TeX Live's new infrastructure

Norbert Preining

Vienna University of Technology, Austria

BACHOTEK 2007

Bachotek, Poland 30 April 2008

The TeX Live distribution

- includes all the free stuff from CTAN
- available for a wide range of platform–operating system combinations
- currently is replacing teTeX in many (Unix) distributions as default TeX system

Upstream organization

- SVN repository where many people have write permissions
- loads of supporting scripts doing a variety of jobs:
  - preparing the installation for mastering
  - installation from various media
  - installation of packages from CTAN into the SVN repository
  - performing various checks on the whole archive (coverage, double inclusion, etc.)

Upstream organization

- SVN repository where many people have write permissions
- loads of supporting scripts doing a variety of jobs:
  - preparing the installation for mastering
  - installation from various media
  - installation of packages from CTAN into the SVN repository
  - performing various checks on the whole archive (coverage, double inclusion, etc.)

Problems

- hand-crafted, hard to read
- not easily extensible
- no documentation

Previous infrastructure: tpms

TEX Package Manage[r/ment] files collected a variety of information into on XML file:

- file patterns and file lists
- title, description, license, versions
- activation of map files, formats, hyphenation patterns

Previous infrastructure: tpms

TEX Package Manage[r/ment] files collected a variety of information into on XML file:

- file patterns and file lists
- title, description, license, versions
- activation of map files, formats, hyphenation patterns

Problems with tpms

- mixture of generated and static information
- duplicated information (version, license, descriptions, ...) which were outdated
- hard to parse, thus the necessity to generate another set of files for the installer

Aims of the new infrastructure

- Separation of static from generated content
  Going from a ‘source’ to the ‘object’ should include automatically data from various other sources (mainly the Catalogue and the repository)

Aims of the new infrastructure

- Separation of static from generated content
  Going from a ‘source’ to the ‘object’ should include automatically data from various other sources (mainly the Catalogue and the repository)
- Getting rid of any ‘additional’ files (list files)
  any additional files tend to be outdated, all the necessary information should be present in *one* place and be easily parseable.

Aims of the new infrastructure

- Separation of static from generated content
  Going from a ‘source’ to the ‘object’ should include automatically data from various other sources (mainly the Catalogue and the repository)
- Getting rid of any ‘additional’ files (list files)
  any additional files tend to be outdated, all the necessary information should be present in *one* place and be easily parseable.
- Single package updates via the web
  updates to single packages (e.g., a new beamer release)

Aims of the new infrastructure

- Separation of static from generated content
  Going from a ‘source’ to the ‘object’ should include automatically data from various other sources (mainly the Catalogue and the repository)
- Getting rid of any ‘additional’ files (list files)
  any additional files tend to be outdated, all the necessary information should be present in *one* place and be easily parseable.
- Single package updates via the web
  updates to single packages (e.g., a new beamer release)
- Better documentation
  since TEX Live is replacing teTEX we want to give people incorporating TEX Live into distributions a better documented and easier to handle system

The central `texlive.tlpdb`

One installation or media is now completely described by one file, the TeX Live Database:

- simple text file – easily parseable
- revision number for the single packages
- generated from static content (the tlpsrc files)
- enriched with information from the TeX Catalogue
- format documented in detail (POD documentation in the respective perl module)

How does `texlive.tlpdb` look like

texlive.tlpdb

```
name␣abbr
...

name␣memoir
...
```

- sequence of `key value` pairs
- separated by an empty line (or more)
- one group per package
- some ‘meta’-packages for configuration options and special purpose packages

The single package: `tlpobj` by example I

aoposter.tlpobj

```
name␣a0poster
category␣Package
revision␣7340
shortdesc␣Support␣for␣designing␣posters␣on␣large␣paper.
longdesc␣Provides␣fonts␣in␣sizes␣of␣12pt␣up␣to␣107pt␣and␣also␣makes␣sure
longdesc␣that␣in␣math␣formulas␣the␣symbols␣appear␣in␣the␣right␣size.␣Can
longdesc␣also␣create␣a␣PostScript␣header␣file␣for␣dvips␣which␣ensures
longdesc␣that␣the␣poster␣will␣be␣printed␣in␣the␣right␣size.␣Supported
longdesc␣sizes␣are␣DIN␣A0,␣DIN␣A1,␣DIN␣A2␣and␣DIN␣A3.
docfiles␣size=47
␣texmf-dist/doc/latex/a0poster/a0.pdf␣details="Package␣documentation␣(German)"␣↪
      ↪language="de"
␣texmf-dist/doc/latex/a0poster/a0.tex
␣texmf-dist/doc/latex/a0poster/a0_eng.pdf␣details="Package␣documentation␣(English)↪
      ↪"␣language="en"
␣texmf-dist/doc/latex/a0poster/a0_eng.tex
runfiles␣size=4
␣texmf-dist/tex/latex/a0poster/a0poster.cls
␣texmf-dist/tex/latex/a0poster/a0size.sty
catalogue-version␣1.22b
catalogue-date␣2006-11-28␣22:38:04␣+0100
catalogue-ctan␣/macros/latex/contrib/a0poster
catalogue-license␣lppl
```

