TEX Live's new infrastructure

Norbert Preining

Abstract

Since the release of TEX Live 2007 a new infrastructure for TEX Live distribution and management has been developed. This article presents the reasons for this switch, the ideas behind the new infrastructure, software developed, and ways to incorporate this new infrastructure. We will close with a look at what new features this new infrastructure could bring to the TEX (Live) world.

Sommario

Dopo l'uscita di TEX Live 2007 è stata sviluppata una nuova infrastruttura per la distribuzione e la gestione di TEX Live. L'articolo illustra le ragioni di questo cambio, le idee alla base della nuova infrastruttura, il software sviluppato e i modi per incorporarla. L'articolo chiude con uno sguardo alle nuove possibilità che la nuova infrastruttura introduce nel mondo di TEX (Live).

1 Introduction

TEX Live is an easy way to get up and running with TEX. It provides a comprehensive TEX system with binaries for most flavors of Unix, including GNU/Linux, and also Windows. It includes all the major TEX-related programs, macro packages, and fonts that are free software, including support for many languages around the world.

Since 1996 it tries to bring to all TEX-users as much material from CTAN as possible, packaged for 'consumption', i.e., for using it as a live system from DVD, or installing it on a variety of operating systems/architecture combinations.

These two requirements, incorporating as much as possible from CTAN, and providing binaries for a wide range of OS/arch combinations, has led to a huge number of supporting scripts, in a wide variety of programming languages (Perl, shell, XML, C, ...). These scripts were (and are) used to generate installation media, check consistency, update important files (e.g., for the installer), incorporating packages from CTAN, just to name a few. There have been many contributors; Sebastian Rahtz, Fabrice Popineau, and Karl Berry are the principal authors.

As always with overload volunteers working on big projects, not much was documented, programming was done by trial and error (see for example the packaging scripts of Debian, one of my own more horrible creations). This wouldn't have been reason enough to change things this deep in the intestines of TEX Live, but other illusions and dreams have driven us to rebuild from the ground up.

2 A world full of TPMs

Up to TEX Live 2007 everything in TEX Live was organized in `tpm`s, an acronym for 'TEX Package Manage[r/ment]'. One `tpm` described more or less one package from CTAN, containing the list of included files, title, description, license information, sometimes version numbers, additional information necessary for incorporation (activation of map files, hyphenation patterns, and formats).

Maintenance of these `tpm`s was mostly done by a Perl module `Tpm.pm` written by Fabrice Popineau, and the accompanying mystical program `tpm-factory.pl`, a script so full of possibilities that nobody besides Fabrice probably ever understood everything that could be done with it. Some of the jobs of this `tpm-factory.pl` were

- regeneration of the `tpm`s, this included some magic in finding the right files
- creation of a new `tpm` for a newly installed package from CTAN
- checking the coverage, i.e., checking that every file present in the repository is actually contained in a `tpm`.
- duplication and dependency checks

Alas, there have been some problems with all the `tpm`-business:

- they contained a mixture of generated (file lists) and static information (actions to be carried out);
- they were full of duplicate information: version, license, descriptions were taken now and then from the TEX Catalogue, but were typically horribly out of date or otherwise wrong;
- we had to generate so-called 'lists' files (line-oriented plain text file lists) from the `tpm`s for the installer because the XML syntax of the `tpm`s is not practical to parse from a simple shell script.

3 Aims of the new infrastructure

So around mid-2006 discussion of a new infrastructure started, but unfortunately, due to the usual

time constraints of all involved, without much concrete outcome except many emails. At least a preliminary catalogue of items to be improved sprang into existence:

Separation of static from generated content

As already mentioned, the `tpm`s contained a wild mixture of stuff from various sources (the file itself, the TEX Catalogue, the repository). The new infrastructure should have some kind of 'source' files where *only* the necessary information is stored, and all other content is added at generation time (of whatever). Naturally that could have been done on top of the `tpm`s, too, but starting from scratch seemed to be a better option.

Getting rid of lists files

Using the `tpm`s and some XSLT processing, the lists files were generated before release. These lists files (one for each package, about 2000 in all) must be read by the installer from the DVD, which caused a lot of headache and slow installations.

The sole reason for the existence of these lists files was the fact that the installer, written in shell, couldn't parse the XML `tpm`s, since XML parsing programs cannot be assumed to be present at all installations. The new infrastructure should be based on some format that is easily parseable using stuff normally already present, or at least necessary for TEX Live anyway.

