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