The Debian TEX sub-policy

The Debian TEX mailing list <debian-tex-maint@lists.debian.org>

generated from Id

Abstract

This document provides a set of rules for the packaging of applications, fonts and input files related to TEX within the Debian GNU/Linux distribution.

Copyright Notice

Contents

Chapter 1

About this document

This document provides a set of rules for the packaging of applications, fonts and input files related to TEX within the Debian GNU/Linux distribution. It is still a in a draft state – some things might not yet be fully implemented, and others are advisable, but not strictly necessary. If in doubt, please ask on `debian-tex-maint@lists.debian.org`.

The latest copy of this document can be found in the `Debian-TEX-Policy` files in the `tex-common` package.

Chapter 2

Terms and Definitions

The following terms are used in this document:

TEX-related package Any Debian package that uses or provides parts of the TEX infrastructure, i.e. the TEX or METAFONT program or derivatives thereof, fonts or input files in a *TEXMF* tree, etc.

`tex-common` This package provides basic infrastructure and some configuration files for all TEX-related packages, including the `configuration update programs`.

Basic TEX packages A Basic TEX package is a Debian package that provides the basic infrastructure for TEX-related programs. It should provide sufficient functionality for typesetting most generated (La)TEX code, e.g. from docbook, debiandoc, or texinfo sources. Usually, the Basic TEX packages will be divided into an architecture-dependent and an architecture-independent package.

The arch-dependent package must provide at least one binary that is fully compatible with Donald E. Knuth's original TEX program, and it should provide the original TEX itself. The output formats `dvi`, PostScript and Adobe PDF must be available, either directly or by conversion of other output formats. The arch-independent package must provide at least the files necessary to create the formats for plain TEX and LATEX and the input files required by the LATEX distribution, as well as the Computer Modern fonts.

TDS The TEX Directory Structure, which describes file placement for TEX input files. The current version of the TDS is installed with this document as tds.pdf (`file:///usr/share/doc/tex-common/tds.pdf`) and tds.html (`file:///usr/share/doc/tex-common/tds.html`). The latest version of the TDS is available at `http://www.tug.org/twg/tds/`.

***TEXMF* tree** One directory tree, arranged according to the TDS

TEX input file A file that is meant to be used by a TEX-related program; technically any file that can be found by the `/kpathsea/kpse` library. This includes e.g. Type1 font files.

configuration update programs The configuration information from files provided by different TEX-related packages must be merged and made available in appropriate form to the various programs. This is usually done by scripts that write files into the *TEXMFSYSVAR* tree.

Currently, the configuration update programs provided by `tex-common` are: `update-texmf`, `update-fmtutil`, `update-language`, `update-updmap`.

Chapter 3

TEX packages for the impatient

- A package that only installs TEX input files, e.g. a new LATEX package, should install them in the *TEXMFDEBIAN* tree (`/usr/share/texmf/`) at the place indicated by the TDS, see tds.html (`file:///usr/share/doc/tex-common/tds.html`) and 'File searching and `libkpathsea` / `libkpse`' on page 9, and register them in the maintainer scripts, usually by calling `dh_installtex` in `debian/rules`
- Packages that add fonts, hyphenation patterns or formats, or want to change the basic configuration in `texmf.cnf`, need to follow the rules in 'Configuration update programs' on page 11 in addition to that.

Chapter 4

Meta-packages and dependencies

The TEX Live collection of basic and add-on TEX packages provides some meta-packages for the convenience of users.

Depending on the `texlive-*` metapackages is only acceptable for editors, IDEs and other tools which handle user-generated code. TEX add-on packages, as well as automated input generators etc., must instead depend on a list of individual texlive packages which are actually used. [1]

[1] This is, for example, required to be able to adapt dependencies of metapackages according to the users' needs.

Chapter 5

File Placement

This chapter describes the placement of TEX input files, so that they can be found by programs. Files that are not input files for TEX or related programs must not be put in a TEXMF tree (put them into `/usr/share/`*`package`* instead). As an exception, documentation files in plain text may be used inside a *TEXMF* tree, e.g. to explain the purpose of an otherwise empty directory.

