# TEX Live Manager's rare gems: User mode and multiple repository support

Norbert Preining

## Abstract

This article describes two features of TEX Live Manager: *user mode*, where the TEX Live Manager manages an arbitrary directory instead of the system distribution, and *multi-repository support*, where multiple sources can be used to fetch TEX Live packages.

## 1 Introduction

The TEX Live Manager (`tlmgr`) is responsible for managing a TEX Live installation. It can be used to search for packages as well as doing usual package management tasks (install, update, remove, backup, configure). In the last couple of releases, two more features are available that have been requested for a long time: *user mode* and *multiple repository support*.

In user mode, rather than managing the system tree and installation, `tlmgr` is used to manage an arbitrary `texmf` tree, for example the user's `TEXMFHOME` (this is in fact the default user mode tree). Although not all functionality is possible in user mode, the basic operations of package installation, removal, and updates can be handled by any user without requiring write permission to any system trees.

Multiple repository support was introduced to allow for easy handling of additional repositories of TEX Live packages. A few of them have come into existence (`tlcontrib`, `tlcritical`, `tlptexlive`, Korean support, ...), but until now one had to update packages one by one from these repositories. With multiple repositories this has been made a bit more convenient.

We will describe the background and operation of both of these features — why it was implemented, how it works, example runs, and warnings on usage.

## 2 User mode

This feature was probably one of the most requested in recent times: the ability to use `tlmgr` to manage an arbitrary `texmf` tree. Although many points remain to be improved, the current implementation allows users to install most packages, without needing write access to the main TEX installation.

The TEX Live Manager `tlmgr` has for many years been the main tool for configuration and management of a TEX Live installation. Most importantly, it is the main tool for updating existing packages and installing new ones.

In contrast to the original distribution of TEX Live from TUG, repackaged versions of TEX Live for full operating systems (Debian, SuSE, Fedora, ...) normally do not ship `tlmgr` (see article on pages 297–301), as it would interfere with the system package manager. But since distributions cannot feasibly keep up with the constant flow of TEX package updates, users have often asked for an option to use `tlmgr` to handle packages within their `TEXMFHOME`.

### 2.1 Background and description

Although I assume that the TEX Live Manager is by now sufficiently well known, here is a short reminder of some of the responsibilities of `tlmgr` in managing the full installation:

- options: paper size, installation source, creation of formats, etc.
- platform: adds/removes support for different platforms.
- recreates core configuration files: `updmap.cfg`, `fmtutil.cnf`, hyphenation definitions.
- packages: installs, removes, backups, restores.
- information: search and browse through the TEX Live database.

Most of these operations need full write access to the installation, which, particularly in the cases of distribution-repackaged or shared installations, may well be unavailable.

In contrast to this, `tlmgr` in *user mode* allows only a restricted set of functionality, and makes changes only below the home directory of the current users. Almost all actions are supported, with one important general restriction:

> Only *relocatable* packages can be installed in user mode. A relocatable package resides completely within one `texmf` tree. This excludes all packages with binaries or scripts.

Otherwise, operations work similarly to normal mode, but changing the respective local files. For example, the `generate` action creates various configuration files in the user's `TEXMFVAR` instead of in the system-wide directory `TEXMFSYSVAR`.

In the following we will go through the necessary steps and give a few examples.

### 2.2 Setup and operation

By default, user mode is not selected, so the command line argument `--usermode` has to be given on *all* operations.

#### 2.2.1 Initializing the user tree

Before starting to use user mode, a target tree has to be initialized. By default this is the user's `TEXMFHOME`, but any other tree can be specified with the command line option `--usertree`. This initialization is done by running `tlmgr --init-usertree`, which sets up the initial empty TEX Live database and creates necessary files. If the file TEXMFHOME/tlpkg/texlive.tlpdb already exists, the command will bail out. Other directories generated are TEXMFHOME/web2c and TEXMFHOME/tlpkg/tlpobj.

The initial texlive.tlpdb contains the very same options as the main installation concerning installation source, installation of documentation and source files, etc. After the initial setup, these options can be changed within the user tree in the normal way.

#### 2.2.2 Installation of packages

As already mentioned, not all packages can be installed in user-managed trees; all the packages that contain files *outside* the texmf-dist hierarchy are not installable in user mode. Technically, only those packages tagged as relocatable are installable in user mode — which is most of them. Looking at the current TEX Live database, at the moment there are 2454 packages that can be installed in user mode.

