Siep Kroonenberg
siep@elvenkind.com

configuration

Juggling texmf trees

abstract
Texmf trees can make a TEX installation more maintainable. With creative use of environment variables, it is possible to run different versions and different configurations in different xterm- or console windows.

keywords
texmf trees, configuration, environment variables, file searching

Texmf trees are being cursed for making life unnecessarily complicated. There may be some truth to this, but they also let you break your TEX installation in pieces which can be used and maintained independently:

- Keep your own enhancements in a separate tree, which will be untouched when you upgrade the main tree.
- Give a fast-changing package such as Context a tree of its own. If you want to, you can run two Context versions side-by-side.
- Keep your own installation compact but hook up the TEX Live cd when you need some esoteric feature.
- Unhook your local tree(s) to check whether your journal submission compiles on a vanilla TEX system.

Configuration with TEXMFCNF and `texmf.cnf`

The core components of a TEX system all find their files by the same method: they read one or more `texmf.cnf` configuration files, which, among other things, tell them where to find their files.

TEX data files (macros, format files, fonts etc.) are stored in one or more `texmf` directory trees, which have a more or less fixed internal organization. TEX is configured to always search the current directory for your own files.

teTeX and TEX Live are designed to run 'out of the box'. All the programs have a compiled-in default location for looking for configuration files *relative to their own location*. If you just add the appropriate binaries directory to your path then you will have a runnable system without further manual configuration. Let us see how this magic is accomplished:

This is the relevant fragment of the `texmf.cnf` of the current (january 25 2002) teTeX beta; the version for the TEX Live 6 cd is very similar:

```
% The main tree, which must be mentioned in $TEXMF, below:
TEXMFMAIN = $SELFAUTOPARENT/texmf
% A place for local additions to a "standard" texmf tree.
TEXMFLOCAL = $SELFAUTOPARENT/texmf-local

% User texmf trees can be catered for like this...
HOMETEXMF=$HOME/texmf
```

```
% A place where texconfig stores modifications (instead of the
% TEXMFMAIN tree). texconfig relies on the name, so don't change it.
VARTEXMF = $SELFAUTOPARENT/texmf-var

% Now, list all the texmf trees. If you have multiple trees,
% use shell brace notation, like this:
TEXMF = {$HOMETEXMF,!!$VARTEXMF,$TEXMFLOCAL,!!$TEXMFMAIN}

% Where to look for ls-R files.  There need not be an ls-R in the
% directories in this path, but if there is one, Kpathsea will use it.
TEXMFDBS = $TEXMF
```

That is, there are four `texmf` trees configured: the main tree, `TEXMFMAIN`; two trees for local and user files, and one tree for generated files such as format files or font bitmaps.

The division in 'local' and 'user' makes sense for a shared TEX installation but if you have a TEX installation just for yourself you may want to dispense with one or the other, or reserve one of them for a special purpose.

The order in which `TEXMF` enumerates the trees determines the order in which trees are being searched. If you don't like a version of a file in the main tree then you can put another version in the local or home tree, which will be found before the original version.

The '!!' in front of two of the four tree names indicates that these directories should be searched *only* via the corresponding filename database, which bears the name `ls-R` and resides at the root of the tree. For the other trees, which presumably aren't very large, ordinary file searching is allowed[1].

The variable `SELFAUTOPARENT` gets its value dynamically: if the binaries are in `/usr/local/teTeX/bin/linux`, then `SELFAUTOPARENT` will be the grandparent directory, i.e. `/usr/local/teTeX`. As long as the texmf trees and the binaries are in the same relative position, TEX will be able to find its files.

`texmf.cnf` is thoroughly commented. I recommend that you study it closely. You can find additional information in the kpathsea documentation, which should be present on your system in pdf- and other formats. Also study the structure of the (main) texmf tree. Your local tree doesn't have to be this elaborate, though.

Using environment variables for versioning

For a harddisk installation, you may simply accept the directory structure as it is configured. For a cd installation however, the default value for `VARTEXMF` and `TEXMFLOCAL`, `$SELFAUTOPARENT/texmf-var` and `$SELFAUTOPARENT/texmf-local`, point to directories on the cd so you'll probably want to change them. You can simply overrule them by setting environment variables:

```
VARTEXMF=/var/tmp/texmf-var
export VARTEXMF
```

for Unix (Bourne-compatible shell), or

```
set VARTEXMF=/var/tmp/texmf-var
```

1. The reason not to search on disk is efficiency, but if you disallow searching on disk then you *must* maintain a filename database. The command for this is `mktexlsr`. Without parameters, the command will try to (re)generate the `ls-R` file for all trees listed in TEXMFDBS. You can also specify a texmf tree as parameter.

for DOS/Windows.