5.1 File searching and `libkpathsea` / `libkpse`

File locations must follow the TEX Directory Structure, TDS. The TDS specification is available as tds.pdf (`file:///usr/share/doc/tex-common/tds.pdf`) and tds.html (`file:///usr/share/doc/tex-common/tds.html`), and the latest version of the TDS is available at `http://www.tug.org/twg/tds/`. It is a bug if a package only conforms to an outdated TDS version. It is a more severe bug, however, if it conforms to the current TDS version but does not make sure to depend on an appropriately recent version of the Basic TEX packages or `tex-common` (that supports this TDS version).

The Basic TEX packages must provide a mechanism for searching through *TEXMF* trees that allows different files to be found depending on the invoking program and the specified file format. The only existing implementation is the `libkpathsea` library. Unfortunately, it was not originally designed for use as a dynamic shared library. A rewrite is under way to create a `libkpse` library with proper API specification and ABI compatibility. For the time being, the Basic TEX packages can provide a shared library, and program maintainers can decide to use it, or to link statically against their own copy of the code.

For use in scripts, the Basic TEX packages provide the utilities `kpsewhich`, `kpsepath`, `kpsexpand`, and `kpsestat`.

5.2 Directory trees

The following *TEXMF* trees are defined, as outlined below:

1 `/usr/share/texlive/texmf-dist/`, referenced as *TEXMFDIST*

2 `/usr/local/share/texmf/`, referenced as *TEXMFLOCAL*

3 `/usr/share/texlive/texmf/`, referenced as *TEXMFMAIN*

4 `/usr/share/texmf/`, referenced as *TEXMFDEBIAN*

5 `/var/lib/texmf/`, referenced as *TEXMFSYSVAR*

6 `/etc/texmf/`, referenced as *TEXMFSYSCONFIG*

7 Any directories listed in the *TEXMFHOME* configuration variable in `texmf.cnf` or as an environment variable,

8 optionally user-specific directories for configuration files (*TEXMFCONFIG)* and generated files (*TEXMFVAR*)

The search order is from bottom up (files in *TEXMFHOME* taking precedence over files in *TEXMFMAIN) etc.*

The role of the trees *TEXMFMAIN* and *TEXMFDIST* in Debian parallels the usage in upstream TEX Live. Upstream uses *TEXMFMAIN* for the files that have to match the binary executables and *TEXMFDIST* for other TEX input files that are

replaced when a new texmf tarball appears; *TEXMFDEBIAN* is an additional tree where TEX add-on packages can put their files.

Debian packages generally install files in *TEXMFDEBIAN*, and may ship or create empty directories in the other trees, in accordance with Debian Policy. Configuration file handling in *TEXMFSYSCONFIG* is described below in 'Configuration files' on the facing page. Packages should take care to ignore *TEXMFHOME* in their maintainer scripts.

5.3 Generated files

Generated files should be created below *TEXMFSYSVAR* (or the user-specific variable directories, *TEXMFVAR*), with the subdirectory structure conforming to the TDS. Generated font files will either be created in each user's *TEXMFVAR* tree, or in the *VARTEXFONTS* tree[1]

An exception is the generated file `/etc/texmf/web2c/texmf.cnf`. Local administrators should not edit this file, as manual changes will be overwritten later on. Instead, configuration file snippets in `/etc/texmf/texmf.d` must be used.

5.4 Filenames and installation of alternative files

Packages may not install files with the same name as a file already installed in a *TEXMF* tree, unless both files are in subdirectories where they will only be found by different applications, as determined by the `--progname` or `--format` switches to kpsewhich.

There are two exception to this rule:

1 Basic TEX packages install their files into their *TEXMFDIST* directory and will usually contain files that are also in other basic TEX packages.

2 Packages that need newer versions of a file than already supplied by a basic TEX package and installed in *TEXMFDIST* can place them into *TEXMFDEBIAN*. Thus, the outdated file will be shadowed, and the new one is in effect.

