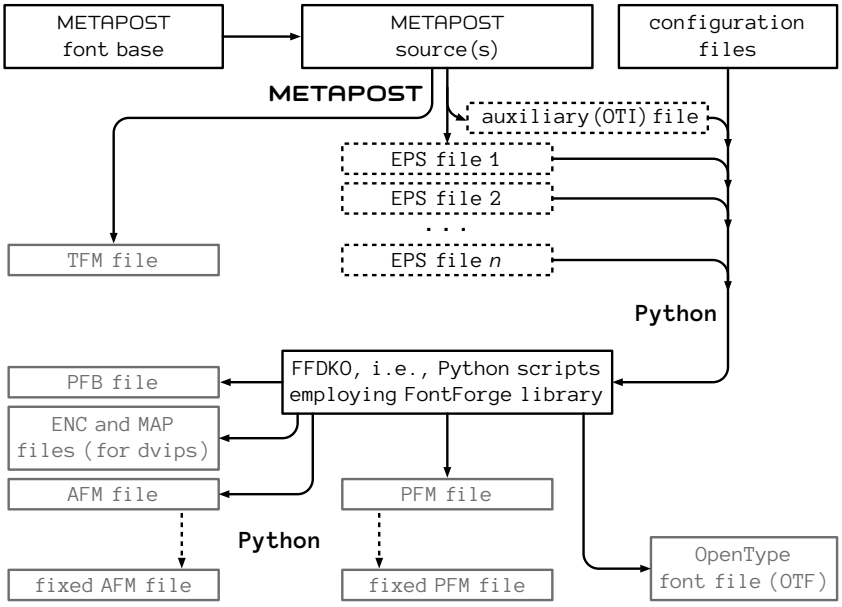


Figure 1. New MetaType 1 engine: working scheme

The glyph repertoire

As was mentioned, one of the important stimulus for the "face-lifting" of the TG text fonts were our efforts on TG math fonts: a lot of symbols do not need mathematical extension of the font structure (MATH table in OTF files); they, however, may prove useful in typesetting technical texts; for example, mathematical symbols (operators, relational symbols), arrows, geometrical symbols, etc. – see Figures 2 and 3 (next page). The number of glyphs grew from circa 750 to more than 1600. It may grow further in the future (see Section "Plans for the future").

The symbolic glyphs in the TG math fonts were designed only for regular serif variant fonts. The code, however, turned out flexible enough and with a few changes it was possible to generate the bold and sans-serif variants.

Apart from enriching the repertoire, many glyphs were amended, due to, among others, employing FontForge which, by default, minutely checks glyph outlines. For example, a tilde in TG Adventor was drawn from scratch, axes in several glyphs were corrected and so on.

Math-oriented glyphs existing so far in the text fonts have been replaced with slightly different forms, better suited for math formulas. The old forms can be reached, if required, by using the OTF mechanism called features [5, 6, 7], namely, the 'stylistic set' feature `ss10`. Moreover, the Pagella Greek alphabet was taken from TG Pagella Math, that is, from Diego Puga's excellent Mathpazo with the kind permission of the Author who agreed for us to use a fragment of his font under the GUST Font Licence (GFL, cf. [11]). The latter change involves significant change of the metric data. We are generally very reluctant to introduce such changes, but believe that the elegance of the Mathpazo Greek alphabet justifies that decision. Some glyphs from the Greek alphabet of TG Adventor (programmed in MetaType 1) required improvements which also implied changes in metric data.

$$f(x) = 1/x \qquad f(x) = 1/x$$
$$(x+1)(x-3) \geq 0 \qquad (x+1)(x-3) \geq 0$$

Figure 4. Default math-oriented glyphs (left) vs old glyphs produced by the OTF `ss10` feature (right)

Rolling with punches, we decided to abandon our initial idea of a full compatibility with the metric of the renowned Adobe 35 fonts [4]. The reason is two-fold: first, Adobe metric data is, as we pointed out in the documentation of the TG fonts [2], inconsistent in several cases; second, preserving full compatibility makes sense only when the relevant metric files are used for previewing PostScript files to be printed on a printer with built-in Adobe Type 1 fonts. The TG fonts might have been used for such previewing, but, as it turned out, they have been not (neither in Ghostscript nor in TeX Live; for example, the URW replacement for the Adobe 35 is used). Eventually, we decided to tune the TG metric data according to our experience whenever required. We believe that we will manage to avoid such changes in the future.

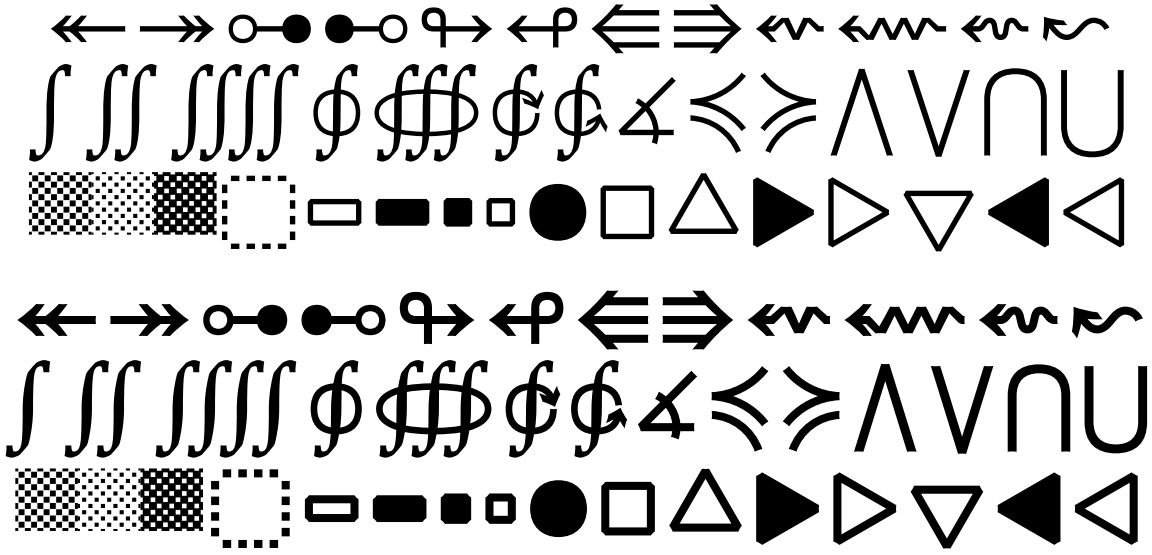


Figure 2. A sample of added glyphs: TG Pagella regular (top) and bold (bottom)

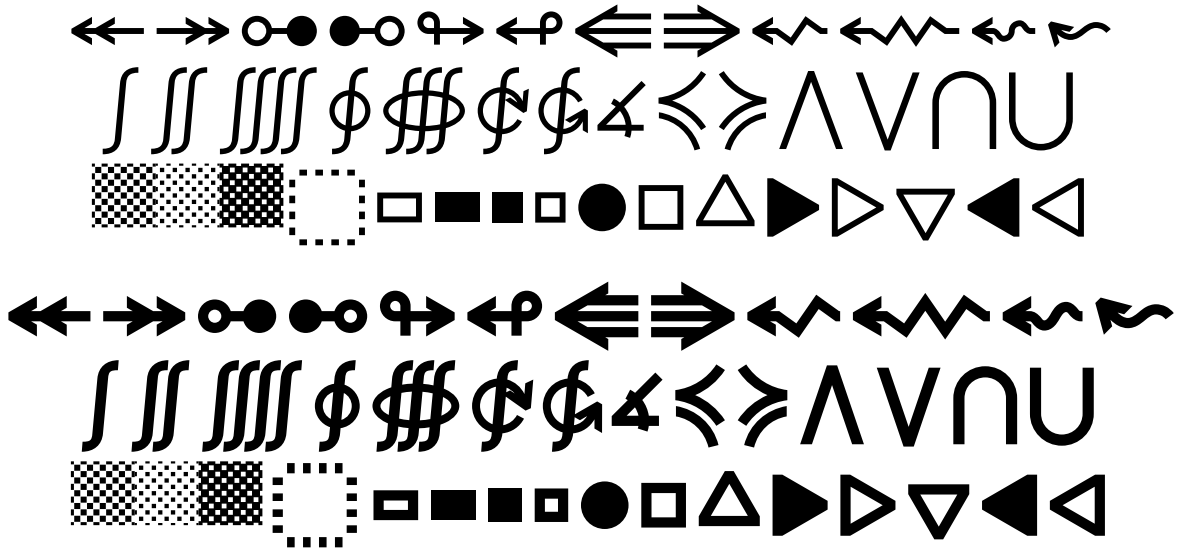


Figure 3. A sample of added glyphs: TG Adventor regular (top) and bold (bottom)

The font structure

The structure of the OTF fonts has been been enhanced with the "backward compatible math style" feature (`ss10`) mentioned above and, moreover, with the mechanism of *anchors*, although the name "snaps" seems to us to be more adequate. Anchors enable putting accents precisely over glyphs. Roughly speaking, the anchor mechanism can be considered the analogue of the TEX `\accent` mechanism. Anchors, however, are implemented in much more intricate way: three features, obscurely documented in [6, 7], namely, `ccmp` (glyph composition / decomposition), `mark` (mark positioning, precisely, accent-to-base or mark-to-base positioning), and `mkmk` (mark-to-mark positioning, or, in other words, accent-to-accent positioning)[1] are used for this purpose, and yet the OTF anchor mechanism turns out insufficiently efficacious.