Single package updates via the web

Some TEX distributions, notably MiKTeX, already support single package updates over the Internet. TEX Live, as big and nice as it is, still lacks this lovely feature. Of course it was put high on the priority list to allow for single package updates.

Better documentation

Last, but not least, we hoped that by rewriting the infrastructure and documenting it on the way we could provide a more stable foundation for more development and improvement, thus attracting more contributors. Furthermore, since teTEX development has been stopped, TEX Live is taking the place as the TEX system of choice in many GNU/Linux and other free distributions, and better documentation would only help providing those distributions with some aid for the migration.

4 New infrastructure – the basics

There are three type of files: `tlpsrc`, `tlpobj`, `tlpdb`. The format of these files are very similar to Debian's Package files: a simple list of

```
key    value
```

(without leading spaces). Empty lines at the beginning and end of a `tlpsrc` or `tlpobj` file are ignored. Also lines starting with `#` are ignored (to allow for comments). Some special cases apply for `tlpobj` files and the `tlpdb` file, see below.

4.1 `tlpsrc` file format

The possible keys and their respective interpretation for the `tlpsrc` files are:

name identifies the package, `value` must consist only of `[-_a-zA-Z0-9]`.

category identifies the category into which this package belongs. Possible categories are `Collection`, `Scheme`, `TLCore`, `Documentation`, `Package`. There are no particular checks as to whether a `tlpsrc` file called `collection-something` actually belongs to the category `Collection`. Most packages will fall into the `Package` category. These categories were inherited from the `tpm` world and may be modified at some point; for now, they suffice.

catalogue identifies the name under which this package can be found in the TEX Catalogue. If not present the name of the packages is taken as the Catalogue entry.

shortdesc gives a one line description of the package. Susequent entries will overwrite the former ones. In TEX Live only used for collections and schemes.

longdesc gives a long description of the package. Susequent entries are added up to a long text. In TEX Live only used for collections and schemes.

depend gives the list of dependencies of the package in the form

```
Category/Name
```

for `value`. All the depend lines contribute to the dependencies of the package.

execute gives a free form entry of install time jobs to be executed. Currently the following values are understood by the installers:

- `execute addMap font.map` enables the font map file `font.map` in the `updmap.cfg` file.
- `execute addMixedMap font.map` enables the font map file `font.map` for Mixed mode in the `updmap.cfg` file. By the way, the purpose of MixedMap is to help users with printers that render the Type 1 versions of the fonts worse than the (mode-tuned) bitmap-based Type 3 fonts. The entries from MixedMap are not added to psfonts_pk.map; that's the only difference. It's used for fonts which have both Metafont and (typically) autotraced Type 1 instantations.

- `execute BuildLanguageDat NN` activates all the hyphenation patterns found in `Master/texmf/tex/generic/config/language.NN.dat` in the generated `language.dat` file.
- `execute BuildFormat FMTCFG` activates all the formats present in `Master/texmf/fmtutil/format.FMTCFG.cnf` in the generated `fmtutil.cnf` file.

(src|run|doc|bin)pattern pattern adds a pattern (see below) to the respective list of patterns.

4.1.1 Patterns

To automatically generate the file lists in `tlpobj`s a specific pattern language was developed. Patterns can include or exclude files or whole sub-directories into the file list of a `tlpobj`. Patterns are of the form

```
[PREFIX]TYPE PAT
```

where `PREFIX` can be `+`, `!+`, or `!`, and `TYPE` one of the letters `t`, `f`, `d`, `r`. An initial `+` for the pattern indicates that the automatically generated pattern should not be cleared (see below), while the `!` indicates that the matching files should be excluded.

The meaning of the various pattern types is

f path includes all files which match `path` where *only* the last component of `path` can contain the usual glob characters * and ? (but no others!).

d path includes all the files in and below the directory specified as `path`.

t word1 ... wordN wordL includes all the files in and below all directories of the form

```
word1/word2/.../wordN/.../any/dirs/../wordL/
```

i.e., all words but the last form the prefix of the path, then there can be an arbitrary nesting of directories, followed by `wordL` as another directory.

r regexp includes all files matching the Perl regexp `/^regexp$/`

4.1.2 Autogenerated patterns

In the case that one of the pattern sections is empty or *all* the provided patterns have the prefix `+` (e.g., `+f ...`), then the following patterns are *automatically* added at expansion time (but never written to the textual representation):

- for runpatterns of category `Package`

```
t texmf-dist topdir name
```

where `topdir` is one of: bibtex, context, dvips, fonts, makeindex, metafont, metapost, mft, omega, scripts, tex, vtex. `name` refers to the name of the package as given by the `name` directive.