The maintainer of the basic TEX package should be made aware of the problem [2] The package maintainer must make sure to follow new releases of the basic TEX packages and not continue shadowing a file that is newer than the version provided by the shadowing package.

The package must make sure that the newer version is backward-compatible, meaning it must not break compilation of any TEX document, and it should not change the output file. A change of the output file may be acceptable if an obviously buggy behavior is corrected, **and** if it had previously not been possible to easily fix this behavior in user's documents (or if the updated package and a possible fix in the document combined lead to a correct document).

Installing more than two versions of a file will most likely lead to confusion. Therefore, the possibility to shadow a file once should be enough, and the usage of `dpkg-divert` is discouraged.

It is also discouraged to use a file other than from the canonical source for that file, usually the CTAN network.

5.5 Documentation

Packages should make documentation available to `texdoc`. This can be done be either installing the files below `/usr/share/texmf/doc`, or by providing symlinks from subdirectories of that location to the actual documentation files. To allow partial parallel installation of different basic TEX packages, these always install their documentation files into `/usr/share/doc/`*`packagename`* and put symlinks into their respective *TEXMFDIST*.

A package must not install files into (subdirectories of) `/usr/share/texmf/doc`, which is a symbolic link to `/usr/share/doc/texmf`.

The entry points for documentation should have names that indicate what they document. Names like `manual.pdf` or `index.html` should be avoided, even if the directory name is unmistakable [3].

[1] Per default, this tree is located in the world-writeable directory `/tmp/texfonts/`, in order to allow automatic package builds to work without user directories. On multi user systems, the admin might want to change this to a persistent directory and set up proper permissions

[2] A wishlist bug on the shadowing package, blocked by an other wishlist bug on the basic TEX package, can help tracking these issues.

[3] This allows users to say `texdoc` *`packagename`* directly. Otherwise they will first have to find the right command line (e.g. `texdoc` *`packagename`*`/user.dvi`) using `texdoc -s` *`keyword`*

Chapter 6

Configuration

6.1 Configuration files

Files that are used to modify the behavior of executables must be treated as any other configuration file in a Debian package. However, files that are used to control the typeset output - the appearance of documents - need not be treated as configuration files. It is up to the maintainer of the package to decide which files make sense to be used for site-wide (as opposed to per-project or per-document) customization.

A typical case for a site-wide configuration file is a file that must be changed if a style file should use additional modules (installed, for example, into TEXMFLOCAL). Options that only control document output are rather used for a particular document or documentation project and should usually not be installed as a configuration file.

Note that `/etc/texmf/` is a usual TDS tree. Files can be put into appropriate TDS-conforming subdirectories (e.g. `/etc/texmf/fonts/map/`), but directories not specified in TDS (or added Debian-specifically in `tex-common`'s files in `/etc/texmf/texmf.d/`) are generally not searched for TEX input files and can be used by packages for configuration files that are not TEX input files (e.g. the files in subdirectories `fmt.d` or `hyphen.d`).

6.2 Configuration update programs

Configuration files in the TEX world come in two classes: stackable and unstackable. The first class means that the respective programs read *all* configuration files found, while in the later case only the top or first configuration file is used.

Stackable configuration files in TEX are `TEXMFTREE/web2c/texmf.cnf` (central configuration for TEX applications) and `TEXMFTREE/web2c/updmap.cfg` (font configuration), while unstackable configuration files are `TEXMFTREE/tex/generic/config/language.dat` (language support/hyphenation patterns for latex based formats), `TEXMFTREE/tex/generic/config/language.def` (the same for etex based formats), `TEXMFTREE/tex/generic/config/language.dat.lua` (the same for luatex based formats), and `TEXMFTREE/web2c/fmtutil.cnf` (for format definitions).

In Debian, by default the respective configuration files of the following trees are used: For `texmf.cnf`: TEXMFDEBIAN (the texmf.cnf file is a link to the one in TEXMFMAIN). For `updmap.cfg`:TEXMFDIST, TEXMFDEBIAN. For the unstackable configuration files the respective copies in TEXMFSYSVAR are used.