We were suprised with the complexity and laboriousness of the implementation of such a simple concept. Having read the explanations below, the Reader and our virtual successors should feel forewarned and thus be less surprised.

"Anchors" or "marks" are actually pairs of numbers (planar points); the features `mark` and `mkmk` are supposed to position two glyphs in such a way that the respective anchors of the accent and accentee coincide. The former feature is used to position accents over or below basic glyphs, the latter one – to position accents over or below accents. In the TG fonts, following common practice, only so called combining accents (a subset of the block of the combining diacritical marks, [9], that is, zero-width glyphs, protruding entirely to the left) are used for accenting and, thus, are equipped with anchors. In order to reduce the amount of anchor data, we decided to use as anchored accentees only accentless Latin letters plus letters "welded" with cedilla, horn, ogonek, and, additionally, 'l·', 'L·', 'ł', 'Ł', 'ø', and 'Ø'.

The `ccmp` feature enables the transformation of the input stream, namely: replacing glyphs and assembling a series of glyphs into a composed character or disassembling a composed glyphs into a series of glyphs. The respective substitutions, in principle, must be defined in the font. Some engines, however, know better and perform such substitutions even if there is no relevant data in the font. For example, Microsoft Word® replaces 'i' (U+0069) followed by a combining top accent, say 'caroncomb' (U+030C), by a single glyph 'icaron' (U+01D0), provided that the latter is available in a given font; no further information, in particular, the `ccmp` feature, is required. Similarly, XeTeX joins accents with the basic glyph into a single glyph, provided that the assembled form is present in the font; otherwise, accents are being put using anchors. This behaviour cannot be turned off – XeTeX simply uses system libraries which know better…

In the TG fonts, the `ccmp` feature is used to disassemble accented glyphs (but not glyphs with cedilla, ogonek, or horns) and to join into a single glyph letters followed by combining cedilla, ogonek, or horn (provided that the resulting glyph belongs to the repertoire of a given font); otherwise, anchors are used. Moreover, `ccmp` is used for replacing of certain basic glyphs and accents by their alternative forms; for example, 'i' and 'j' in the vicinity of the top combining accents are replaced by their dotless forms, top combining accents following an uppercase letters or ascenders are replaced by their "high" (flattened) variants.

The process of accenting using anchors, seemingly a trivial task, is, in fact, quite sophisticated. Unicode Standard recommends that if a text processor is being fed with a stream of text data containing a glyph, having assigned a unicode slot, which is followed by a series of combining accents, then the text processor may position these accents over the main glyph [8], provided that the font contains the relevant positioning information. A typical example of the aplication of the anchor mechanism involving the `ccmp`+`mark`+`mkmk` features (as implemented in the new TG fonts) is depicted in Figure 5.

In the picture, the names of features written in small size denote the type of anchor (mark), large ones denote application of the respective features, labels 'TOP' and 'BOT' are defined by user; the assumed input string is: 'i', 'macronbelowcomb', 'caroncomb', 'tildecomb' (that is, in unicode lingo: U+0069 U+030C U+0331 U+0303). The anchors have descriptors given in braces: *donor* and *acceptor* (excerpted from physical chemistry).

The process of accenting works as follows in this case:

- first, the `ccmp` feature enters the scene: the letter 'i', when followed by a combining upper accent, is replaced with 'dotlessi';
- next, the `mark` feature acts: the 'caroncomb' glyph is placed over 'dotlessi' in such a manner that its 'TOP' donor anchor coincides with the 'TOP' acceptor anchor of the glyph 'dotlessi'; as a result, both anchors become inactive;
- next, the `mark` feature enters once again: the 'macronbelowcomb' glyph is placed below 'dotlessi' in such a manner that its 'BOT' donor anchor coincides with the 'BOT' acceptor anchor of the letter 'dotlessi'; as a result both anchors become inactive;
- finally, the `mkmk` feature intervenes: the 'tildecomb' glyph is placed above newly placed 'caroncomb' in such a manner that its 'TOP' donor anchor coincides with the 'BOT' acceptor anchor of the 'caroncomb' glyph; as a result, both anchors become inactive;
- the resulting assembled glyph has still two active anchors, 'TOP' and 'BOT', that could be used by the `mkmk` feature, provided that the relevant glyphs appear in the input stream (immediately after 'tildecomb' in this case).

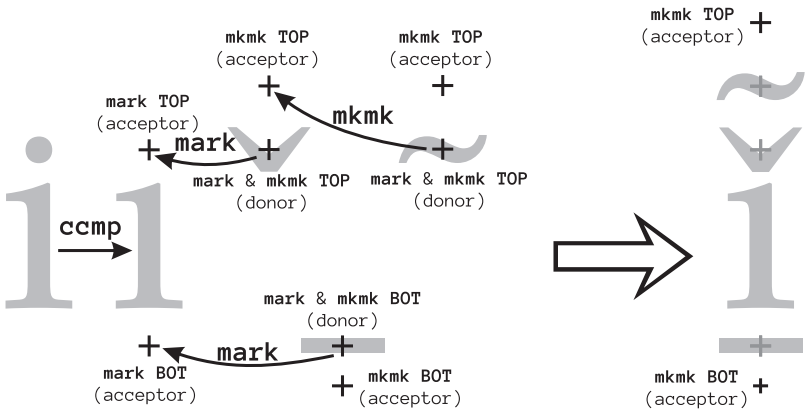


Figure 5. Anchor mechanism scheme – an example (explanations in the text)

As one can see, the process of assembling glyphs using anchors is actually fairly complex. It should be admitted, however, that it enables handling such peculiarities as replacing a caron glyph with a comma-like variant if glyphs 'l', 'L' or 'J' are to be accented with caron, replacing a comma accent by a turned comma accent above 'g' (normally comma accent goes below a letter), or a singular positioning of a dot below accent at a letter 'y', as shown in Figure 6.

```
L%
ˇ% U+030C (caroncmb, caroncomb)
g%
,% U+0326 (uni0326, commaaccentcomb)
y%
.% U+0323 (uni0323, dotbelowcomb)
```

Figure 6. Peculiar positioning of certain accents (TEX source – right; the result – left)

Unfortunately, not all practically important cases can be reliably handled. Notable example is the replacement of letters 'i' and 'j' by their dotless forms: the result depends to a large extent on the order of the glyphs in the input stream. In Figure 7, six cases are shown, with different order of glyphs in the input stream, namely (here, *i* stands for the letter 'i', *c* stands for 'caroncomb', and *m* stands for 'macronbelowcomb'): 1. *ic*; 2. *imc*; 3. *immc*; 4. *immmc*; 5. *immmmc*; 6. *icccmmmm*. Observe a malpositioned caron in one case – it is the result of our "design decision". The replacement 'i'→'dotlessi' is performed only if the top accents occur close to the letter 'i', preferably immediately after it. The OTF feature specification permits contextual replacements, that is, a certain number of bottom accents may precede the top one, but the preceding sequences must be enumerated explicitly. We decided to limit the length of the context to three glyphs (case 4. in Figure 7). If more bottom accents intervene between the letter 'i' and the top accent, the replacement is not performed and the glyphs are just overlapped (case 5. in Figure 7; as was mentioned, combining accents have zero width and protrude to the left). Some fonts define longer contexts (for example, Charis SIL), but we decided that for practical purposes three is enough.