The origin of this `a0poster.tlpobj`

aoposter.tlpsrc

```
name a0poster
category Package
```

- minimal input file with static data
- rest is generated from actual svn repository (revision, size)
- enriched with information from the TEX Catalogue (catalogue-*, specification of the documentation files)
- tagged documentation files (details, language), information again from the Catalogue

The single packages: `tlpobj` by example II

bin-bibtex8 and friends

```
name␣bin-bibtex8
category␣TLCore
revision␣7340
depend␣bin-bibtex8.ARCH
docfiles␣size=15
␣texmf/doc/bibtex8/00readme.txt
␣texmf/doc/bibtex8/HISTORY
␣texmf/doc/bibtex8/csfile.txt
␣texmf/doc/bibtex8/file_id.diz
runfiles␣size=10
␣texmf-dist/bibtex/csf/base/88592pl.csf
␣texmf-dist/bibtex/csf/base/cp1250pl.csf
␣texmf-dist/bibtex/csf/base/cp852pl.csf
␣texmf-dist/bibtex/csf/base/iso8859-7.csf

name␣bin-bibtex8.alpha-linux
category␣TLCore
revision␣7340
shortdesc␣binary␣files␣of␣bin-bibtex8␣for␣alpha-linux
binfiles␣arch=alpha-linux␣size=62
␣bin/alpha-linux/bibtex8

...
name␣bin-bibtex8.win32
category␣TLCore
revision␣7340
shortdesc␣binary␣files␣of␣bin-bibtex8␣for␣win32
binfiles␣arch=win32␣size=25
␣bin/win32/bibtex8.exe
```

The origin of the above `bin-bibtex8`

bin-bibtex8.tlpsrc

```
name bin-bibtex8
category TLCore
runpattern d texmf-dist/bibtex/csf/base
docpattern f texmf/doc/bibtex8/*
binpattern f bin/${ARCH}/bibtex8
```

- various patterns for capturing files
- tricks to capture binaries on unis and windows
- separate objects for the binary files of the package

The pattern language

patterns are of the form

```
[PREFIX]TYPE PAT
```

where PREFIX can be +, !+, or !,

The pattern language

patterns are of the form

```
[PREFIX]TYPE PAT
```

where `PREFIX` can be `+`, `!+`, or `!`,and `TYPE PAT` can be:

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!).

The pattern language

patterns are of the form

```
[PREFIX]TYPE PAT
```

where `PREFIX` can be `+`, `!+`, or `!`,and `TYPE PAT` can be:

- `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`.

The pattern language

patterns are of the form

```
[PREFIX]TYPE PAT
```

where `PREFIX` can be `+`, `!+`, or `!`,and `TYPE PAT` can be:

- 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/
```

The pattern language

patterns are of the form

```
[PREFIX]TYPE PAT
```

where `PREFIX` can be `+`, `!+`, or `!`,and `TYPE PAT` can be:

- **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/
  ```

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

Example patterns

- `runpattern f texmf/chktex/*`
  includes all files *in* Master/texmf/chktex/

Example patterns

- `runpattern f texmf/chktex/*`
  includes all files *in* Master/texmf/chktex/
- `binpattern f bin/${ARCH}/bibtex`
  includes the bibtex binaries into the bin-bibtex package, depending on the architecture

Example patterns

- `runpattern f texmf/chktex/*`
  includes all files *in* Master/texmf/chktex/
- `binpattern f bin/${ARCH}/bibtex`
  includes the bibtex binaries into the bin-bibtex package, depending on the architecture
- `runpattern d texmf/tex/lambda/base`
  includes all files in and under the above path

Example patterns

- `runpattern f texmf/chktex/*`
  includes all files *in* Master/texmf/chktex/
- `binpattern f bin/${ARCH}/bibtex`
  includes the bibtex binaries into the bin-bibtex package, depending on the architecture
- `runpattern d texmf/tex/lambda/base`
  includes all files in and under the above path
- `runpattern t texmf-dist omega uni2char`
  includes all files in texmf-dist/omega/.../uni2char/

Example patterns

- `runpattern f texmf/chktex/*`
  includes all files *in* Master/texmf/chktex/
- `binpattern f bin/${ARCH}/bibtex`
  includes the bibtex binaries into the bin-bibtex package, depending on the architecture
- `runpattern d texmf/tex/lambda/base`
  includes all files in and under the above path
- `runpattern t texmf-dist omega uni2char`
  includes all files in texmf-dist/omega/.../uni2char/
- `runpattern r .*/foobar`
  includes the files matching the regexp

Autogenerated patterns

To keep `tlpsrc` files small, if a pattern section is empty or all patterns are prefixed with +, the following patterns are automatically generated:

- runpatterns in category Package:

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

- docpatterns in category Package:

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

- docpatterns in category Documentation:

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

- srcpatterns in category Package:

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

- srcpatterns in category Documentation:

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

Additional tricks

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

bat/exe/dll/texlua for win32 For `binpatterns` of the form `f bin/win32/foobar` files `foobar.bat`, `foobar.dll`, `foobar.exe`, and `foobar.texlua` are also matched.