The stackable configuration files are either static (texmf.cnf) or generated automatically in the background without any need for configuration, since changes can be included in a higher order configuration file.

The non stackable configuration files plus the file `/etc/texmf/web2c/texmf.cnf` are generated by configuration update programs from configuration files in subdirectories of `/etc/texmf`. For all of them this is the only method of configuration.

Packages are free to add configuration items to the common configuration files, but they should not try to override configuration items that are supplied by other packages. Rather, shared configuration items should be supplied by the Basic TEX packages or any other package on which all involved packages depend, with a setting appropriate for all. If this is impractical, the involved packages must at least agree on the way different packages override other's settings[1].

The configuration update programs should be called without any options to allow for internal changes, e.g. of the directories where the generated files are placed.

[1] Note that in `texmf.cnf`, as well as in the sequence of multiple `texmf.cnf` files that are read, earlier entries override later ones.

Packages that changed `updmap.cfg` must call `updmap-sys` as detailed in 'Font configuration' on the current page. Packages that changed `language.dat` or `fmtutil.cnf` must call `fmtutil-sys` (see below). They must make sure to issue the `mktexlsr` command before this.

6.2.1 Font configuration

A package that provides PostScript Type 1 fonts for TEX should be usable with any Basic TEX Package. The recommended way to implement the configuration scheme described below is to use the `debhelper` program `dh_installtex` provided by `tex-common`. See `dh_installtex(1)` for usage details.

Description of manual font package setup

This section describes how `dh_installtex` manages font packages, and what packages need to do that want to do without it.

For the rest of this section, we'll assume we are dealing with a package named *package* that installs PostScript Type 1 fonts for TEX. *package* should fulfill the following requirements:

1 It should depend on `tex-common` but not on any Basic TEX Package, unless needed for another task than simply installing the fonts for TEX.

2 It should install the necessary map files (`.map` extension) below *TEXMFMAIN*`/fonts/map`. The precise location must conform to the applicable TDS version.

3 It should also obviously install other needed or useful files provided by upstream to use the fonts with TEX-related programs (`.pfb`, `.tfm`, `.enc`, `.fd`, `.sty`, documentation, etc.).

4 It should install one or more configuration files with names following the pattern `20`*name*`.cfg` into `/etc/texmf/updmap.d/`[2]. Such files will be later merged by `update-updmap` to form `/var/lib/texmf/web2c/updmap.cfg`, the effective configuration file for `updmap-sys`.

Exactly what to put in these files is documented in `update-updmap(1)`. Basically, they should contain the pseudo-comment:

```
# -_- DebPkgProvidedMaps -_-
```

as well as the usual `Map` and/or `MixedMap` lines that *package* needs to add to `/var/lib/texmf/web2c/updmap.cfg`.

5 It should install a file named `/var/lib/tex-common/fontmap-cfg/`*package*`.list` that contains a reference to every `.cfg` file from the previous step, one per line. For instance, if *package* installs `20foo.cfg` and `20bar.cfg` into `/etc/texmf/updmap.d/`, the contents of `/var/lib/tex-common/fontmap-cfg/`*package*`.list` should be:

```
20foo
20bar
```

This *package*`.list` file must be shipped in the `.deb`, so that when *package* is removed (not necessarily purged), *package*`.list` disappears from `/var/lib/tex-common/fontmap-cfg/`.

6 It should run:
- in *package*`.postinst`;
- when *package*`.postrm` is called with `remove` or `disappear` as its first argument

the following commands in this order: `update-updmap --quiet`, `mktexlsr` and `updmap-sys`.

Since `mktexlsr` and `updmap-sys` are provided by the Basic TEX Packages, *package*`.postinst` has to ensure that they are only called when found in `$PATH` (unless *package* depends on the Basic TEX Packages for some reason). In *package*`.postrm`, the same considerations must be taken into account, with the addition that `tex-common` (that provides `update-updmap`) can be unconfigured or even uninstalled.