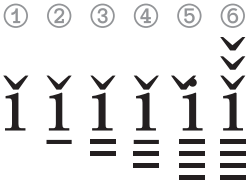


Figure 7. Troublesome replacement of 'i' by 'dotlessi' (explanations in the text)

In order to avoid such situations, we would recommend that the top accents should go first, next bottom accents (case 6. in Figure 7). The problem with our recommendation is that the order can be reversed by a text processing agent: according to the Unicode Standard recommendation, bottom accent should go first and "canonical ordering behavior cannot be overridden by higher-level protocols" [10]. Some text processing agents apply the algorithm defined in [10] at the phase of reading the unicode stream. In general, a typesetter cannot rely safely on the text processor. Even in TEX, the same text can be processed differently depending on the implementation.

In TEX, selected features, such as `ccmp`, `mark`, `mkmk`, etc., can be switched on or off on demand. Not all text processors offer such a possibility. Notable example is Microsoft Word® which has these features switched on by default (it is not obvious whether it makes use

of the Unicode ordering algorithm). As was mentioned, not all engines (in particular Microsoft Word®, but also XeTeX) obey rules coded in the features ccmp, mark, mkmk. Incidentally, Figure 7 was created usin LuaTeX.

In our opinion, the complexity of implementation of anchors, resulting in a variety of approaches and implementations, is caused by the oversimplified mechanism of the OTF specification: the only allowed operations on a glyph are (re)positioning and substituting which is directly related to the OTF table structure and the basic tables, namely, GPOS and GSUB. The former operation is restricted merely to shifting, the latter – to one-to-one, one-to-multiple and multiple-to-one replacements (which excludes reordering). Replacements can be either explicit or contextual, which adds complexity and does not help too much. In particular, fairly aged, not to say fossil, regular expressions are not allowed in contextual replacements.

Plans for the future

The next step (besides obvious cleaning of the sources, both Python and MetaPost) will undoubtedly be extending in a similar way of the remaining TG text fonts, both sans-serif (Heros) and serif (Bonum, Cursor, Schola, and Termes). TG Chorus, as a chancery font, is not suitable for such an extension. We consider naming the stylistic features used in the TG fonts – it needs consideration, however, wrong names may likely introduce mess rather than order.

Having gathered experience with the text fonts, we would like to revisit TG math fonts, with attention paid to sidebearings and math staircase kern.

Moreover, we plan to remove all non-Python modules. As was mentioned, the path MetaType 1 sources → OTF and Type 1 fonts is governed by Python; the reverse path, OTF and Type 1 fonts → MetaType 1 sources, currently employs gawk and T1utils, thus, it cannot be used for converting TTF and OTF fonts to MetaType 1 sources. We believe that the employing of FontForge (as a Python library) is the remedy.

We have no clear answer to the question whether "small figures", accessible by features subs (subscripts), sups (superscripts), sinf (scientific inferiors) numr (numerators), and dnom (denominators), should be included in the text fonts; in math fonts math sub- and superscripts can be used instead. If we include these glyphs, then next question arises: do we need special figures for small caps, smcp, other than traditional in the TeX realm old-style figures, also dubbed nautical? And do the small figures need variants commonly used for "normal" figures, that is, lnum (lining figures), onum (old-style figures), pnum (proportional figures), and tnum (tabular figures)? We are somewhat reluctant to add such a hodgepodge to already intricate font structure.

Acknowledgements

We are indebted to all people and TeX groups that supported our font enterprises. Almost all the GUST e-foundry projects were kindly supported by the Czechoslovak TeX Users Group CS TUG, the German-speaking TeX Users Group DANTE e.V., the Polish TeX Users Group GUST, the Dutch-speaking TeX Users Group NTG, TUG India, UK-TUG, and, last but not least, TUG. In a few cases, GUTenberg, the French-speaking TeX Users Group, supported us too.

The exceptional, personal thanks we owe to our friends who kept our spirits up for many years and tirelessly encouraged us to work on fonts: Hans Hagen, Johannes Küster, Jurek Ludwichowski, Volker RW Schaa, Jola Szelatyńska, Ulrik Vieth — hearty thanks!

References

[1] Bogusław Jackowski, Piotr Strzelczyk, and Piotr Pianowski. "GUST e-foundry font projects". In: *TUGBoat* 37.3 (2016), pp. 317–336.

[2] Bogusław Jackowski, Janusz M. Nowacki, and Piotr Strzelczyk. *TeX Gyre fonts collection.* URL: http://www.gust.org.pl/projects/e-foundry/tex-gyre (visited on 12/18/2017).

[3] Bogusław Jackowski, Piotr Strzelczyk, and Piotr Pianowski. *TeX Gyre math fonts collection.* URL: http://www.gust.org.pl/projects/e-foundry/tg-math (visited on 12/18/2017).

[4] Adobe Systems Incorporated. *Adobe Metric Files.* URL: ftp://ftp.adobe.com/pub/adobe/type/win/all/afmfiles/base35/ (visited on 02/02/2017).

[5] Microsoft Corporation. *OpenType Font Format, ver. 1.60, ISO/IEC 14496-22.* URL: https://www.microsoft.com/typography/otspec160/ (visited on 12/18/2017).

[6] Adobe Systems Incorporated. *Feature file syntax.* URL: https://www.adobe.com/devnet/opentype/afdko/topic_feature_file_syntax.html (visited on 04/17/2017).

[7] Microsoft Corporation. *Registered features.* URL: https://www.microsoft.com/typography/otspec/featurelist.htm (visited on 02/01/2017).

[8] Unicode Consortium. *The Unicode Standard 10.0.0; chapters 2.3. Compatibility Characters, 2.11. Combining Characters, and 2.12 Equivalent Sequences and Normalization.* URL: http://www.unicode.

org/versions/Unicode10.0.0/ch02.pdf (visited on 02/06/2017).

[9] Unicode Consortium. *Combining Diacritical Marks*. URL: https://www.unicode.org/charts/PDF/U0300.pdf (visited on 04/05/2018).

[10] Unicode Consortium. *The Unicode Standard 10.0.0; chapter 3.11. Normalization Forms*. URL: http://www.unicode.org/versions/Unicode10.0.0/ch03.pdf (visited on 02/06/2017).

[11] GUST e-Foundry. *GUST Font License*. URL: http://www.gust.org.pl/projects/e-foundry/licenses (visited on 12/30/2017).

Notes

1. Actually, there is one more anchor feature mset (mark positioning via substitution) meant for handling peculiarities of the typesetting of Arabic texts.

Bogusław Jackowski
b_jackowski (at) gust dot org dot pl
Piotr Pianowski
p.pianowski (at) gust dot org dot pl
Piotr Strzelczyk
p.strzelczyk (at) gust dot org dot pl

TLaunch, the TEX Live Launcher

Abstract
The TEX Live Launcher offers Windows users of a network TEX Live installation similar conveniences as a locally-installed TEX Live. It is easy to integrate additional TEX-related software.
This paper describes the launcher and its configuration. As an example, it shows how it is used at the Rijksuniversiteit Groningen.

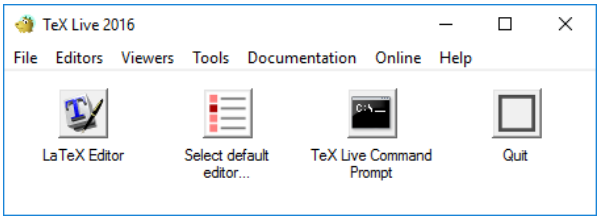


Figure 1. The default TEX Live Launcher

1 Overview

The TEX Live launcher gives users on Windows workstations easy access to a TEX Live installation already present on the network.

The launcher interface contains menus and buttons to invoke programs, and to access related local and online resources (see figure 1).

It also takes care of the usual Windows-specific configuration: at first run, TEX Live is added to the search path and relevant filetype associations are set up.

Because of prior experience with users running the initializer or installer when they really want to run the already initialized or installed TEX Live, I opted for a launcher that configures itself automatically, without requiring a separate initialization step.

Users can replace the default TEX editor from within the launcher interface, either with an editor defined in an ini file or with a third-party editor present on the filesystem (see figure 2).

For the sake of full access to the Windows API, I wrote the launcher in C. It has no dependencies whatsoever, aside from a TEX Live installation and Windows version 7 or later.