Having set up the initial tree, installation is straightforward (some line breaks are editorial):

```
$ tlmgr --usermode install 12many
tlmgr: package repository /home/norbert/tlnet
[1/1, ??:??/??:??] install: 12many [376k]
tlmgr: package log updated: /home/tl/texmf/
    web2c/tlmgr.log
running mktexlsr ...
```

Dependencies of collections and of packages are installed automatically, which allows installation of a full collection (more precisely, all of the installable packages from a collection):

```
$ tlmgr --usermode install collection-xetex
tlmgr: package repositories: /home/norbert/
    tlnet
[1/28, ??:??/??:??] install: arabxetex [618k]
[2/28, 00:00/00:00] install: euenc [153k]
[3/28, 00:00/00:00] install: fixlatvian [123k]
...
Package xecyr is not relocatable, cannot
    install it in user mode!
...
[27/28, 00:07/00:08] install: collection-xetex
    [1k]
tlmgr: package log updated at /home/tl/texmf/
    web2c/tlmgr.log
running mktexlsr ...
```

Package removal, updates, etc., all work in the analogous way.

### 2.3 Warnings and future work

User mode support is still relatively new, and has not been used extensively, so expect bugs to creep in. If you run across one, please let us know how to reproduce.

Another caveat with user mode is that all files in TEXMFHOME override system-wide files. In other words, if the system installation gets updated and the user installed packages become older, then the older version will still be used (which can be the desired behavior, after all).

Also, be aware that the size of TEXMFHOME can grow rapidly, especially when installing collections. Since this tree is not (by default) searched via ls-R, searching can become quite slow.

Last but not least, be warned if you are tempted to create copies of configuration files (see my other article in this volume cited earlier), as they are an endless source of problems — especially ones that are forgotten.

Concerning future developments of user mode, the set of supported actions in tlmgr is already sufficient. What remains to be done: support in the GUI, a distribution installation mode where there is no initial TEX Live database available, and perhaps an independent installation support, where there is not even a main TEX Live installation required.

## 3 Multi-repository support

Online updating of a TEX Live installation is done by fetching the TEX Live database of the remote repository, determining the set of updated packages, fetching the updated packages from that same remote repository, and finally unpacking and installing them. Multi-repository support allows for fetching from several sources, on a per-package basis.

### 3.1 Background and description

Over the years several additional repositories have come into existence. The first, as part of TEX Live itself, is called tlcritical, and provides testing releases of the TEX Live core infrastructure packages (e.g., tlmgr). Other notable repositories in use today are tlptexlive (for testing Japanese TEX integration and binary updates) and tlcontrib (for testing releases and items not distributable in TEX Live).

Since the very beginning, doing the full update circle as described above from any repository has been possible. But it was repetitive: a user wanting to update from different sources had to run tlmgr several times, once for each repository. The multi-repository support now allows configuring tlmgr to pull automatically from several repositories at the same time, somewhat similar to (though still different from), for example, apt-get in Debian.

There are a few points that set the TeX Live implementation of multi-repository support apart from similar features in other systems:

- There is a distinction between the main and subsidiary repository. The main repository should always be our `tlnet` distribution channel.
- By default everything is taken from the main repository. Subsidiary repositories are never used unless explicitly requested.
- To request a package from a subsidiary repository one has to *pin* this package to the respective repository, as explained below.
- Absolute revision numbers are *not* compared between repositories — only the pinning counts. Revision numbers are only used to compare between the local version and the selected version from the repository.

The necessity to pin a package explicitly to a subsidiary repository arose from our wish to avoid unnecessary splitting of repositories. In general, we want to have all freely available TeX-related packages to be available via TeX Live's `tlnet` channel. Furthermore, we want users to think twice before using packages from subsidiary repositories, as it can easily lead to problematic situations.

## 3.2 Setup and operation

By default `tlmgr` works in single-repository setup, well-known and well-tested. This repository can be set and changed with the invocation:

`tlmgr option repository` ⟨*url*⟩

### 3.2.1 Inspecting and adding repositories

To work with multiple repositories, a new `tlmgr` action `repository` has been added, with three sub-actions: `list`, `add`, `remove`. Initially there is only one repository:

```
$ tlmgr repository
List of repositories (with tags if set):
       /home/norbert/tlnet (main)
```

(The path here is my local copy of the `tlnet` directory from CTAN.) Giving `tlmgr repository` without any sub-action executes the `list` sub-action.

If we want to add a repository, we use the `add` sub-action:

`tlmgr repository add` ⟨*url*⟩ [*tag*]

where the *tag* is an optional short-hand for ⟨*url*⟩. The main repository always has the tag `main`. As an example, let's add a local copy of the `tlptexlive` repository:

```
$ tlmgr repository add \
  /home/norbert/tlptexlive tlptexlive
tlmgr: added repository with tag tlptexlive:
  /home/norbert/tlptexlive
$ tlmgr repository
List of repositories (with tags if set):
       /home/norbert/tlnet (main)
       /home/norbert/tlptexlive (tlptexlive)
```

### 3.2.2 Pinning

As mentioned above, before a subsidiary repository is used at all, it is necessary to pin the desired packages to the repository. Pinning is defined in the file `TEXMFLOCAL/tlpkg/pinning.txt`. This file consists of empty lines, comments starting with `#`, and lines of the form:

⟨*repo*⟩`:`⟨*pkgglob*⟩[`,`⟨*pkgglob*⟩] ...

where ⟨*repo*⟩ is a full URL or a repository tag given to `repository add`. ⟨*pkgglob*⟩ is a shell-style glob for package names, with multiple items separated by commas.