Note that even when `tex-common` is configured, it cannot be assumed that `update-updmap`, `mktexlsr` and `updmap-sys` can be safely run whenever available, because they internally use `kpsewhich` which only works after the `libkpathsea` library in a separate package has been configured properly.[3] The following check can be used to determine whether `libkpathsea` is configured:

[2] Filenames starting with `10` are reserved for the Basic TEX packages. However, sorting order is actually only relevant for snippets for `texmf.cnf`, `fmtutil.cnf` and `language.dat`.

[3] However, `update-updmap` uses `libkpathsea` only in user-specific-mode. In system-wide mode, it doesn't matter whether `libkpathsea` is configured or not.

```
if kpsewhich --version >/dev/null 2>&1; then
    echo "kpsewhich is installed and libkpathsea is configured."
else
    echo "Either kpsewhich is not installed, or libkpathsea is not configured."
fi
```

A sample implementation of this scheme can be found in 'Sample code for font packages' on page 17, but the recommended way to implement this scheme is to use `dh_installtex`.

Rationale

The rest of this section explains the rationale behind the previous recommendations.

- The dependency on `tex-common` ensures that in *`package`*`.postinst`, `update-updmap` can be run and `texmf.cnf` is in a sane state, so that `mktexlsr` and `updmap-sys` can be run safely (if present and if `libkpathsea` is configured).
- The recommended order for running the programs `update-updmap`, `mktexlsr` and `updmap-sys` ensures that `updmap-sys` can locate the newly-installed files (in particular, the map files shipped by *package*), since `mktexlsr` is run before `updmap-sys`. It is also run after `update-updmap`, because `/var/lib/texmf/web2c/updmap.cfg` might have been created by `update-updmap`, although it more probably already existed. And since it would be of no use to call `mktexlsr` before `update-updmap`, we recommend to run it after, just in case.
- Now, about the "magic comments" in `/etc/texmf/updmap.d/*.cfg` and the *`package`*`.list` file in `/var/lib/tex-common/fontmap-cfg/`. When that *package* is removed, but not purged, it has to make sure that its `update-updmap` configuration files in `/etc/texmf/updmap.d/` are ignored. Otherwise, any call to `updmap-sys` by an other package or the local admin would fail because it cannot find *package*'s map files. Besides, we want the `/etc/texmf/updmap.d/*.cfg` files to be conffiles (unless we really have no other choice), because then `dpkg` automatically handles upgrades while preserving user modifications for them. As a consequence, moving the `.cfg` files from *package* out of the way when it is removed is not an option. Moreover, the user would wonder where his configuration files have gone in such a case.

  The solution we chose was to add a little bit of logic into `update-updmap`, so that whenever it sees a `.cfg` file (let's call it `20foo.cfg`) that has the "magic comment", it actually includes its contents into `updmap.cfg` if, and only if:

  - it is up-to-date (which is assumed if `20foo.cfg.dpkg-new` doesn't exist in the same directory);
  - `20foo` appears on a line by itself in one of the `.list` files in `/var/lib/tex-common/fontmap-cfg/`.

  Additionally, that `.list` file should be named *`package`*`.list` if `20foo.cfg` comes from *package*, for simple reasons of tidiness.

  With this little mechanism in place, all the rest follows as expected:

  - When *package* is removed, but not purged, *`package`*`.list` is first removed by `dpkg` from `/var/lib/tex-common/fontmap-cfg/`, thus disabling the the `.cfg` files shipped by *package* as far as `update-updmap` is concerned. Then, *`package`*`.postrm` calls `update-updmap`, `mktexlsr` and `updmap-sys`, with the result that *package*'s map files aren't listed anymore in the final map files (`psfonts.map`, `pdftex.map`...) generated by `updmap-sys`.
  - If *package* is reinstalled later, two files are first created by `dpkg` during the unpack phase: `/var/lib/tex-common/fontmap-cfg/`*`package`*`.list` and `/etc/texmf/updmap.d/20foo.cfg.dpkg-new`. As long as the second one exists, the conffile `/etc/texmf/updmap.d/20foo.cfg` will be ignored by `update-updmap`[4] because it may be outdated. Eventually, *package* is configured; *`package`*`.postinst` runs `update-updmap`, `mktexlsr` and `updmap-sys`, and the `.cfg` files shipped by *package* aren't ignored by `update-updmap` this time, since they are referenced in `/var/lib/tex-common/fontmap-cfg/`*`package`*`.list` and the `.dpkg-new` files don't exist anymore. Thus, the map files shipped by *package* do end up in the final map files generated by `updmap-sys`.

