Managing Multiple TDS Trees

Michael J Downes
American Mathematical Society, Providence, Rhode Island 02904, USA
mjd@ams.org

Abstract

The TDS specification for TEX directory structure provides a good framework for organizing a single TEXMF tree such as the one at the heart of Thomas Esser's teTEX distribution. In practice, however, because of the normal processes of local changes and ongoing upgrades it is often advantageous to set up a more elaborate system of multiple TDS trees. This permits organizing the pieces that have been added above and beyond the original base tree in a way that makes it easier to cope with continuing change.

I could have more accurately indicated the scope of this article by using a longer title: *Managing multiple TDS trees in a* ***web2c****-based TEX system*. But it should be fairly clear how to apply the general principles discussed here to other kinds of TEX systems with similar file searching capabilities.

Introduction

TEX Directory Structure After TEX had been in use for some years people started to notice that the set of files needed by TEX when processing documents was growing rather large and unwieldy. The simplest way for TEX to look for input files was to keep them all in a single directory. But this made maintenance difficult.

In response a volunteer committee analyzed the categories of files involved and drew up a specification known as "TDS": TEX Directory Structure. (See http://tug.org/tds.)

The TDS spec gives a recommended hierarchical structure for collections of TEX-related files in a generic form that is intended to be usable and compatible across the various kinds of computers in common use nowadays: PC/Windows, Macintosh, Unix, and so forth. A **TDS tree** is a tree of directories with each major subcategory of files residing in its own subtree — fonts, macros, and documentation, to name a few.

But a TDS tree has everything branching from a single root. In practice it may often be desirable to set up more than one TDS tree, as mentioned in the TDS documentation:

> `... we shall designate the root TDS directory by 'texmf' (for "TeX and METAFONT"). We recommend using that name where possible, but the actual name of the directory is up to the installer. On PC networks, for example, this could map to a logical drive specification such as 'T:'.`
>
> `...`
>
> `...`
>
> `A site may choose to have more than one TDS hierarchy installed (for example, when installing an upgrade).`

In fact, my experience suggests that in most cases it is useful to set up quite a few trees, branching freely whenever you find a key difference in usage or upgrade policy, as long as your TEX system can support it. For example, consider this quote from a recent message on the teTEX mail list:

> `(We're even using one and the same nfs archive for three platforms, Solaris, Linux and W2k. Only trouble is you cannot update executables for one platform without putting the *@:format files in special local places.)`

A solution for such a problem would be to make a separate, very sparse TDS tree that only holds format files (and any other files that are tied to specific binaries). I would probably put the binaries in the same tree, if that could be done without causing extra work elsewhere.

TEX systems: web2c, teTEX, MiKTEX, etc. Installing a TEX system The TEX Users Group has a good listing of TEX systems for various platforms at http://www.tug.org/interest.html. I am going to place my discussion of TDS trees within the framework of a TEX system called teTEX, which

is layered in turn on top of another system called **web2c**. One of the most important elements in **web2c** is a library called `kpathsea`, designed as a tool for other programs to use when they need to efficiently search a tree containing a large collection of files, and in particular to provide ways for users to specify search rules that reflect the needs of a given application. Thus, the versions of TEX, METAFONT, `dvips`, `bibtex`, and other programs included in the **web2c** distribution are all adapted to use the `kpathsea` library.

The use of the `kpathsea` library is the critical factor that determines the exact syntax and configuration methods one uses when making arrangements to have TEX search multiple TDS trees. Other TEX systems such as MiKTEX or TrueTEX that support recursive searching through a TDS tree differ in various details, but many of the general principles will be the same.

The **web2c** guys, notably **Karl Berry** & **Olaf Weber** have pulled the source code for the programs TEX, METAFONT, `dvips`, `xdvi`, into a coherent collection that compiles easily and plugs in the `kpathsea` path searching library where applicable. (Thank you, Karl and Olaf, for all your hard work!)

