

installation and configuration

Building a T_EX installation for distribution

Siep Kroonenberg
Kluwer Academic Publishers
Dordrecht
email siepo@cybercomm.nl

keywords

Texmf tree, Windows, Linux, Perl, Configuration

At Kluwer Academic Publishers, KAP in short, we use T_EX for typesetting journals. Since there are obvious advantages to using a standardized distribution, we provide our typesetters with one on CD. This way we can be sure that the right fonts and stylefiles are available, and it also eliminates much of the guesswork when we need to investigate technical problems.

The installation is based on a Web2c T_EX implementation. The binaries are from fpT_EX [1], Fabrice Popineau's Web2c T_EX implementation for Win32 platforms. There are two texmf trees; the main tree is based on a recent texmf tree from Thomas Esser's teTeX [2], and the 'local' tree contains some additions from various sources: the Kluwer stylefiles, some commercial fonts and some additional packages. I maintain a third tree for personal and internal use. In this tree I put NTG stuff and development versions of the Kluwer styles.

Maintenance and backups

The main argument for using multiple trees is simpler maintenance: updating major standard packages requires downloading a new texmf tree from Thomas Esser's site and applying a small number of fixes. Our own customizations reside in another tree and remain unaffected.

The combined size of the full zipped distribution is roughly 50MB. By far the largest chunk is the main texmf tree, which can easily be recreated from online sources and therefore doesn't require backing up. The remaining pieces are small enough to keep some old versions around: more for my own scripts, less for collections of downloaded material. As an additional safeguard I keep an informal log of my work, in which I write down what I got from where, among other things.

Linux and Windows

I use the T_EX installation both under Linux and under versions of Windows. I'll explain below how I let both plat-

forms share configuration files. Sharing format files is possible only if the binaries of both platforms have been built from matching sources. At the moment, this precludes the use of the latest versions of the binaries.

I much prefer Linux, but some things have to be done under Windows. Moving back and forth between Windows and Linux is a real hassle; because of the differences in file systems and textfile formats, textfiles may acquire DOS line endings, files may get their executable flags set, or capitalization of filenames may change. This is not only untidy but can also cause real trouble. The zip and unzip programs have some options to fix capitalization and line endings, but this is only a partial remedy.

A much more convenient solution is to maintain the T_EX installation on a Linux system which runs Samba to make T_EX available to Windows. If VMware [3] is installed, the Windows system may even be a virtual machine running under Linux on the same physical machine. But I haven't yet persuaded my boss to provide me with these goodies. See also the screenshot at the end.

TEXMF.CNF: cascaded configuration

For those readers who haven't delved into the mysteries of Web2c configuration, here a quick summary:

Most configuration of a Web2c T_EX system is done in a file `texmf.cnf`, which is consulted by all Web2c components. There is a default location for `texmf.cnf` relative to the location of the executables, so that a standard Web2c installation will run 'out of the box' without any manual configuration. These are a couple of settings from the 'main' `texmf.cnf`:

```
TEXMFMMAIN = $SELFAUTOPARENT/share/texmf
TEXMF = !!$TEXMFMMAIN
```

The first line defines the location of the main texmf tree relative to the T_EX executables, the second line specifies that this tree is the only tree (the exclamation marks mean that no searching outside the ls-R file database should be done).

But it is also possible to specify the location of `texmf.cnf` in an environment variable, or even to specify several directories. The `texmf.cnf` files in all the specified directories will be read, earlier settings taking precedence over later ones. So you can leave the original `texmf.cnf` untouched, and create a new higher-precedence one containing merely the settings which need to be changed. I

created `texmf.cnf` files for the local tree and for the private tree, so that the private tree can be added to my \TeX environment merely by changing the `TEXMF` environment variable.

The scheme is even more flexible because environment settings can also be used for the variables in `texmf.cnf`. If *e.g.* under DOS/Windows the following environment variables have been set:

```
SET TEXMF= c:/tex/local/texmf/web2c;
    c:/tex/share/texmf/web2c
SET TEXMFLOCAL=c:/tex/local/texmf
SET TEXMFMAIN=c:/tex/share/texmf
```

and in our local `texmf.cnf` we have a line

```
TEXMF = {!!$TEXMFLOCAL;!!$TEXMFMAIN}
```

then both the local and the main tree will be active, the local tree having precedence. This same configuration file can be shared between Windows and Unix since the operating system-dependent names of the root directories are defined outside `texmf.cnf`. Note that, in `texmf.cnf`, semicolons for path separators and forward slashes for directory separators are acceptable to both DOS/Windows and Unix. The `texmf.cnf` file of the third tree defines

```
TEXMF = {!!$KLUDIR,!!$TEXMFLOCAL,!!$TEXMFMAIN}
```

where `KLUDIR` has been defined somewhere else.

A Perl installation script

An installation script involves some user- and some system interaction, and Perl is an obvious choice for the job.

Which Perl. I didn't want to assume or require that Perl be installed, let alone a specific version of Perl. So I included on the installation disk a copy of the executables and modules of the old ActiveWare Win32 implementation of Perl 5.003: it is good enough; it is much smaller than the current Perl [4]; it runs fine without explicit installation, and in the licensing conditions this use of this Perl is explicitly allowed without permission.

A couple of tricks:

Wrapping a Perl script in a batchfile. This is very old hat but I still think it is a neat trick. Perl has a command-line option `-x` which instructs Perl to skip to the first line starting with `#!` (shebang) and containing the word `perl`. A batchfile `install.bat`

```
.\perl -x install.bat
goto endofperl
#!perl
<perl code>
__END__
:endofperl
```

is first read by the command interpreter, which calls Perl with itself as input file, and then jumps to `:endofperl`, skipping over any Perl code. In its turn, Perl skips to its own code because of the `-x` option, and stops reading at `__END__` stepping over any non-Perl code.

The installation script calls Perl as `.\perl` to make sure that the right copy of `perl.exe` gets called. In the actual script there is also a bit more code for the command interpreter than just the call to Perl.

A fake backquote. The Perl used for the installation has a backquote function: the output of a 'back-quoted' command can be assigned to a variable, *e.g.*

```
$pwd = `cd`
```

Unfortunately, it doesn't work under the original release of Windows 95. The following code fakes it (note that there were also some problems with redirection which had to be gotten around):

```
open TEMPBAT, ">$tempbat" ||
    die "can't make tempbat";
print TEMPBAT ($command, " >", $tempout, "\n");
close TEMPBAT;
system ($tempbat);
open TEMPFILE, "<$tempout" ||
    die "can't open tempfile";
my $sysout = <TEMPFILE>;
chomp $sysout;
close TEMPFILE;
```

Dialogs. At an earlier stage, I considered using Perl/Tk but dismissed this – with some regret – because Perl/Tk requires a newer Perl and the presence of DCOM, and that would make the installation procedure a lot more invasive. Instead, I just read user responses using `<STDIN>`. This doesn't allow for fancy user interaction, but will do. I have been told that modules exist, also under Win32, for more sophisticated character-mode dialogs.

Additional issues

Ghostscript. It seems unwise to disregard previously installed copies of Ghostscript. A more or less fool-proof method to detect a preexisting version would require searching all local and network drives. In some cases this might be very slow. Instead, I check the registry and the environment (`GS_LIB` and the search path) and look in some obvious places.

Setting environment variables. I can think of some good arguments for setting environment variables globally