The launcher makes it easy for the TEX Live installation maintainer to add menu- or button controls and filetype associations for additional TEX-related software.

Filetypes, menus and buttons are defined in a Windows ini file. If necessary, pre- and post configuration script files can be configured as well.

The ini file included in TEX Live provides functionality more or less equivalent to the classic Windows TEX Live installation.

In the following sections, we have a more detailed look at the launcher and its configuration. The package documentation contains the full details.

Section 6 looks at the TEX Live installation at the Rijksuniversiteit Groningen, for which the launcher was created.

2 The ini file

The launcher reads its configuration from a conventional Windows ini file with sections, definitions and comment lines. If the TEX Live installation contains Windows binaries, then `tlaunch.exe` will be in the `bin/win32` directory, and `tlaunch.ini` in `texmf-dist/web2c`.[1] A custom ini file, supporting different software or with localized strings, can be placed in a higher-priority tree.

It is also possible to place `tlaunch.exe` and `tlaunch.ini` together in the root of the installation.

In general, entries which refer to non-existent items are silently ignored.

2.1 Strings

There is a Strings section for string variables. String variable names are case-insensitive. Some string variables, such as `%TLCONFIG%`, are required. This variable indicates the directory where the "forgetter" (see section 3) will be placed.

The optional variables `%PRE_CONFIG%`, `%POST_CONFIG%` and `%PRE_FORGET%` are the names of scripts to be run before and after configuration, and before forgetting respectively. The default values of these variables are empty strings.

Some variables are just conveniences to simplify subsequent definitions.

Some string variables, such as `%tlroot%` and `%version%`, can be used outright because they are already set when the launcher starts parsing the ini file. Environment variables, *e.g.* `%appdata%` or `%UserProfile%` can also be used outright. See the package documentation for the full list.

First published in *TUGboat* **38**:2 (2017), pp. 193–196.

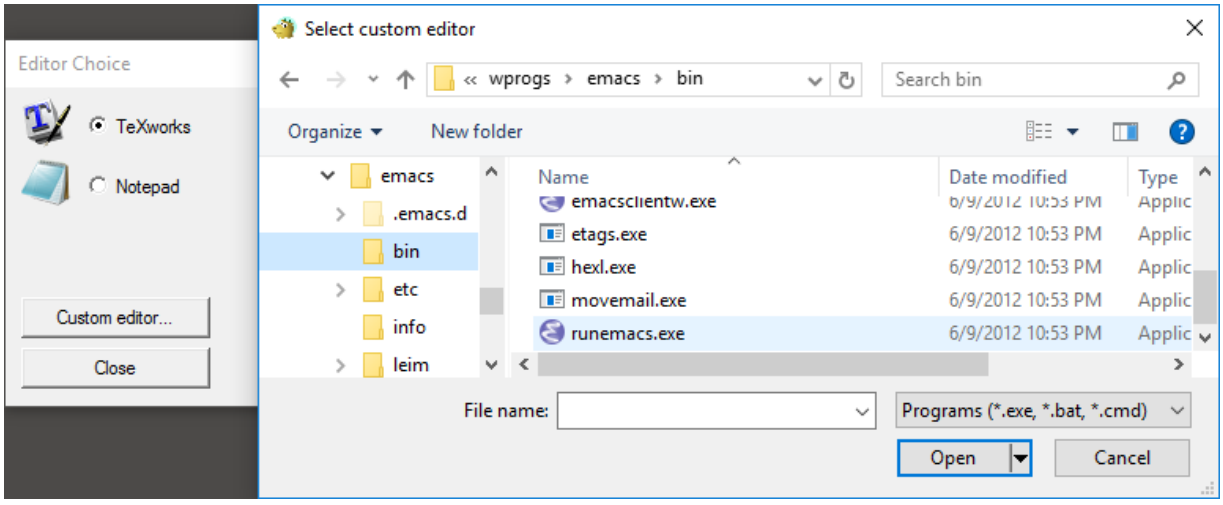


Figure 2. Selecting a custom editor

A few example string definitions:

```
[Strings]
TLNAME=TeX Live %VERSION%
; tlaunch configuration directory
TLCONFIG=%userprofile%\.texlive%VERSION%\tlaunch
TLSCRIPTS=%tlroot%\scripts
POST_CONFIG=%TLSCRIPTS%\post_config.cmd
; optional announcement text
ANNOUNCE=TeX Live Launcher with extras
```

2.2 Filetype associations

In Windows, the association of a filename extension with a program is indirect: an extension is associated with a filetype and a filetype is associated with a command. An example of a filetype definition in the ini file:

```
[FT:TL.TeXworks.edit.%VERSION%]
COMMAND="%tlroot%\bin\win32\TeXworks.exe"
EXTENSIONS=.tex .cls .sty
```

The name of the ini file section consists of the filetype name with an 'FT:' prefix. When a file with a listed extension is double-clicked in a file manager, Windows will run COMMAND with the (quoted) filename appended. If a more complex command is required, *e.g.* with parameters coming after the filename, the section can define a more complex command-line with a SHELL_CMD entry.

2.3 Menus and buttons

The ini file can contain a buttons section and sections for menus, with the latter indicated by an MN: prefix to the section name. Within the section entries, the key is the string to be displayed and the value is the action to be taken.

In the case of a button, the display string is put underneath the button (see figure 1). The launcher tries to find a suitable icon to place on the button itself, but has a fallback icon if it cannot find anything. This fallback icon is used for the Quit button in figure 1.

A few examples:

```
[MN:File]
Browse installation=explorer.exe "%tlroot%\.."
Quit=FU:quit

[MN:Viewers]
PostScript Viewer=FT:TL.PSView.view.%VERSION%
DVI Viewer=FT:TL.DVIOUT.view.%VERSION%

[MN:Documentation]
LaTeX Introduction=SO:%tlroot%\...\lshort.pdf
FAQ=SO:%tlroot%\...\newfaq.pdf

[Buttons]
LaTeX Editor=FU:default_editor
Select default editor...=FU:editor_select
Quit=FU:quit
```

The value, which is the associated action, can take several forms:

- No prefix: a command to be executed.
- With a prefix FT:, the associated action is the COMMAND of the indicated filetype, which should be defined earlier in the file.
- Prefix SO: (shell object) meaning in this case a file or URL that Windows should know how to open.
- Prefix SC: indicates a script object defined earlier in the ini file; see the package documentation.

- Prefix `FU:` indicates a predefined function; see the package documentation.

2.4 The General section

The most important options in this section replicate options from the TEX Live installer:

`FiletypesAllowed` values are `none`, `new` (default) and `overwrite`
`searchpathAllowed` values are `0` and `1` (default)

Both entries and the section itself are optional. For example:

```
[General]
FILETYPES=new
SEARCHPATH=1
```

3 Forgetting

The launcher has functions to undo and redo configuration, which can be assigned to menu items. However, the installation may not be under the user's control and may no longer be around when the configuration is to be cleared out.

Therefore, the launcher creates a so-called forgetter as part of its first-time initialization. This forgetter consists of a copy of the launcher and a modified copy of the configuration file, both placed under the user's profile. This copy knows from its location that it is intended to run as forgetter and not as launcher.

4 Scripts

The launcher can run scripts and command-line utilities, and display their output in a window. The ini file can specify scripts for *e.g.* supplemental initialization and cleanup (see section 2.1). Section 6.1 shows some examples. It is also possible to assign scripts to menu entries and to buttons. More about scripts is in the package documentation.

5 Launcher-based installations

The 2017 TEX Live installer offers the option of creating a launcher-based installation, as an alternative to creating menu shortcuts. If this option is selected, then no path adjustment is done and no filetype associations are created by the installer itself. The installer invokes the launcher with a special option to 'install' itself, *i.e.* to create a start menu shortcut and an uninstaller registry entry for itself. In case of a single-user install, it also performs a first-time initialization.

With such an installation, the TEX Live installer no longer has direct dealings with the Windows API, or with the Perl modules providing API access.

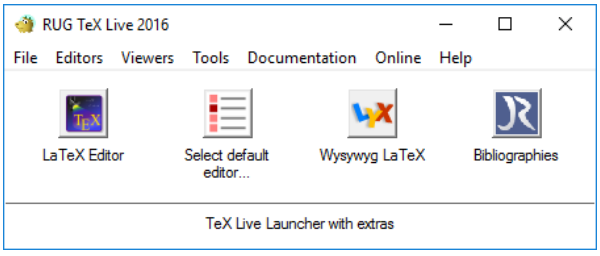


Figure 3. The TEX Live Launcher at the Rijksuniversiteit Groningen

For purposes of trying out the launcher, TEX Live includes a script `tlaunchmode` which can switch the installation between classic and launcher mode without reinstalling TEX Live.