6.2.2 Language/Hyphenation configuration

A package that provides additional hyphenation patterns for TEX should be usable with any Basic TEX Package. The recommended way to implement the configuration scheme described below is to use the `debhelper` program `dh_installtex` provided by `tex-common`. See `dh_installtex(1)` for usage details. Note that for `language.dat`, order is important: english should always be the first language.

[4] An `update-updmap` call could take place if another package such as `texlive-*` is configured in the meantime. That happens sometimes with APT.

These packages should put the actual hyphenation file into the respective places in *TEXMFMAIN*, and have them registered by putting a configuration file with extension `.cnf` into `/etc/texmf/language.d` and calling `update-language`. The file contents will then be incorporated into `/var/lib/texmf/tex/generic/config/language.dat`, the effective configuration file for TEX and friends' hyphenations.

Hyphenation patterns present the same problem as described in the previous section for font configuration files: If the package is removed, but not purged, the patterns are deleted, but the configuration information is still in `/etc/texmf/language.d/`, and the format generation would fail if they would be included in `language.dat`. Therefore, an analogous mechanism has been implemented as described for `update-updmap`: If a file in `/etc/texmf/language.d/` contains the "magic comment"

```
# -_- DebPkgProvidedMaps -_-
```

it will only be used as long it is:

- up-to-date (which is assumed if the same file with `.dpkg-new` suffix doesn't exist in the same directory);
- listed in a file in `/var/lib/tex-common/language-cnf/` which should have the name *`package`*`.list`.

Calling update-language is *not* sufficient to be able to use the new hyphenation patterns; instead the formats that use it need to be regenerated. This can be done by running `fmtutil-sys --byhyphen 'kpsewhich --progname=latex language.dat'`.

If a package that provides additional hyphenation patterns is removed, it must make sure the formats are properly recreated without it. With the "magic comment" mechanism, this means to run `update-language` and `fmtutil-sys --byhyphen 'kpsewhich --progname=latex language.dat'` in `postrm`

There is currently no mechanism (i.e., no `update-language`) for automatic addition of hyphenation patterns to formats that do not use the same hyphenation configuration file as LATEX.

The recommended way for implementing this scheme is to use `dh_installtex`.

6.2.3 Format configuration

As with font map configuration and language hyphenation patterns configuration, packages that provide additional formats should be usable with any Basic TEX Package. The recommended way to implement the configuration scheme described below is to use the `debhelper` program `dh_installtex` provided by `tex-common`. See `dh_installtex(1)` for usage details. Note that for `fmtutil.cnf`, order is important: Formats will be created for each line, and thus format files created from later lines will overwrite earlier ones.

These packages should put a configuration file according to `fmtutil.cnf(5)` into `/etc/texmf/fmt.d/`, run `update-fmtutil` and subsequently create the format with `fmtutil-sys --byfmt` *`format`*. `fmtutil-sys` will only try to create the format if it can find the corresponding *`format`*`.ini` file (the last argument in an `fmtutil.cnf` line). Therefore the *`format`*`.ini` file should not be a conffile.

If a package needs to create formats at runtime, it should use a local `fmtutil.cnf` with the appropriate entries and specifiy its location to `fmtutil` on the command line, using the `--cnffile` switch.

Upon package removal, `update-fmtutil` must be called in postrm, and the created formats and log files should be removed from the directory specified by `'kpsewhich -var-value=TEXMFSYSVAR'/web2c`.

The recommended way for implementing this scheme is to use `dh_installtex`.