The teTEX guys, notably **Thomas Esser**, have built a big standard collection that includes **web2c** as a base, but (a) is designed to simplify installation for ordinary users who don't necessarily want to have all the sources and don't plan to make a practice of recompiling things; and (b) includes a rather comprehensive set of commonly needed runtime files, which makes the programs easier to use after they are installed. (Thank you, Thomas, for all your hard work!)

Documentation

Much of the most useful documentation for **web2c** and `kpathsea` is in "Info" form, which can be read with Emacs or with the standalone `info` application. I found the following docs especially useful:

web2c: `web2c.info`

kpathsea: `kpathsea.info`

dvips: `dvips.info`

xdvi: man page for `xdvik`

teTEX: See `TETEXDOC.*` and `teTeX-FAQ`:

```
TEXMF/doc/tetex/teTeX-FAQ
TEXMF/doc/tetex/TETEXDOC.dvi
```

If your TEX system is teTEX, you can use the command

```
texdoc TETEXDOC
```

to automatically find the `.dvi` file and start up a viewer (probably xdvi).

When I tried `texdoc teTeX-FAQ`, however, it didn't work. You may have better luck. My attempt was with teTEX 1.0.

Recursive directory searching For programs whose file searching is based on the `kpathsea` library,

- A 'path' is the name of a directory (aka folder), which may have several components in its name, one for each level.
- In the `texmf.cnf` file the generic separator char between components of a path is slash `/`.
- In a list of multiple paths, the generic separator char between paths is semicolon `;`. (On Unix systems you would normally see colon `:` here.)
- Trailing `//` on a directory name means to search subdirectories (recursively).
- A leading `!!` means to look only in the `ls-R` database, never on the disk.
- A leading/trailing/doubled `;` in the paths will be expanded into the compile-time default path.

Use of filename DB to speed up file searches The `kpathsea` library supports the use of a "filename database" to speed up file searches. (This database is the referent of texhash, mktexlsr, ls-R files and so on.) It is a simple lookup table that gives the location of each file in a TDS tree.

In practice, using the DB seems to cause users a lot of trouble because it is easy to forget to update the DB after adding files. Because most systems nowadays are fast enough that the search time is negligible for even a fairly large collection of files, I would recommend in most cases to freeze the TEXMFMAIN tree and use the DB lookup for TEXMFMAIN but *not for any other trees*. Whether this recommendation is a good idea in your situation depends on the following general principles:

1. Use the DB for TEXMF trees that don't change.
2. Use the DB for TEXMF trees where file access is slow (e.g., across a slow network).

For example, consider the Usenet post shown in Figure 1.

Using an alternate `texmf.cnf` file If you want to specify a different arrangement of TEXMF trees than the one in the out-of-box cnf file (and in practice this is almost always going to be the case), you have two choices:

1. Edit the `texmf.cnf` file in the TEXMFMAIN tree

```
From: Peter Neergaard <turtle@cs.bu.edu>
Subject: Re: Why does miktex need to refresh data base
Newsgroups: comp.text.tex
Date: 16 Jul 2001 12:24:59 -0400

On July 16, 2001, Robin Fairbairns wrote:

RF> consider the following example from my texmf.cnf file.

RF> TEXMF = {$HOMETEXMF,!!$TEXMFLOCAL,!!$TEXMFMAIN,!!$TEXMFEXTRA}

RF> the !! indicators mean that ls-R search is obligatory.
[...]

In some cases you might furthermore need to set TEXMFDBS which
specifies where tetex looks for ls-R files.

On my work, I am actually troubled by the fact that most of the file
systems are NFS mounted.  I did not get a decent speed of file
searching before I started building ls-R files for my private
hierachies.
```

Figure 1: Some discussion of file searching speed

2. Set the environment variable TEXMFCNF to point to a different location (just give the directory name, you don't need to give the full file name)

Obviously, if you do not have privileges to change files in the TEXMFMAIN tree, you will need to resort to the latter option.