For other categories *no* patterns are automatically added to the list of runpatterns.

- for docpattern of category `Package`

```
t texmf-dist doc name
```

and for docpattern of category `Documentation`

```
t texmf-doc doc name
```

- for srcpattern of category `Package`

```
t texmf-dist source name
```

and for srcpattern of category `Documentation`

```
t texmf-doc source name
```

`binpatterns` are never automatically added.

In addition some magic tricks have been added to make writing of `binpatterns` easier:

arch expansion In case the string `${ARCH}` occurs in one `binpattern` it is automatically expanded to the respective architecture.

bat/exe/dll for win32 For `binpatterns` of the form `f bin/win32/foobar` (i.e., also for a binpattern of the form `f bin/${ARCH}/foobar`) files `foobar.bat`, `foobar.dll`, and `foobar.exe` are also matched.

The above two properties allows to capture the binaries for all architectures in one binpattern:

```
binpattern f bin/${ARCH}/dvips
```

will include `bin/win32/dvips.exe` in the runfiles when arch=win32.

Note that the bat/exe expansion *only* works for patterns of the `f`-type.

4.2 `tlpobj` file format

These files are structurally similar to `tlpsrc` files; the only difference is that the `*pattern` keys are not allowed, their place being taken by `*files` keys having a peculiar syntax (see below). Furthermore, an additional key `revision` and all keys matching `catalogue-*` are allowed. The `catalogue-*` keys specify information simply copied from the TEX Catalogue. We'll discuss the others below.

4.2.1 The `revision` entry

The `revision` key is *not* related to the package's version as specified by the author. Those versions are far too random, hard to extract, and sometimes not even present, to be a reliable basis for a packaging system. Instead, we look at all the package's files in the TEX Live source repository, which is currently Subversion, and use the maximum version number found there. Since Subversion uses simple integers for version numbers, this is simple to use in programs, and just as important, this whole process of finding the version number can be reliably automated.

4.2.2 The `*files` entries

While the `tlpsrc` files specify the files to be included using the patterns, the `tlpobj` files contain the already expanded file list. All the 'files' keys have in common that they are followed by a list of files (and possibly additional tags) indented by exactly one (1) space. They differ only in the first line itself (described below).

srcfiles, runfiles each of these lines is (or better should be) tagged with `size=NNNN` where `NNNN` is the sum of sizes of the single files (currently in bytes), e.g.,

```
srcfiles size=NNNNNN
```

docfiles The docfiles line itself is similar to the `srcfiles` and `runfiles` lines above:

```
docfiles size=NNNNNN
```

But the lines listing the files are allowed to have additional tags `details` and `language`:

```
file detail="foo bar" language="en"
```

For an example see listing 1. The reason to add these tags is the hope that someone will write a nice replacement for `texdoctk`. Note that these tags' source is the TEX Catalogue, and are in no way obligatory.

binfiles Since `binfiles` are different for the different architectures, one `tlpobj` file can contain `binfiles` lines for different architectures. The architecture is specified on the `binfiles` lines using the `arch=XXX` tag. Thus, `binfiles` lines look like

```
binfiles arch=XXXX size=NNNNN
```

A more complete example taken from `bin-dvipsk.tlpobj` can be seen in listing 2.

Listing 1: Excerpt from achemso.tlpobj

```
...
docfiles␣size=135842
␣texmf-dist/doc/latex/achemso/⤸
    ⤷README␣details="Package␣⤸
    ⤷Readme"␣language="de"
␣texmf-dist/doc/latex/achemso/⤸
    ⤷achemso.pdf␣details="⤸
    ⤷Package␣documentation"␣⤸
    ⤷language="de"
␣...
```

Listing 2: Excerpt from bin-dvipsk.tlpobj

```
name␣bin-dvipsk
category␣TLCore
revision␣4427
docfiles␣size=959434
␣texmf/doc/dvips/dvips.html
␣...
runfiles␣size=1702468
␣texmf/dvips/base/color.pro
␣...
␣texmf/scripts/pkfix/pkfix.pl
binfiles␣arch=i386-solaris␣size⤸
    ⤷=329700
␣bin/i386-solaris/afm2tfm
␣bin/i386-solaris/dvips
␣bin/i386-solaris/pkfix
binfiles␣arch=win32␣size=161280
␣bin/win32/afm2tfm.exe
␣bin/win32/dvips.exe
␣bin/win32/pkfix.exe
...
```

4.3 `tlpdb` file format

A `tlpdb` file will describe the status of an installation, that is it will contain all the installed packages' `tlpobj` files. Thus, the format of a `tlpdb` file is quite simple. It is the concatenation of `tlpobj`s with (at least) one empty line between different `tlpobj`s.