6.3 Best practices for packages that `build-depend` on the TEX system

6.3.1 Configuration

If packages that build-depend on the TEX system need a changed configuration, they should not try to provide it statically. If settings in any other configuration file are inappropriate for a package to build, this is (usually) a bug in the package that provides the file. It should be fixed in this package, not circumvented by a workaround in the build process. Such workarounds have proven to be problematic, because they might stop working after changes in the depended-on package, and such failure cannot be foreseen by its maintainers. If a change is still necessary, the package should use the configuration update programs with the `--outputdir` and `--add-file` options.

6.3.2 Font cache data

Font cache data are created each time a font in METAFONT format is used, and placed by default in *TEXMFVAR*. During package build, this has to be avoided. In order to be able to clean up the generated files (and only those), the font cache should instead be put below the build directory. This can be achieved by setting *TEXMFVAR* to a subdirectory of the current directory, e.g. `$(CURDIR)/.texmf-var`, using Make's built-in variable. Packages which do not change *TEXMFVAR must* not create documentation that uses METAFONT fonts in the `binary` target.

6.4 Command execution and format files

If TEX formats need to be generated before execution, this should be done in the post-installation script. Packages that depend on an executable can thus simply declare `Depends:` on the package providing the executable, and *only* do that. Any additional checks, e.g. for the existence of format files, is unnecessary and harmful, causing internal changes (e.g. of format file extensions) to break the depending package that does this check. Maintainer scripts or programs in Debian packages should always use `fmtutil` or `fmtutil-sys` for format generation, and either add a `fmtutil.cnf` snippet in `/etc/texmf/fmt.d/` (with `fmtutil-sys`, for site-wide formats), or use `fmtutil` with the `--cnffile` option and an appropriate local `fmtutil.cnf` (for runtime programs)

Local administrators can override settings from `texmf.cnf` with environment variables; this has sometimes lead to errors in `postinst` scripts. It is recommended that `postinst` scripts unset relevant variables before format creation or other problematic tasks.

If an add-on package generates a format upon installation that needs a base format (e.g. latex.fmt), it must not load the existing base format [5]. Instead the `fmtutil.cnf` snippet and the *`format`*`.ini` file must be changed so that the process of format creation is repeated. For example, if upstream creates their format by loading latex:

```
latex           pdfetex         language.dat    -translate-file=cp227.tcx *latex.ini
jadetex         etex            language.dat    &latex jadetex.ini
```

and the following `jadetex.ini` file:

```
\input jadetex.ltx
\dump
```

then the Debian package maintainer must load `latex.ini` instead of `latex.fmt`, making sure that `\dump` in `latex.ltx` has no effect, and create the following new `jadetex.ini`:

```
\let\savedump\dump
\let\dump\relax
\input latex.ini
\let\dump\savedump

\input jadetex.ltx
\dump
```

and the following snippet for `fmtutil.cnf`:

```
jadetex         etex    language.dat    -translate-file=cp227.tcx *jadetex.ini
```

6.5 The Dpkg Post-Invoke Mechanism

This section was intended to deal with a once-planned mechanism that would allow to delay running of mktexlsr, updmap and perhaps even "fmtutil –all" until all TEX-related packages that want to do this are configured. Thus, it would be unnecessary to call the programs multiple times. Coding this is not hard, however it is unclear how it could be made sure that failures get attributed to the correct package. Therefore this plan has been dropped.

[5] The reason is that, in order to avoid other problems, `update-fmtutil` ignores files in `/etc/texmf/fmt.d` that have a corresponding `.dpkg-new` file, and that it is necessary to recreate all formats when pool files or engines are updated. Thus, some Basic TEX packages call `fmtutil --all` in their postinst scripts. When Basic TEX packages are upgraded together while a package that loads `latex.fmt` is installed and configured, then one of the Basic TEX packages' postinst will call `update-fmtutil` and `fmtutil --all` while others are is still unconfigured and have `.dpkg-new` files. Consequently, no format information for e.g. LATEX is available, and the generation of the format that wants to load it would fail. However, since all files needed to create e.g. `latex.fmt` are available, the depending format can `\input latex.ini` and create its own format without problems.