To ensure that your cnf file will be read instead of the system-wide cnf file, define the environment variable TEXMFCNF in your shell startup file (`.cshrc`, `.profile`, or the equivalent):

```
# for .cshrc:
setenv TEXMFCNF /home/abc/texmf/web2c
```

The first three lines of the standard teTEX `texmf.cnf` file at the present time look like this:

```
texmf.cnf -- original runtime path
   configuration file for \fn{kpathsea}.
(If you change or delete 'original'
  on the previous line, the
distribution won't install its
  version over yours.)
```

If you use an alternate `texmf.cnf` file, outside the TEXMFMAIN tree, the comment in lines two and three is irrelevant. You don't have to change anything.

Single user setup For a typical single user situation, I suggest three trees: MAIN, PLUS (added-public), and PRIVATE (added-private), and using file-database lookup only for the MAIN tree, which is kept frozen. The distinction between public and private is simply this: Put a file into the PRIVATE tree if it is not publicly released somewhere on the World-Wide Web for all to use. This could include

- packages or fonts that you created yourself
- slightly modified versions of publicly available packages, that you use yourself but that are not intended for any wider use.
- packages or fonts that were sent to you privately by a colleague
- old versions of packages that are no longer available from CTAN (but if this becomes a major component of your system you will probably want to add another OLDVERSIONS tree; see the "**Old Versions**" section below.

So suppose you install version 1.2 of teTEX. The MAIN tree will be the TDS tree as given in the teTEX distribution, out of the box, with absolutely no changes. The PLUS TDS tree will be for any packages that you fetch from CTAN as needed, i.e., whenever you go to use a package and find that it is not present in the MAIN tree. The PRIVATE TDS tree is for any others that are not readily obtainable from the WWWW.

In the `texmf.cnf` file, use the `!!` prefix (DB lookup) only for the TEXMFMAIN tree. Suppose that you have: small

This example shows the top of a `texmf.cnf` file as it would look if you stripped it down to bare essentials in an attempt to understand it before beginning to add more TDS trees. The standard `texmf.cnf` file that comes with teTEX includes sample definitions for TEXMFLOCAL and VARTEXMF as shown here, commented out.

```
% texmf.cnf -- runtime path configuration file for kpathsea.
%
% The main tree, which must be mentioned in $TEXMF, below:
TEXMFMAIN = /usr/local/teTeX/texmf

% A place for local additions to a "standard" texmf tree.  For example:
%   TEXMFLOCAL = /usr/local/teTeX/texmf.local

% If defined, teTeX's texconfig stores modifications here (instead of the
% TEXMFMAIN tree).
%   VARTEXMF = /usr/local/teTeX/texmf-var
  ...

TEXMF = !!$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
```

Figure 2: A minimal `texmf.cnf` file

```
TEXMFMAIN = /usr/local/teTeX/share/texmf
TEXMFPLUS = /usr/local/texmf/plus
TEXMFPRIVATE = /usr/local/texmf/private
```

(Note the absence of "teTeX" in the latter two.) Then you would define `$TEXMF` as

```
TEXMF =
 {$TEXMFPRIVATE;$TEXMFPLUS;!!$TEXMFMAIN}
```

The capabilities of other systems in this respect can vary, of course. The impression I have from looking at the MiKTEX documentation is that multiple TDS trees are supported but at the present time you do not have the option of turning the database lookup on or off for different trees. If this is true then you would have to run the MiKTEX command that updates the database every time you add a new file, regardless of which tree it goes into. (See http://www.miktex.org/.)

Multi-user setup For a typical multi-user situation, I suggest a set of TDS trees that is basically analogous to the single-user situation but with one or two additional considerations. If your company's name is Bingle Publishing Enterprises, then use MAIN and PLUS trees as before but define a BINGLE tree instead of PRIVATE. You may also want to consider using a VAR tree and, possibly, an OLD tree.

TEXMFMAIN The basic TDS tree provided by teTEX

TEXMFPLUS Additional packages not found in MAIN but needed for Bingle work.

TEXMFBINGLE Additional packages needed for Bingle work but not publicly released (items created in-house or otherwise privately obtained).