6 The launcher at the Rijksuniversiteit Groningen

Workstations at our university are mostly centrally managed. Typically, users have a centrally managed Start menu on their Windows workstation. The IT people put the TEX Live Launcher in this menu, so users are just one click away from starting to use TEX Live.

Settings are centrally backed-up on logout and restored on login. So a user's desktop looks very similar, whatever physical workstation [s]he works on. This same desktop is also available remotely. In addition, users have a network share for storing their own files. This share is also available from any workstation on which they log in.

6.1 Additional software

The additional programs at our university include:

- More editors: TeXnicCenter and TeXstudio. Both offer extensive assistance in editing math.
- The PDF viewer SumatraPDF. This viewer provides source–PDF synchronization for TeXnicCenter, which has no built-in PDF viewer.
- The Java-based bibliography manager JabRef.
- The `epspdf` GUI with bundled single-file Tcl/Tk runtime (`https://ctan.org/pkg/epspdf`).
- The pseudo-WYSIWYG LyX LaTEX editor.

There are menu items for additional documentation, such as the LaTEX classes for the university house style. Controls for the TEX Live Manager and for uninstalling TEX Live itself are omitted, since those tasks are reserved for the maintainer of the installation.

All these programs were installed on a scratch system and from there copied into the TEX Live installation tree. 'Installed' this way, most of them run more or less ok from their new location.

However, some fixes were desirable and were implemented via a postconfig script (see section 2.1):

TeXnicCenter While TeXnicCenter can autoconfigure itself nicely for MiKTEX, it asks TEX Live users a series of questions about what is where. To spare users those questions, I wrote a vbscript which emulates the MiKTEX autoconfiguration for TEX Live, and which is invoked by the postconfig script.

TeXstudio This editor by default checks at startup whether there is a new version. The postconfig script turns this option off in an existing or newly-created TeXstudio configuration file.

TEXworks borrows some dictionaries from TeXstudio.

SumatraPDF This PDF viewer also tests for updates, which are dealt with in the same way as for TeXstudio. It also requires a registry setting to specify that this is *not* a portable installation, and it should store its settings under the user's profile.

LyX First-time initialization can take a very long time. Therefore, a LyX user configuration directory has been prepared in advance. The postconfig script copies it to the user's profile.[2]

The launcher documentation contains a file `rug.zip` with slightly sanitized versions of the scripts and configuration files actually used at our university installation.

7 Problems

7.1 Non-roaming filetype associations

In a standard Roaming Profiles setup, filetype associations do not roam. I plan to add an option to the launcher to restore missing filetype associations on login. This is not a problem with the centrally-managed desktops at our university. On the other hand, on those centrally-managed desktops some filetype associations are preempted and cannot be permanently changed. This includes PDF files.

7.2 Search path

Another problem associated with our desktop management software is that programs ignore the user search path.

This is not a problem for software started from the launcher.

Some programs do not absolutely need TEX Live on the search path. Others, such as TEXworks and the DVI- and PostScript viewers included in TEX Live, are invoked via a wrapper which takes care of the search path. But for TeXstudio I had to provide a wrapper myself to take care of the search path.

And then the university offers software such as R and WinEdt which also need LaTEX on the search path, but which are not under my control; the IT department has to handle these.

7.3 Uninstalling

The third problem I want to mention is uninstalling under Windows 10. This is not specific to the launcher.

There are two ways to give a user access to an uninstaller:

- Via a Start menu item. However, Windows 10 may somehow decide not to display such an item.
- Via an uninstaller registry key. This way, it will show up in Settings / Apps / Apps & features. However, Windows may decide to pop up a User Account Control (UAC) prompt even if it is a user install. Still, a user-installed launcher *can* be uninstalled via right-clicking its icon under the Start menu, or from within the launcher itself.

8 Finding out more

Earlier, I mentioned the tlaunch manual. If you have a fully updated 2016 or a later installation of TEX Live with Windows platform support, then you should have the tlaunch binary and documentation on your system. But you can also visit its CTAN directory at `https://ctan.org/pkg/tlaunch`.

For experimentation, you can run the script `tlaunchmode` mentioned at the end of section 5. With this script you can switch an existing TEX Live installation to launcher mode and back.

Notes

1. If during installation non-default options are selected for file associations or path adjustment, then a second, modified copy will be written to `texmf-var/web2c`.
2. There is also a shared LyX configuration file which had to be patched, but this is not a task for a per-user postconfig script.

Siep Kroonenberg
Groningen
The Netherlands
siepo (at) cybercomm dot nl

updmap and fmtutil — past and future changes (or: cleaning up the mess)

Abstract
This article serves first as an introduction to two of the central utility programs in any TEX Live installation, updmap and fmtutil, describing the general functionality as well as the syntax of the configuration files. In addition, we report on changes that we have carried out over the last few years relating to the operation mode. These changes include switching to multiple configuration files, and the user-mode versus system-mode changes to be introduced in TEX Live 2017. Last but not least, we close with a list of best practices to help guide users.

If you only want to know how best to install fonts (or formats) and are not particularly interested in the details, jump to Section 5.

1 Introduction

Two central utility programs in any TEX installation are updmap, responsible for creating font maps for various programs, and fmtutil, responsible for (re)creating format dumps.

For many years the venerable shell scripts by Thomas Esser were used on Unix-like systems with only minimal changes. For Windows, TEX Live used binary programs developed independently. Having two independent implementations hindered development of new features. Thus, some years ago we started rewriting them in Perl: first updmap (TEX Live 2012), and later fmtutil (2015).

With the rewrites in place, the first new feature added was already a considerable change in internal behavior: While the original shell scripts used a single configuration file, the new versions read configuration files on a per-tree basis. This helped users preserve their configuration across TL upgrades, and gave OS distributions better ways of integration into their respective packaging infrastructure.

With TEX Live 2017, we will go further and eliminate the biggest source of confusion: Users invoking the scripts in the so-called *user mode* (in contrast to *system mode*), thus generating local configuration files shadowing the global ones. The origin of this confusion is the widespread misinformation to call simply updmap (fmtutil) when the available fonts change.

TEX Live 2017 and later disable calls to updmap and fmtutil without an explicit mode request. This means that users who unknowingly call them will get a warning message — and hopefully afterwards will use the right mode.

1.1 Layout of the article

Section 2 will start with an explanation of the functionality of the scripts and how they fit into a TEX (Live) installation. While the general functionality of these scripts will be similar in other TEX distributions, some options described here are probably not available in other installations. In this section we also introduce the original system and user modes.

Section 3 describes the changes introduced with multiple configuration files, and explains how this can be used in single and multi-user environments.

Section 4 introduces the changed operational mode introduced in TEX Live 2017.

Section 5 has recommendations and best practices for dealing with local fonts and formats.

A running example for the installation of the MathProII fonts will exhibit the usage changes.

2 Functionality of updmap and fmtutil

Although updmap and fmtutil are central to TEX operations and are automatically executed on many occasions, both scripts have remained relatively mysterious and are often misused.

2.1 updmap

Many of the fonts shipped in a TEX system are PostScript Type 1 fonts. The original TEX does not know anything about this (or any glyph) font format; it only uses the metrics from TFM files. The output drivers on the other hand need to know how TFM names map to glyphs.

First published in *TUGboat* **38**:2 (2017), pp. 188–192.

Typical output drivers are

pdf(la)tex the TEX engine extended with direct PDF output. Since producing PDF needs the actual fonts, pdftex is also an output driver.

dvips the classical output driver. TEX engines can produce DVI (DeVice Independent) files, which can be translated to PostScript (or other) formats. To do this, the fonts have to be embedded.

(x)dvipdf(m(x)) the family of DVI-to-PDF converters. Instead of going to PostScript first, these programs support direct translation of DVI into PDF. XƎTEX uses one of these in the background. Japanese users often use dvipdfmx, since it has good support for Japanese fonts.

xdvi online X11 display program, which of course needs access to the fonts to render the glyphs.

These output drivers have supported font mapping in slightly different ways, changing over the years, and here is where updmap comes into the game: It reads a list of specifications, and creates configuration files in the needed formats.