Appendix A

Sample code

This section contains sample code that implements the recommodations of this document.

A.1 Sample code for font packages

Sample postinst script:

```
#
# postinst-texfonts
#
# postinst snippet for installing Type 1 fonts for TeX
#
# Author: Florent Rougon <f.rougon@free.fr>
#
update_fontmaps()
{
    update-updmap --quiet
    # All of the following needs an installed and configured
    # basic TeX system, so check this.
    if kpsewhich --version >/dev/null 2>&1; then
        # mktexlsr is recommended now because updmap-sys relies
        # heavily on Kpathsea to locate updmap.cfg and the map files.
        # Also, it is slightly better not to specify a particular
        # directory to refresh because updmap.cfg is typically found
        # in TEXMFSYSVAR while the map files are in TEXMFMAIN or
        # TEXMFDIST.
        if which mktexlsr >/dev/null; then mktexlsr; fi
        if which updmap-sys >/dev/null; then
            printf "Running updmap-sys... "
            updmap-sys --quiet
            echo "done."
        fi
    fi

    return 0
}

case "$1" in
    configure|abort-upgrade|abort-remove|abort-deconfigure)
        update_fontmaps
    ;;

    *)
        echo "postinst called with unknown argument '$1'" >&2
        exit 1
    ;;
esac
```

Sample postrm script:

```
#
# postrm-texfonts
#
# postrm snippet for installing Type 1 fonts for TeX
#
# Author: Florent Rougon <f.rougon@free.fr>
#
tell_that_errors_are_ok()
{
    # Cheap option handling...
    if [ "$1" = -n ]; then
        prog="$2"
```

```
        endwith=' '
    else
        prog="$1"
        endwith='\n'
    fi

    printf "\
Trying to run '$prog' (error messages can be ignored if tex-common
is not configured)...$endwith"

    return 0
}

# The function name is *try_to*_update_fontmaps because the following
# scenario might happen:
#    1. this package is deconfigured
#    2. tex-common and texlive-binaries are removed
#    3. this package is removed or purged
#
# (cf. Policy § 6.5, step 2, about a conflicting package being removed due
# to the installation of the package being discussed).
#
# In this case, update-updmap, mktexlsr and updmap-sys would all be gone once
# tex-common and texlive-binaries are removed, so we must append "|| true" to
# their calls.
try_to_update_fontmaps()
{
    # Don't print alarming error messages if the programs aren't even
    # available.
    if which update-updmap >/dev/null; then
        tell_that_errors_are_ok -n update-updmap
        update-updmap --quiet || true
        echo "done."
    fi

    # All of the following needs an installed and configured basic TeX system.
    # If there is one, register the fonts. Otherwise, that will be done later
    # when the basic TeX system is configured, so we can exit without
    # worrying.
    kpsewhich --version >/dev/null 2>&1 || return 0

    # mktexlsr is recommended now because updmap-sys relies heavily on
    # Kpathsea to locate updmap.cfg and the map files. Also, it is slightly
    # better not to specify a particular directory to refresh because
    # updmap.cfg is typically found in TEXMFSYSVAR while the map files are in
    # TEXMFMAIN.
    if which mktexlsr >/dev/null; then
        tell_that_errors_are_ok mktexlsr
        mktexlsr || true
        echo "done."
    fi

    if which updmap-sys >/dev/null; then
        tell_that_errors_are_ok -n updmap-sys
        updmap-sys --quiet || true
        echo "done."
    fi

    return 0
}

case "$1" in
    remove|disappear)
        try_to_update_fontmaps
    ;;

    purge)
        # Supposing updmap.cfg & Co are clean (which I think is a reasonable
        # assumption), we don't need to call try_to_update_fontmaps().
        # Calling it on remove _and_ on purge just for hypothetical users
        # who would break their config before purging this package seems to
        # be more annoying than useful (it takes a lot of time).
    ;;

    upgrade|failed-upgrade|abort-upgrade|abort-install)
    ;;

    *)
        echo "postrm called with unknown argument '$1'" >&2
        exit 1
    ;;
esac
```