VARTEXMF This is the tree alluded to in the `texmf.cnf` file, which is intended to hold format files and font files generated at your site. If you use this tree it is also a good place to put the Bingle version of `texmf.cnf`.

TEXMFOLD A tree to hold obsolete versions of packages that may still need to be available for some of your documents. Or maybe more than one tree (see **Old Versions**).

Then your normal definition of TEXMF would be

```
 {$TEXMFBINGLE,$VARTEXMF,
       $TEXMFPLUS,!!$TEXMFMAIN}
```

and when processing documents that need something from the TEXMFOLD tree, you would change the definition of TEXMF in some appropriate way (it might depend on whether or not any filenames

```
% Earlier entries (in the same or another file) override later ones, and
% an environment variable foo overrides any texmf.cnf definition of foo.
%
% All definitions are read before anything is expanded, so you can use
% variables before they are defined.
%
% If a variable assignment is qualified with '.PROGRAM', it is ignored
% unless the current executable (last filename component of argv[0]) is
% named PROGRAM.  This foo.PROGRAM construct is not recognized on the
% right-hand side. For environment variables, use FOO_PROGRAM.
%
% Which file formats use which paths for searches is described in the
% various programs' and the kpathsea documentation.
%
% // means to search subdirectories (recursively).
% A leading !! means to look only in the ls-R db, never on the disk.
% A leading/trailing/doubled ; in the paths will be expanded into the
%   compile-time default. Probably not what you want.
%
% You can use brace notation, for example: /usr/local/{mytex:othertex}
% expands to /usr/local/mytex:/usr/local/othertex.  Instead of the path
% separator you can use a comma: /usr/local/{mytex,othertex} also expands
% to /usr/local/mytex:/usr/local/othertex.  However, the use of the comma
% instead of the path separator is deprecated.
%
% The text above assumes the path separator is a colon (:).  Non-UNIX
% systems use different path separators, like the semicolon (;).
```

Figure 3: Comments from the teTEX `texmf.cnf` file

in TEXMFBINGLE shadow filenames in the TEXMFOLD tree).

Feel free to choose your preferred tree names

There is nothing sacrosanct about the TDS tree names that I have suggested. The only thing you have to worry about is whether some particular tree names get special handling in your TEX system; this is true in teTEX for VARTEXMF (and VARTEXFONTS), but to the best of my knowledge not for anything else. So instead of MAIN PLUS PRIVATE, you might prefer

`TEXMFMAIN TEXMFEXTRA TEXMFLOCAL`

or

`TEXMFBASE1 TEXMFBASE2 TEXMFBINGLE`

And I can think of a few more possible names for the first auxiliary tree, such as TEXMFADD or TEXMFAUX or TEXMFPUBLIC or ...

SELFAUTOPARENT or SELFAUTODIR?

If you look at a real `texmf.cnf` file you may see that the definition of TEXMFMAIN refers to some puzzling variables SELFAUTOPARENT or SELFAUTODIR. There are two ideas here:

1. When TEX or another web2c program starts up, it should be able to decide where to look for the cnf file by checking its own location — i.e., if the full path for the `dvips` program is `/usr/local/teTeX/bin/dvips`, then by default `dvips` will look for the cnf file in `/usr/local/teTeX/texmf/web2c`.
2. The full path of programs such as `dvips` may vary according to an installation option: whether to include an extra level that indicates the architecture and operating system for which the programs were compiled.

SELFAUTODIR If the executable for TEX is found at

`/usr/local/teTeX/bin/tex`

then when tex first starts up it sets

- `$SELFAUTOLOC` = `/usr/local/teTeX/bin`
- `$SELFAUTODIR` = `/usr/local/teTeX`
- `$SELFAUTOPARENT` = `/usr/local`

These affect all subsidiary searches that refer directly or indirectly to one of the SELFAUTO variables. Your setting of TEXMFMAIN should refer to SELFAUTODIR, e.g.,

```
TEXMFMAIN = $SELFAUTODIR/share/texmf
```

SELFAUTOPARENT But there is a "multiplatform" installation option for web2c, which would have multiple tex executables in places such as:

```
/usr/local/teTeX/bin/sparc-sun-solaris2.7/tex
```

Then

- `$SELFAUTOLOC` = `/usr/local/teTeX/bin/sparc-sun-solaris2.7`
- `$SELFAUTODIR` = `/usr/local/teTeX/bin`
- `$SELFAUTOPARENT` = `/usr/local/teTeX`

In this case your definition of TEXMFMAIN should refer to SELFAUTOPARENT.