What does updmap *do?*

Font definitions are necessarily a complicated beast in the TEX world; many components have to play well together for the final document to contain the correct fonts. Here is an overview of the main items necessary to understand updmap:

font definition maps a TFM file name to an external font (font name and file name), with optional additional transformations. A simple example:

```
eufm10 EUFM10 <eufm10.pfb
```

which says that the TFM name eufm10 should be resolved by a font internally named EUFM10, which is defined in the file eufm10.pfb. Far more complex font definitions are possible, catering to different encodings and more, but the basic purpose of mapping a TFM to an external font always remains.

font map file is a file of font map definitions, normally collecting together related fonts from a package. The above definition for eufm10 is contained in euler.map, which contains all the Euler-related font definitions.

updmap config file lists the font map files, with additional specifications concerning bitmap vs. outline fonts, as well as a few settings for updmap itself (details in the next section). Continuing our example, in a normal TEX Live installation the font map file euler.map is listed in texmf-dist/web2c/updmap.cfg:

```
Map euler.map
```

generated files Finally, updmap generates configuration files in various formats (see above).

Output drivers don't have (or need) the slightest idea that updmap and the related intermediate files even exist; they only read the ultimately-generated configuration file to determine which fonts are available. This means that if, somewhere in the middle, one of the steps fails or is incorrect, the output will probably not have the right fonts.

Configuration of fonts in updmap.cfg

The central configuration file for updmap is (always) named updmap.cfg. In former times, only the first one found by the Kpathsea library was used, but now all updmap.cfg files are read (see below). Each updmap.cfg can contain the following items:

1. Empty lines, comments beginning with '#'; these are ignored.
2. Map directives, in one of the forms:

   ```
   Map foo.map
   MixedMap bar.map
   KanjiMap baz.map
   ```

   Map is used for fonts that are available only in PostScript Type 1 format; MixedMap is for fonts where both Metafont and PostScript variants are present; and KanjiMap is for creating the special Kanji map file.
3. updmap configuration lines, of the form

   ```
   <settingName> <value>
   ```

   with the following setting names and values (* indicates the default):

   dvipsPreferOutline values *true, false
   Whether dvips prefers bitmaps or outlines, when both are available.

   dvipsDownloadBase35 values *true, false
   Whether dvips includes the 35 standard PostScript fonts in its output.

   pdftexDownloadBase14 values *true, false
   Whether pdftex includes the 14 standard PDF fonts in its output.

   pxdviUse values true, *false
   Whether maps for pxdvi (Japanese-patched xdvi) are under updmap's control.

   (ja|sc|tc|ko)Embed, jaVariant values strings
   Controls kanji font embedding for Japanese (ja), Simplified Chinese (sc), Traditional Chinese (tc), and Korean (ko).

   LW35 values *URWkb, URW, ADOBEkb, ADOBE
   Controls which fonts are used for the 35 standard PostScript fonts.

The ..Embed and the jaVariant settings were added to the TEX Live implementation recently, and might not be supported in other TEX distributions.

2.2 fmtutil

In the years long ago, when memory was scarce, computers slow, and Knuth went forth to create the most advanced typesetting system, he devised a way to speed things up and at the same time conserve space: format dumps. This is not the place for details but in short, you can think of them as dumps of the state of the program (TEX, Metafont, ...) after a (slow, painful) initialization, which can be easily and quickly loaded and used as a starting point for actual typesetting and font design work.

When there was only one TEX program and one Metafont program, managing these dumps was a simple task, but over time the situation grew more complex: more programs, more formats, various additions for internationalization. Nowadays, we're at a point that people often do not know what is going on when a *formats are rebuilding* message appears.

What does fmtutil *do?*

Written long ago by Thomas Esser for his teTEX, fmtutil supports specifying the available format in a line-based configuration file, and for rebuilding them in various ways. The script has served the TEX community for many years. The shell script mentions a first change in 2001, but the script is much older than that (considerably predating TEX Live).

2.3 Configuration of fonts in fmtutil.cnf

fmtutil is a rather friendlier colleague than updmap, with no need for all the complicated layers of definitions. The configuration files for fmtutil, named fmtutil.cnf, define the formats which can be made. The most commonly used format is LaTEX, but there are many more, some of which are quite esoteric (e.g., utf8mex).

Each format definition is on exactly one line, and consists of four parts:
<fmtname> <engine> <hyphenfile> <options>
Let us look at two examples from TEX Live:

```
aleph aleph - *aleph.ini
latex pdftex language.dat
           -translate-file=cp227.tcx *latex.ini
```

The first one defines the format aleph, the second one the format latex. (The second is broken across lines only for Maps; in the actual source file, it's all on one line.)

name aleph, latex — the first item in a format definition is the format name, which (usually) coincides with the program name.

engine aleph, pdftex — the second item defines the base engine, the program that is run to load the definitions and dump the image. As shown, sometimes the format and the engine have the same name. For the LaTEX format, TEX Live has used the pdfTEX engine for many years.

hyphenfile -, language.dat — the third item specifies a file name for hyphenation pattern definitions, or a literal - to indicate that no patterns are used.

options — the rest of the line comprises command line arguments passed to the engine. In the aleph line we see that only one file is passed to the engine, while in the latex case we also pass an additional option.

As specified on its own command line, fmtutil reads fmtutil.cnf, invokes some or all of the engines with the respective options in turn, and puts the resulting dump files in the right place so that the engine can load the dump.

2.4 Previous behavior and system mode vs. user mode

The original shell scripts read only one configuration file, found by searching with Kpathsea. This is the very same method TEX uses to find files when they are read (e.g., via \include) To cater for user-supplied font maps, the original updmap program allowed for enabling and disabling, adding and removing individual entries from the configuration file.

While this approach works nicely in a single user installation where the user has complete control over all files, in a multi-user setting it would be chaos if users changed a system-wide configuration file, adding their private fonts. Thus, soon after their inception, Thomas Esser added an additional *system mode* to these scripts, distinguished from the normal invocation style in *user mode*. The only difference between user mode and system mode is *where* generated files are saved: In user mode this was the directory defined by the Kpathsea variable TEXMFVAR, while in system mode it was TEXMFSYSVAR.

System mode was specified by invoking the program under the name updmap-sys (fmtutil-sys), while user mode was the default.

This was the state of affairs for more than a decade. The advantages of this system were that all configurations were contained in a single file, and the operation mode was easy (easier?) to understand.

In my case, as I had purchased the MathProII fonts, every year and on every computer I used I had to manually disable the open-source clone enabled by default in belleek.map, add the necessary map file for the MathProII fonts, and run updmap. While this is not much to do, it is easy to forget and error-prone.

3 Per tree configuration

With the Perl reimplementation of the scripts we have also switched to a different way of handling configuration files: the two programs now read not just a single configuration file, but *all* configuration files found, in a stacked manner, meaning that files read later can override parameters from those read earlier. *Override* here means the following: disabling a map that is enabled in a lower level configuration file, and changing settings from a value set in a lower level configuration file.

To see which configuration files will be used, these two commands will output the list of all configuration files used by the two programs:

```
kpsewhich -all updmap.cfg
kpsewhich -all fmtutil.cnf
```

This new method allows configuration of available fonts and formats to be put in the same tree where the respective fonts or formats are installed. Formerly, activation of a map file or format would not survive (re)installing a release of TEX Live. Now, local fonts can be installed under TEXMFLOCAL, and listed in TEXMFLOCAL/web2c/updmap.cfg, and they will automatically be picked up across updates.

Similarly, users can have personal fonts or formats without needing to maintain a copy of the system's updmap.cfg or fmtutil.cnf.

3.1 Default locations searched

By default, updmap and fmtutil check the following directories for updmap.cfg and fmtutil.cnf, in the order given.

User mode only:	TEXMFCONFIG/web2c
	TEXMFVAR/web2c
	TEXMFHOME/web2c
Both user and system modes:	TEXMFSYSCONFIG/web2c
	TEXMFSYSVAR/web2c
	TEXMFLOCAL/web2c
	TEXMFDIST/web2c

with these default values those variables:

TEXMFSYSCONFIG	TL/YYYY/texmf-config
TEXMFSYSVAR	TL/YYYY/texmf-var
TEXMFDIST	TL/YYYY/texmf-dist
TEXMFLOCAL	TL/texmf-local
TEXMFHOME	~/texmf
TEXMFCONFIG	~/.texliveYYYY/texmf-config
TEXMFVAR	~/.texliveYYYY/texmf-var

Making use of this information, let's continue the previous example of the MathProII fonts. As mentioned above, TEX Live ships the free Belleek fonts which use the same TFM names; thus, we have to disable belleek.map and add mtpro2.map:

1. Put the MathProII files, including mtpro2.map, in TEXMFLOCAL.
2. Edit TEXMFLOCAL/texmf/web2c/updmap.cfg:
   - ☐ disable Belleek by adding
     `#! Map belleek.map`
   - ☐ enable MathProII by adding
     `Map mtpro2.map`
3. Run updmap-sys.