Additional tricks

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

bat/exe/dll/texlua for win32 For `binpatterns` of the form `f bin/win32/foobar` files `foobar.bat`, `foobar.dll`, `foobar.exe`, and `foobar.texlua` are also matched.

Effects of auto generation and tricks

total number of tlpsrc files: 1644
total number of tlpsrc files with patterns: 172
number of bin- and hyphen- tlpsrc files with patterns: 123
number of 'normal' packages with patterns: 49

Allowed fields for `tlpobj`

- name: identifies the package
- category: one of (currently) `Collection`, `Scheme`, `TLCore`, `Documentation`, `Package`
- shortdesc, longdesc: description of the package
- depend: `Name` (multiple entries possible)
- execute: activating maps, formats, hyphenation patterns
- runfiles, docfiles, srcfiles, binfiles
  every files section has a size attribute, and the binfiles section can occur more then once with different arch tags (see above)
- revision: maximum svn revision number of the contained files, since version numbers are not parseable, trustworthy, or not even present
- catalogue-* keys: stuff taken from the catalogue
  for example catalogue-version, catalogue-authors, catalogue-license

Some special packages

Some packages do not relate to actual files but are used to save options and configurations into the database by putting them into a depend line.

- `00texlive.config` general configuration (release, src/doc container split)
- `00texlive-installation.config` only present in a final installation and contains the choices the admin made when doing the installation (paper a4/letter, pre-generate formats, installation location, ...)
- `00texlive.core` actually contains files, but those are never installed and this package is only here to collect files which are not contained in any package, thus making the coverage check squeak

Perl programming API

Important for ‘users’ or integrators

TeXLive::TLPOBJ for `tlpobj` files, basic functionality like read, write, and member access and change functions, etc.

TeXLive::TLPDB access to the TeX Live database.

TeXLive::TLPostInstall collects post installation actions

Perl programming API

Important for ‘users’ or integrators

TeXLive::TLPOBJ for `tlpobj` files, basic functionality like read, write, and member access and change functions, etc.

TeXLive::TLPDB access to the TeX Live database.

TeXLive::TLPostInstall collects post installation actions

Important for ‘us’ as developer:

TeXLive::TLTREE properties of the subversion repository, in principle it is `svn status -v`

TeXLive::TeXCatalogue simple interface to the TeX Catalogue

TeXLive::TLPSRC for `tlpsrc` files, basic functionality like reade, write, etc

TeXLive::TLUtils some handy functions

TeXLive::TLMedia abstracts an arbitrary installation media

Other (planned/wished) APIs

texlua	next on the list to be done, would really help us a lot
python	minimal code present (by Jim Hefferon)
C	some code present, was a GSoC project, but no slot available (code by Jjgod Jiang)
bash	maybe, some code present (by the author)
...	no idea what else ...

Documentation

- all modules contain a full documentation in pod format
- additional text API document
- article in ArsTEXnica, 2007:4, 69–73, and in proceedings of this conference (hopefully)

What to do with this stuff

- installer: more or less done, see other talk
- distribution inclusion: will hopefully work well
- texdoctk++: tagged documentation files could be used to write a better texdoctk, but depends on the information of the authors in the TeX Catalogue
- 'TeX Live Manager' `tlmgr`

TeX Live Manager `tlmgr`

New program collection several scripts under one hood, currently supports:

- installation of additional packages or collections, with or without automatic dependency installation
- same with removal
- update all packages to the newest versions available
- paper configuration like texconfig, but also for Windows
- listing of available and installed architectures, and adding new architectures to the installation
- searching the installed and all available packages
- list all schemes, collections, packages
- setting some default values like the installation location
- regenerate fmtutil.cnf, language.dat, and updmap.cfg from the information stored in the database and local additions
- uninstall the whole installation

Resources

- `tex-live@tug.org` – main contact point
- `www.tug.org/texlive` – the main entry point, with links to developers' resources, documentation
- `www.tug.org/svn/texlive/trunk/` – web view onto the subversion repository;
- `svn://tug.org/texlive/trunk` – svn repository, anonymous access
- `www.tug.org/texlive/pkgupdate.html` – an explanation how updates from CTAN to TeX Live are done.