Using the SELFAUTO... variables is not mandatory

If it becomes too much trouble to sort all this out, you can always fall back on giving the full directory name explicitly in the `texmf.cnf` file and don't try to make the SELFAUTO idea work.

```
TEXMFMAIN = /usr/local/teTeX/share/texmf
```

It won't make any practical difference unless and until you do something such as copy the tree to another machine, and for whatever reason don't put it in the same location. Then you would need to edit the `texmf.cnf` file to match.

VARTEXMF? VARTEXFONTS?

The option of generating PK files on demand, which has been around for some years now, brings with it some questions about where to put the files that are generated. On a multi-user system, ordinary users cannot normally create files in the main TEXMF tree. If I understand correctly, the idea of setting up a VARTEXFONTS area for the generated files, which was the earliest approach, was later generalized to the idea of a VARTEXMF tree to hold several kinds of locally generated files: `pk` files, `tfm` files, `tex` format files, and configuration files.

The evolution and documentation of this idea was not particularly easy for me to follow, and I believe that if you have a PRIVATE or BINGLE tree it can serve as the location of all the files that would otherwise go into the VARTEXMF tree. But I have not tested this hypothesis extensively enough to rule out the possibility that the name "VARTEXMF" gets special treatment either by the `kpathsea` library or by some of the teTEX configuration scripts.

At the present time I am using a separate VARTEXMF tree, setting

```
VARTEXFONTS = $VARTEXMF/fonts
```

and following the related recommendations given in the `texmf.cnf` file that comes with teTEX.

For some related information see also the "Filesystem Hierarchy Standard" at http://www.pathname.com/fhs/.

HOMETEXMF

For individual users on a multi-user system to add personal bits and pieces, there is a convention of using a variable named `$HOMETEXMF`.

1. The sysadmin might enable use of `$HOME/texmf` for everybody by defining

   ```
   HOMETEXMF = $HOME/texmf
   ```

   and including `$HOMETEXMF` in the definition of `$TEXMF`, in the system-wide `texmf.cnf` file. Then anything that you put in that tree immediately becomes usable for you. (Assuming that the sysadmin did not use the `!!` prefix for the `HOMETEXMF` tree.)

   [One caveat: For the sysadmin herself, it might be better *not* to have `HOMETEXMF` included in `TEXMF` when doing TEX system administration tasks.]
2. Or if not, you can put files into `$HOME/texmf` anyway:
   (a) Define the environment variable `TEXMFCNF` = `$HOME/texmf/web2c` in your shell login script. This ensures that tex and other programs will use your personal cnf file instead of the system-wide one.
   (b) Make a small `texmf.cnf` file in that directory that defines `TEXMF` to include `HOMETEXMF`. This ensures that all other kinds of files will be found in the specified search order. You can also copy other lines from the system-wide `texmf.cnf` file if further customization is needed.

Upgrades

Upgrading teTEX Suppose that you hear teTEX 2.0 has been released and you are eager to install it on your system to replace your current version, 1.4. What will your plan of action be?

Naturally you will want to install version 2.0 in an entirely separate tree where it can be tested for a while, so that you feel reasonably sure that teTEX 2.0 will produce the same results (or, possibly, better) with your existing document base.