Now, when I update my TEX Live installation from one year to the next *no* additional work is needed: updmap find the local configuration file, duly disabling the one map and activating the other.

Similarly, these per-tree configuration files have brought considerable simplification for distributors like Debian (indeed, this was the original reason why I implemented this feature).

4 Explicit user mode in tl 2017

4.1 What was the problem?

Let's suppose a user wants to add a private font to the TEX setup (as I had to do during my studies, when I purchased the Lucida fonts for writing my thesis). The steps were these:

- ☐ Copy updmap.cfg into TEXMFHOME;
- ☐ add the additional map entries to it;
- ☐ run updmap.

In itself this was not a problem. The problem comes when the fonts on the system side change (because of an update or addition of new font packages): The user had to re-execute these steps, every time. Not doing so would leave the user with outdated information; in the worst case (but unfortunately a very common case!), some font definitions would no longer be correct, and thus output files would be broken.

The reason was mentioned above: The configuration files for the output drivers generated by updmap in the user's home directory override the ones in the system directory.

We might hope for users to know about this problem, but unfortunately the Internet is full of instructions on how to install fonts for LaTEX, and the typical recommendation is to call updmap, and not updmap-sys. From my experience as the maintainer of the TEX Live packages in Debian, as well as from the TEX Live mailing lists, I can report that this is the single most common point of failure.

That is, most users were simply unaware that calling updmap (as is, thus in user mode) creates copies of configuration files which will *never be updated* unless the user calls updmap again; system changes in the meantime are immaterial.

For fmtutil the problem is the same: Format dumps would remain in the user's home directory and never be updated. As a glaring example, I recall a Debian bug report where a user had called fmtutil once, and *years* later some LaTeX packages stopped working, because he still used the format dump from years ago, all unknowing.

4.2 New operation mode

For TeX Live 2017, we (that is Karl Berry and I) decided to try to get out of this interminable chaos once and for all. Thus, from now on *user mode* cannot be invoked by calling updmap or fmtutil as is; to activate user mode, it's now required to give the option -user, or call the separate scripts updmap-user or fmtutil-user. To summarize:

System mode is invoked by using updmap-sys or fmtutil-sys, or by giving the -sys option.
User mode is invoked by using updmap-**user** or fmtutil-**user**, or by giving the **-user** option.

Calling updmap or fmtutil without -sys or -user now results in a fatal error, with a link to an explanatory web page.

Our hope is that this will prevent some (perhaps many) users from hurting themselves by unintentional switching to user mode. Furthermore, by introducing this new behavior we are explicitly invalidating plenty of documentation on the web that we know to be wrong, and force users to make a conscious decision. We will see next year how it has worked out!

5 Best practice and use cases

There is probably only one thing we should write here, and if you take one thing from this article, it should be this one:

Use system mode.

Anything else will very likely cause trouble. One might ask, so why didn't we abolish user mode completely? Indeed, we pondered this, but firstly, it would be a radical step after so many years, and secondly, there remain rare cases where user mode is needed; see the following use cases.

5.1 Use cases

The following use cases are also listed on a TUG page (tug.org/texlive/scripts-sys-user.html); the scripts refer to this same page in case of missing mode specifications.

Single user computer — add fonts
One of the most common cases: One user, one computer, TeX Live is installed system-wide, and fonts should be available to all (1) users of the machine:

- ☐ put the fonts into TEXMFLOCAL according to the TDS (tug.org/tds);
- ☐ enable the font map(s) in the file TEXMFLOCAL/web2c/updmap.cfg;
- ☐ run (once) updmap-sys (no options needed).

Future (re)installations of TeX Live will pick up these local fonts automatically.

Multi-user computer — add system-wide fonts
A common need in a department or company with organization-specific fonts, which all users should have access to: This case is handled exactly like the previous case, without any changes.

Multi-user computer — private user fonts
This is the only case where user mode is required: A computer with multiple users, but some fonts are private to specific users. Here we cannot install the fonts system-wide, as other users would gain access to them. Thus TEXMFHOME is used instead of TEXMFLOCAL, and updmap-user is run:

- ☐ Put fonts into TEXMFHOME, following the TDS;
- ☐ enable the font map(s) in TEXMFHOME/web2c/updmap.cfg;
- ☐ run (once) updmap-user.

A repeated warning is necessary here, because this is the prime case of misbehavior we have seen: After doing this, changes in the font setup of the system are *invisible* until updmap-user is rerun. Thus, we recommend running it regularly, e.g., from Unix cron, to make sure no discrepancy creeps in between the fonts as actually installed and those registered in the per-user updmap.cfg.

Single user computer — additional formats
While it is uncommon for users create their own formats, in principle the procedure is the same as with updmap. In most cases, the additional formats need not be private, so following the first use case above is suggested:

- ☐ adjust TEXMFLOCAL/web2c/fmtutil.cnf
- ☐ run (once) fmtutil-sys (no options needed).

5.2 Switching back to system mode

Last but not least, here is how to switch back to system mode if by chance one has called updmap or fmtutil in user mode. This is never done automatically, and (at least for now) there is no interface to the two programs to allow easily switching.

To switch back to system mode, what has to be done is to remove the following directory trees (after backing them up, of course):

- for updmap: TEXMFVAR/fonts/map
- for fmtutil: TEXMFVAR/web2c

where under normal circumstances, TEXMFVAR is ~/.texliveYYYY/texmf-var.

6 Conclusion

We hope that the changes made over the last years have made these programs easier to use, and a bit more protective for the casual user. But one should not forget that they are central configuration programs for TEX, so messing around with them always bears some risk.

Final exhortation: Use System Mode!

Norbert Preining
Accelia Inc., Tokyo, Japan
norbert (at) preining dot info

Privacybeleid

Nederlandstalige TEX Gebruikersgroup

De NTG werkt met persoonsgegevens. De NTG vindt het belangrijk dat deze gegevens op een zorgvuldige en veilige manier worden verwerkt. In dit document staat alles over de manier waarop persoonsgegevens worden verzameld en hoe daarmee wordt omgegaan.

Dit privacybeleid zal soms gewijzigd worden door bijvoorbeeld wetswijzigingen of omdat bepaalde procedures binnen de vereniging wijzigen. Het is daarom raadzaam dit document periodiek te raadplegen.

Dit privacybeleid is opgesteld in het kader van de AVG, die vanaf 25 mei 2018 van kracht is. Daarmee wordt ook voldaan aan de vanaf die datum geldende EU GDPR.

Beheerder en verwerker van de persoonsgegevens

De Nederlandstalige TEX Gebruikersgroep (NTG) is de gegevensbeheerder en -verwerker van de persoonsgegevens van de leden van de vereniging. De NTG is te benaderen via de contactpagina (https://www.ntg.nl/lug/nl.html) op de website.

Bewaarde gegevens

De NTG maakt onderscheid tussen noodzakelijke, optionele en overige persoonsgegevens.

