

TeX Live's new infrastructure

Norbert Preining

Vienna University of Technology, Austria

BACHOTEK 2008

Bachotek, Poland 30 April 2008

The T_EX Live distribution

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

T_EX 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

T_EX 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 which were outdated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any additional files tend to be outdated, all the necessary information should be present in *one* place and be easily parseable.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 T_EX Live is replacing t_EX we want to give people incorporating T_EX 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_abbrev`

...

`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_aoposter
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/aoposter/a0.pdf_details="Package_documentation_(German)"_ )
  ( language="de"
_texmf-dist/doc/latex/aoposter/a0.tex
_texmf-dist/doc/latex/aoposter/a0_eng.pdf_details="Package_documentation_(English)"_ )
  ( " language="en"
_texmf-dist/doc/latex/aoposter/a0_eng.tex
runfiles_size=4
_texmf-dist/tex/latex/aoposter/aoposter.cls
_texmf-dist/tex/latex/aoposter/a0size.sty
catalogue-version_1.22b
catalogue-date_2006-11-28_22:38:04_+0100
catalogue-ctan_/macros/latex/contrib/aoposter
catalogue-license_lpl
```

The origin of this a0poster.tlplib

aoposter.tlplibsrc

name_a0poster

category_Package

- ▶ minimal input file with static data
- ▶ rest is generated from actual svn repository (revision, size)
- ▶ enriched with information from the T_EX 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 unix 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 ... word_N word_L 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 `tlp_src` 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 than 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 read, 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 *ArsT_EXnica*, 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 T_EX Catalogue
- ▶ ‘T_EX Live Manager’ `tlmgr`

T_EX 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 T_EX Live are done.