Editing this file manually is simple and perfectly ok to do. `tlmgr` also provides a convenience action to add, modify, and remove pinning data. The respective operations are:

```
tlmgr pinning [show]
tlmgr pinning add ⟨repo⟩ ⟨pkgglob⟩ ...
tlmgr pinning remove ⟨repo⟩ ⟨pkgglob⟩ ...
tlmgr pinning remove ⟨repo⟩ --all
```

As before, the `show` is optional (i.e., the default), and lists all current pins. To add new pinning data, use `pinning add` with the subsidiary's full URL or ⟨*tag*⟩ for ⟨*repo*⟩, and then a list of package globs. As usual, when ⟨*pkgglob*⟩ contains shell meta characters they have to be quoted properly on the command line. The `pinning remove` invocation erases the listed ⟨*pkgglob*⟩s from the list for ⟨*repo*⟩, and does nothing if there is no such pair. Finally, one can remove all entries of a certain repository with the last of the above four invocations.

Continuing from the above example, let us see this in action and specify that we want to install *all* available packages from the `tlptexlive` repository. "All" in shell-glob syntax is just `*`, so:

```
$ tlmgr pinning add tlptexlive '*'
tlmgr: package repositories:
       main = /home/norbert/tlnet
       tlptexlive = /home/norbert/tlptexlive
tlmgr: new pinning data for tlptexlive: *
```

### 3.2.3 Installation of packages

Installation of packages from the selected repository is completely transparent. `tlmgr` checks which packages are pinned to which repository, compares the local revision with the revision in the pinned repository, and updates as necessary.

Let us see this first in action for the installation of a new package not present in the main repository:

```
$ tlmgr install pmetapost
tlmgr: package repositories:
       main = /home/norbert/tlnet
       tlptexlive = /home/norbert/tlptexlive
[1/2, ??:??/??:??] install: pmetapost.x86_64-
    linux @tlptexlive [671k]
[2/2, 00:00/00:00] install: pmetapost
    @tlptexlive [1k]
tlmgr: package log updated: /home/tl/texmf/
    web2c/tlmgr.log
running mktexlsr ...
```

We can see that `tlmgr` is quite verbose in telling the user from which repository the package is installed.

Updates work in the same general way. Here we see that `tlmgr list` shows all the candidates available, from both repositories along with the respective revision numbers:

```
$ tlmgr update --list
tlmgr: package repositories:
       main = /home/norbert/tlnet
       tlptexlive = /home/norbert/tlptexlive
tlmgr: saving backups to /home/tl/tug2013/tlpkg
    /backups
update: dvips.x86_64-linux [136k]: local: 30204,
     source: 31002@tlptexlive
       other candidates: 30204@main
update: ptex.x86_64-linux [530k]: local: 30519,
     source: 31001@tlptexlive
       other candidates: 30519@main
...
```

To actually perform the updates, we would run the usual:

```
$ tlmgr update --all
```

## 3.3 Warnings and future work

Here are the obligatory set of warnings, redoubled for an operation that so deeply changes the normal behavior of `tlmgr`:

*No purely numeric comparison for selecting the candidate:* Candidates are selected solely based on the pinning, and not by selecting the highest number of the revisions in all repositories. This allows subsidiary repositories to have version numbers which are completely independent from the main TEX Live repository, where revision numbers are based on the Subversion revisions and thus are (other than being strictly increasing) unpredictable. As a consequence, this means that pinning a package to an alternative repository where the revision number is *smaller* than the one in `tlnet` will *not* automatically update the package the first time. An invocation of

`tlmgr install --reinstall` ⟨*pkg*⟩

is needed. After having done that the first time all further updates will come automatically from the subsidiary repository.

*Support for actions:* Not all operations of TEX Live Manager can be supported, but those for which it is reasonable are done. The GUI has basic support in that sources can be added/removed, and information is displayed. Pinning is not possible in the GUI at present, and the presentation could be improved.

*Leftover packages:* Due to the fixed pinning, if an outdated package (like a development release) is *not* removed from a subsidiary repository, the user will remain stuck with the development version even if a newer version has found its way into the main repository. Removing the development package from the subsidiary repository makes sure that this does not happen, and is by far the cleanest and best solution.

Last but not least, multi-repository support is a recent addition and should thus be tried with special care and thought.

## 4 Closing

The development of TEX Live Manager has slowed, or more properly stabilized, as we come closer to the (asymptotic) point of feature completeness. The two additions described in this article have been in testing for about two years, and released in small steps to the general TEX audience, based on user requests and with attention paid to the design.

If you find problems, bugs, erratic behavior, unclear documentation, or you have further feature suggestions, please contact us at `texlive@tug.org`.

◇ Norbert Preining
Japan Advanced Institute of
Science and Technology
Nomi, Ishikawa, Japan
`norbert (at) preining dot info`
`http://tug.org/texlive`