Noodzakelijke persoonsgegevens in de ledendatabase

Voor administratieve doeleinden en om als vereniging te kunnen functioneren vereist de NTG de volgende persoonsgegevens van haar leden:

1. uniek lidnummer, door de NTG toegekend, noodzakelijk voor het correct functioneren van de administratie
2. lidmaatschapstype
3. achternaam en voorletters, voor alle communicatie
4. postadres, voor het bezorgen van fysieke post zoals ons tijdschrift Maps en de factuur voor de contributie, indien er geen e-mailadres of factuuradres bekend is
5. e-mailadres, alleen van huidige leden van het bestuur
6. telefoonnummer, alleen van huidige leden van het bestuur
7. jaar van aanvang lidmaatschap
8. taal waarin communicatie wordt gesteld (Nederlands of Engels)

Optionele persoonsgegevens in de ledendatabase

Naast bovenstaande zijn er optionele velden in het aanmeldingsformulier, die de NTG bewaart indien ingevuld:

☐ voornaam en gewenste aanhef (de heer/mevrouw), voor alle communicatie
☐ affiliatie
☐ e-mailadres, voor het bezorgen van digitale post via de ntg-leden e-mail mailing lijst en de factuur voor de contributie, indien er geen factuuradres bekend is
☐ telefoonnummer (voor overige leden)
☐ homepage

- ☐ geboortedatum, alleen voor controle bij gecombineerd lidmaatschap met seniorenkorting van de TEX User Group
- ☐ factuuradres (als postadres of als e-mailadres), voor het bezorgen van een factuur voor de contributie
- ☐ additionele informatie voor op de factuur, bijvoorbeeld een administratief nummer voor de ontvanger van de factuur
- ☐ bezoekadres, indien het postadres een postbus is

Overige persoonsgegevens

Voor bijeenkomsten en jaarvergaderingen bewaart de NTG presentielijsten met daarop de namen van de aanwezige en afgemelde leden. Voor internationale bijeenkomsten worden ook affiliatie, plaats en land, en e-mailadressen bewaard.

De NTG is de beheerder van de wereldwijde TEX local user group (LUG) database. Hierin bevinden zich namen, postadressen, e-mailadressen, en telefoon- en faxnummers van (bestuursleden van) de NTG en haar zusterorganisaties in het buitenland.

In de financiële administratie bevinden zich de bankafschriften van de vereniging met daarop de tenaamstelling en nummer van rekeningen waarvan gelden zijn ontvangen. De NTG bewaart geen bankrekening-informatie in de ledendatabase. De financiële administratie wordt zeven jaar bewaard.

Inzage en correctie van persoonsgegevens

Elk NTG lid kan te allen tijde opvragen welke persoonlijke gegevens de NTG bewaart. Dat kan door een verzoek daartoe te sturen naar de penningmeester van de vereniging.

Leden kunnen eveneens via een bericht aan de penningmeester persoonsgegevens in de ledendatabase laten corrigeren.

Verwijdering van persoonsgegevens

Bij opzegging van het lidmaatschap worden de gegevens in de ledendatabase verwijderd op de einddatum van het lidmaatschap.

Verzoeken gebaseerd op het recht op vergetelheid kunnen worden ingediend bij de penningmeester.

Klachten

Eenieder heeft het recht om een klacht in te dienen bij de Autoriteit Persoonsgegevens, als hij of zij van mening is dat de NTG niet op de juiste manier met zijn of haar gegevens omgaat. Dit kan via: Melden verwerking persoonsgegevens (https://autoriteitpersoonsgegevens.nl/nl/melden/melden-verwerking-persoonsgegevens).

Locatie van gegevens

Ledendatabase

De penningmeester van de NTG is de beheerder van de ledendatabase. Bij overdracht van deze taak wordt de database overgedragen naar de nieuwe verantwoordelijke en worden alle gegevens gewist bij de aftredende penningmeester.

De ledendatabase bevindt zich uitsluitend op de NTG server. Alleen de penningmeester heeft toegang tot de ledendatabase middels een beveiligde login met wachtwoord. Van deze database bestaan computer-backups op de privé computer van de penningmeester en bij de partij die deze server host (Elvenkind B.V.). De ledendatabase wordt niet als een op papier afgedrukte lijst bewaard.

Voor verzending van fysieke poststukken worden door de penningmeester adreslabel-bestanden gegenereerd die worden doorgegeven aan het bestuurslid belast met het verzenden, die deze bestanden vernietigt na gebruik.

NTG Server

De NTG.nl server is een virtuele hosting server die de vereniging deelt met de ConTEXt Group en Boekplan. Naast de NTG website (http://www.ntg.nl) bevinden

zich op deze server tevens de NTG mailing lijsten. De beheerders hebben toegang tot de server via ssh, een beveiligde login met wachtwoord.

Website De NTG website is vrijwel geheel statische html, plaatst geen cookies, en bewaart geen gegevens van bezoekers met uitzondering van de IP-adressen die worden gerapporteerd in het webserver log. Er is geen webshop, via deze website worden geen producten of diensten verkocht. Bezoekers van de website worden niet gevolgd, hun bezoekgedrag wordt niet geanalyseerd; er vindt dus geen enkele vorm van profilering plaats. De webserver logs worden automatisch gewist na zes maanden.

Enkele website formulieren verzamelen tijdelijk (persoons)gegevens die via e-mail verzonden worden naar de verantwoordelijke persoon binnen de NTG, maar deze gegevens worden niet opgeslagen:

- ☐ Het aanmeldingsformulier voor lidmaatschap verzamelt informatie voor opname in de ledendatabase. Informatie wordt verzonden naar de NTG penningmeester.
- ☐ Het nieuws aanmeldingsformulier bevat naam en e-mailadres van de aanmelder. Informatie wordt verzonden naar de NTG secretaris en de NTG webmaster.
- ☐ Het subsidieaanvraag formulier bevat naam en e-mailadres van de aanvrager. Informatie wordt verzonden naar de NTG penningmeester.

Alle formulieren en het ‚alleen voor leden' gedeelte van de NTG website zijn beveiligd via https.

Toegang tot het ‚alleen voor leden' gedeelte van de NTG website is gekoppeld aan het abonnement van de ntg-leden mailing lijst (https://mailman.ntg.nl/mailman/listinfo/ntg-leden). Leden die zich voor deze lijst hebben afgemeld hebben ook geen toegang tot het ‚alleen voor leden' gedeelte van de NTG website.

In het historisch publicatiearchief (in PDF formaat) bevinden zich presentielijsten van eerdere bijeenkomsten en jaarvergaderingen.

Het publicatiearchief is online te bezoeken en daarmee zijn deze gegevens openbaar.

Via de LUG contact pagina (http://www.ntg.nl/lug/) is de LUG database te benaderen en kunnen correcties worden aangemeld.

Mailing lijsten De NTG beheert diverse digitale mailing lijsten. De lijst software gebruikt een database, deze bevat voor elke mailing lijst e-mailadressen en (eventueel) persoonsnamen van abonnees, alsook een archief van geplaatste berichten. Zowel de lijst van abonnees als het archief kunnen openbaar of beperkt zichtbaar zijn.

Enkele van deze mailing lijsten zijn voor intern gebruik door de vereniging:

Alle leden van de NTG worden bij aanmelding automatisch abonnee van de ntg-leden mailing lijst (https://mailman.ntg.nl/mailman/listinfo/ntg-leden). Leden van het NTG bestuur zijn automatisch abonnee van de ntg-bestuur mailing lijst. Leden van het NTG Maps redactieteam zijn automatisch abonnee van de ntg-maps mailing lijst. Leden van het NTG server beheerteam zijn automatisch abonnee van de ntg-server mailing lijst.

Voor deze mailing lijsten voor intern gebruik is online inzage van de lijst van abonnees en het archief alleen mogelijk voor abonnees van deze lijsten.

Daarnaast beheert de NTG ook diverse openbare mailing lijsten voor discussie en overleg met betrekking tot TEX gebruik en ontwikkeling, zoals de ntg-context en tex-nl lijst. Aanmelding als abonnee op deze mailing lijsten is op vrijwillige basis. De privacy-instellingen kunnen verschillen per lijst, maar gewoonlijk is de abonneelijst alleen zichtbaar voor abonnees, en het archief openbaar.

Gebruik van gegevens

De NTG gebruikt de verzamelde persoonsgegevens alleen om als vereniging te kunnen functioneren. De NTG stelt de verzamelde gegevens nooit beschikbaar aan derden, behalve:

- □ Wanneer dit wettelijk is vereist.
- □ Wanneer het lid via de NTG ook lid is van TUG (het NTG-TUG joint lidmaatschap). In dit geval worden de volgende gegevens aan TUG ter beschikking gesteld ter uitvoering van het TUG lidmaatschap: lidnummer, lidmaatschapstype, betaling van lidmaatschap voor het lopende jaar, aanhef, voornaam, voorletters, achternaam, postadres en e-mailadres.

In (financiële) jaarverslagen worden alleen geaggregeerde gegevens uit de ledendatabase gebruikt, zoals actuele en historische ledenaantallen. Daarmee is deze informatie dus anoniem gemaakt.

Functionaris gegevensbescherming

De penningmeester van de vereniging is de functionaris voor de gegevensbescherming, en rapporteert aan het bestuur.