But you may have all sorts of reasons to want the trees in other locations than the preconfigured ones.

Context users may appreciate the possibility to reserve an entire tree, say the local tree, for Context. This greatly simplifies upgrading. It also makes it easy to have different versions around:

```
set TEXMFLOCAL=c:/cnstable
path ...;c:\cnstable\context\perltk;...
```

activates the stable version, and

```
set TEXMFLOCAL=c:/cnbeta
path ...;c:\cnbeta\context\perltk;...
```

activates the latest beta. You may not even need to add `context/perltk` to your path, depending on your TEX version.

One thing to watch out for: create subdirectories `cnstable/web2c` and `cnbeta/web2c`, and immediately move or copy your freshly generated Context format files there, otherwise they will overwrite each other.

So now you can switch back and forth between Context versions simply by changing an environment variable.

Hooking and unhooking trees

TEX installations tend to be unpleasantly big. You may not be using a lot of fancy features yourself but you may have colleagues who like to use every package they can get their hands on. A solution: add a cd-based texmf tree, from e.g. the TEX Live- or the 4TeX5 cd, to compile their papers.

In this example I use a second `texmf.cnf` file consisting of just the lines

```
TEXMFCD=d:/texmf
TEXMF = {$HOMETEXMF,!!$VARTEXMF,$TEXMFLOCAL,!!$TEXMFMAIN,!!$TEXMFCD}
```

So the cd tree is added last, and '!!' ensures that the filename database is used. The first line could have been replaced with an environment variable. In fact, if `TEXMFCD` is defined as an environment variable then the setting in this file won't take effect.

To make sure that TEX reads the new `texmf.cnf`, add its directory to the `TEXMFCNF` environment variable, e.g.:

```
set TEXMFCNF=<directory of the new texmf.cnf>;c:/texmf/web2c
```

This works! This is what happens: TEX will first read the environment, then the new `texmf.cnf` and finally the original `texmf.cnf`. *Once a variable is set, subsequent settings won't overrule it.* So the setting for `TEXMF` in the new `texmf.cnf` will remain in effect but we let the original `texmf.cnf` worry about all the other settings.

Unhooking trees is exactly analogous: create a `texmf.cnf` file with a line

```
TEXMF = {!!$VARTEXMF,!!$TEXMFMAIN}
```

and make sure, as above, that the new `texmf.cnf` is read before the main one.

As indicated at the beginning, when you are about to send your paper elsewhere then this is a good way to check whether your paper compiles on a vanilla system. If it doesn't, copy stuff from the local tree(s) to your working directory until it compiles cleanly.

Testing with kpsewhich

Be sure to test whether the right versions will be picked up from the right trees. Type e.g.

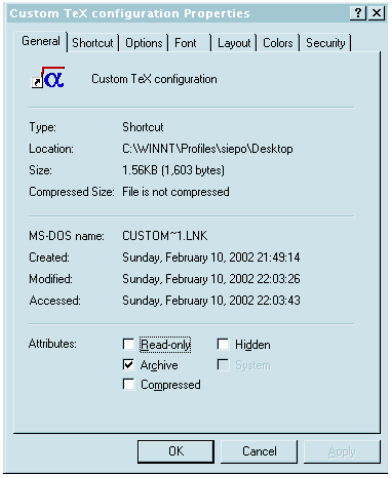
```
kpsewhich cont-en.efmt
```

and you'll see which version of the Context format file is actually going to be used.

Using scripts

Of course, you'll want to put these environment settings into scripts or batchfiles.

Under DOS and at least some versions of Windows, you can set environment variables in a batchfile, and they will stay set after the batchfile has run its course. Under more recent versions of Windows, this may no longer be the case. Anyhow, you can still specify an initialization batchfile for a custom console window:



The icon and the name will show up in the titlebar of the console window.

Under Unix, environment settings defined in a shell script will certainly get lost, unless you 'source' your script, which we shall name `texcd`:

```
source texcd
```

This lets the script run in the current process instead of in a child process. With Bash, you can write '.' instead of 'source':

```
. texcd
```

If you still don't like the extra syntax, you can define an alias:

```
alias tcd='source texcd'
```

All terminal emulators that I know of also let you specify a title to be displayed on the titlebar. If you use bash, try the following:

```
PROMPT_COMMAND="echo -e -n \"\033]0;cdtex \$PWD \07\""
```

to get the tag 'cdtex' and the current directory on your titlebar.