Suppose that you currently have:

```
TEXMFMAIN = /usr/local/teTeX/texmf
VARTEXMF = /usr/local/texmf/var
TEXMFPLUS = /usr/local/texmf/plus
TEXMFBINGLE = /usr/local/texmf/bingle
```

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

You could install the new teTEX distrib at `/usr/upgrd/teTeX-2.0` (say) and for testing purposes set one environment variable:

```
TEXMFCNF=/usr/upgrd/texmf/var/web2c
```

With luck all other adaptations can be handled by editing the new `texmf.cnf` file that you make in the indicated location, primarily defining `TEXMFMAIN` and `VARTEXMF` differently (the latter to hold format files).

```
TEXMFMAIN = /usr/upgrd/teTeX/texmf
VARTEXMF = /usr/upgrd/texmf/var
```

Old Versions

The Comprehensive TEX Archive Network does not attempt to systematically archive old versions of software. At any given time, you may see the previous version and the current version of a particular package, or the current version and a beta version of the next release.

Therefore, if you have a need to run different documents with different versions of a given package, you will need to take it on yourself to keep a copy of all the old versions that you want.

How do you arrange that a particular document gets the appropriate version of a particular package? Well, if you thought things were complicated before ....

If the number of different package/version combinations remains relatively small, you can use additional TDS trees for this purpose. Let's suppose, for example, that in addition to the latest release of LATEX you would like to be able to run older releases on demand, one for each of the years from 1996 through 2000.

Set up a separate TDS tree for each release of LATEX (here the phrase "tree farm" comes to mind).

```
/usr/local/texmf.old/latex1996
/usr/local/texmf.old/latex1997
/usr/local/texmf.old/latex1998
/usr/local/texmf.old/latex1999
/usr/local/texmf.old/latex2000
```

Make links in the teTEX bin directory so that `latex1996`, etc., can be used from the command line. Then in the `texmf.cnf` file put

```
TEXMFOLD=/usr/local/texmf.old
TEXINPUTS.latex1996=
  $TEXMFOLD/latex1996/tex/latex//
  ;$TEXMF/tex/{latex,generic}//
```

and similarly for the others.

Searching

Examining search paths To check what the current value of `TEXMF` is:

```
kpsexpand '$TEXMF'
```

This shows the end result after expanding all intermediate variable references.

Try `kpsexpand '$TEXMFCNF'` and look at the `texmf.cnf` file found there if you want to examine the variable settings in detail.

Checking the TEXINPUTS search path This gives you the generic search path:

```
kpsexpand '$TEXINPUTS'
```

But that one is seldom used in practice. Instead, one gets a format-specific value of TEXINPUTS from the `texmf.cnf` file. To find out what it is:

```
kpsewhich --progname=latex
   --expand-var='$TEXINPUTS'
```

This is the most accurate way to check (short of turning on the debugging flags) because it takes into account any environment variables that may be set.

Precedence of environment variables For a program such as LATEX, the search path is taken from the first of the following possibilities that is discovered to be non-null:

- `$TEXINPUTS_latex` (env variable)
- `$TEXINPUTS` (env variable)
- `$TEXINPUTS.latex` (cnf file)
- `$TEXINPUTS` (cnf file)

This is for a recent teTEX (1.x). Some options might not be searched for earlier versions.

The simplest way of temporarily changing the search path of a given command is still to set `TEXINPUTS`. For example (if you are using sh/bash/ksh):

```
TEXINPUTS=$HOME/mystyles: latex xyz
```

But remember to put the extra colon on the end that indicates "append the usual value here".

Debugging search paths To scrutinize in utmost detail the actual search path that is used for, say, running latex on a particular file:

```
KPATHSEA_DEBUG=122 latex xyz.tex
```

or equivalently, for `csh` users:

```
(setenv KPATHSEA_DEBUG 122; latex xyz.tex)
```

See `kpathsea.info` for more information on the debugging flags that are available. Some programs, for example xdvi, may have special debugging options in addition to the general ones available thru `kpathsea`. Check their documentation.