5 Programming APIs

Wrapping these file formats into a programming API represents the actual work of realizing the new infrastructure. On the way we had to create not only modules for accessing the content of the above files in some object-oriented way, we also created some additional modules for interfacing to the Subversion repository and the TEX Catalogue.

Currently, the following modules are available in the TEX Live subversion repository:

TeXLive::TLTREE an object that exhibits the properties of the subversion repository in some structured form. In principle it is the scooping up of `svn status -v` output with post-processing for easier and faster searches.

TeXLive::TeXCatalogue a very simple interface to the TEX Catalogue, barely providing minimal information. Will be used for enriching the final database.

TeXLive::TLPSRC provides access to `tlpsrc` files, basic functionality like reading in, writing out, and member access functions. In addition, it allows to generate, or expand, a `TLPSRC` object to a `TLPOBJ` object using an instance of `TLTREE`. That is, it takes the patterns specified in the `tlpsrc`, and tries to find all files matching these patterns. After this it recomputes the size and returns a `TLPOBJ` object.

TeXLive::TLPOBJ provides access to `tlpobj` files, and entries stored in a `TLPDB` object. As with the `tlpsrc`, basic functionalities are exported, and also a function to create a zip file for the `TLPOBJ` (which later should be used for web updates or building the 'inst' CD), and together with an instance of a `TeXCatalogue` object some information can be transferred from the TEX Catalogue to the `TLPOBJ` object.

TeXLive::TLPDB provides access to the TEX Live database. These files will serve the central purpose of describing the current status of various media, like an installation on a user computer, or the distribution. Comparing these status descriptions, update programs can deduce what packages should be updated.

TeXLive::TLPUtils exports some handy functions used in all the other modules.

The full API documentation is provided in `pod` format within the perl modules. A Perl API document is available in the TEX Live subversion repository, too.

In addition to the Perl API a start at a shell (sh, bash, etc.) API and implementation has been created, but it presently lacks several key features, and is not up to date. Jim Hefferon has contributed a start at a Python API. However, we assume that on all computers Perl will be available or will be installed during installation.

6 Integration into current TEX Live development

Around these modules several day-in-day-out tasks of the TEX Live developers have been rewritten using the new infrastructure, the most important one being the script `ctan2tl`. This is the script responsible for transferring a package from CTAN into the right places in the TEX Live repository, and preparing it for users.

Other things already converted are automatic update of the `texlive.tlpdb`, the file containing the `TLPDB`.

On the other hand not everything has been done, the biggest piece for sure is the rewrite of the installer. The old installer, relying on the lists files we wanted to get rid of, is still the only one we have. Furtunately we can generate lists files from the `texlive.tlpdb`, but in the not-so-long run, meaning hopefully for TEX Live 2008, we want to have a new installer.

Other things still missing are coverage and duplication checks, but those are quite easy to do using the Unix command world (`grep`, `sort`, and `uniq` being the magic incantations).

7 Closing

Although we are quite confident that the new infrastructure will work well, only the reality of the next release will prove us right or wrong. Many of the key features of the new infrastructure have been designed with the `tpm`-system in mind, and some 'features' carried over will probably proven superfluous. Nothing with the current infrastructure is written in stone, and we are open for proposals and improvements, keeping in mind that releasing TEX Live should not be made more difficult.

That said, we invite contributors and everyone interested to contact us at `tex-live@tug.org`, or take a look at the following resources:

- `http://www.tug.org/texlive` – the main entry point, with links to developers' resources, documentation;
- `http://www.tug.org/svn/texlive/trunk/` – web view onto the subversion repository;
- `svn://tug.org/texlive/trunk` – svn repository, anonymous access, take care when you checking out the repository, it needs several Gigabyte of disk space;
- `http://www.tug.org/texlive/pkgupdate.html` – an explanation how updates from CTAN to TEX Live are done.

Finally, I want to thank all the contributors to TEX Live, there are too many to mention. In the course of developing this infrastructure the discussions with Karl Berry, Paweł Jackowski, Reinhard Kotucha, Siep Kroonenberg, and Jerzy B. Ludwichowski were very helpful.

▷ Norbert Preining
Vienna University of Technology
Wiedner Hauptstr. 10
1040 Wien, Austria
`preining@logic.at`