Slash versus dot in the TDS tree farm In the past some people have used directory names of the form `texmf.local` for auxiliary TDS trees. My hunch is that it will tend to work better in practice to make maximal use of the filesystem hierarchy, if you have enough TDS trees to call it a tree farm. So instead of

```
/ams/texmf.plus
/ams/texmf.prod
/ams/texmf.var
```

I am using

```
/ams/texmf/plus
/ams/texmf/prod
/ams/texmf/var
```

Fallback search paths

In the standard teTEX `texmf.cnf` file the `TEXINPUTS` path for LaTeX currently has an interesting part at the end:

```
TEXINPUTS.latex =
  .;$TEXMF/tex/{latex,generic,}//
```

This means search the following trees in order:

```
$TEXMF/tex/latex//
$TEXMF/tex/generic//
$TEXMF/tex//
```

In other words, if a file is not found in the LaTeX tree, and not in the generic tree, go back and search the entire TEX tree (all formats!), including searching again in the latex and generic trees.

This is helpful if you notice that some files from one format are useful for use with other formats (e.g., LaTeX graphics stuff or certain ConTEXt files). But it also makes you vulnerable to other problems, and ultimately I think it is better to look for a different solution. So I would say:

- Don't put that final comma in.
- And by the way it probably should be a more generic semicolon.
- And by the way the slashes don't really come out right unless you do as I say and omit the comma. (Although `kpathsea` manages anyway by ignoring extra slashes if there are too many.)

Expanding a path string

```
kpsewhich --expand-path=
   '$TEXMFMAIN/tex/{latex,generic}'
```

yields:

```
/usr/local/texmf/tex/latex:
  /usr/local/texmf/tex/generic
```

Recursive expansion

```
kpsewhich --expand-path=
  '$TEXMFMAIN/tex/{latex,generic}//'
```

yields all the latex subdirectories first, then all the generic:

```
/usr/local/texmf/tex/latex:
  /usr/local/texmf/tex/latex/base:
  /usr/local/texmf/tex/latex/config:
  /usr/local/texmf/tex/latex/amsfonts:
    ...
  /usr/local/texmf/tex/generic/thumbpdf:
  /usr/local/texmf/tex/generic/ukrhyph
```

Expansion with two TDS trees

```
kpsewhich --expand-path=
  '{$VARTEXMF,$TEXMFMAIN}/tex/{latex,generic}'
```

yields:

```
/bingle/texmf/var/tex/latex:
  /usr/local/texmf/tex/latex:
  /bingle/texmf/var/tex/generic:
  /usr/local/texmf/tex/generic
```

Recap

Suppose your company name is Bingle. Here in a nutshell is a typical approach you could use.

Environment:

```
export TEXMFCNF=/usr/local/texmf/bingle/web2c
```

`texmf.cnf` file:

```
TEXMFMAIN = /usr/local/teTeX/texmf
TEXMFPLUS = /usr/local/texmf/plus
VARTEXMF = /var/texmf
TEXMFBINGLE = /usr/local/texmf/bingle
TEXMF = {$TEXMFBINGLE,$VARTEXMF,
          $TEXMFPLUS,!!$TEXMFMAIN}
...
TEXINPUTS.context =
   .;$TEXMF/tex/{context,generic}//
```

- Put all upgrades and new packages that you get from elsewhere into the `TEXMFPLUS` tree.
- Put packages in `TEXMFBINGLE` if they are not publicly available. (This might include old versions of public packages that you want to keep.)
- Put format files and config files in `VARTEXMF`.
- Debug by setting `$KPATHSEA_DEBUG=122`.

References

Most of these links are already mentioned earlier in the text, but I gather them here for convenience.

Appendix M: Some MikTEX notes

To give a flavor of the differences between TEX systems when it comes to search paths, I show some fragments from the MikTEX documentation in Figures 4 and 5.

```
[LaTeX]
Input Dirs=.;c:\users\me//;%R\tex\latex//;%R\tex\generic//
```

The direct teTEX equivalent would be

```
TEXINPUTS.latex=!!/users/me//;!!$TEXMF/tex/latex//;!!$TEXMF/tex/generic//
```

although the latter two parts would more usually be combined using brace notation:

```
TEXINPUTS.latex=!!/users/me//;!!$TEXMF/tex/{latex,generic}//
```

Figure 4: From the MiKTEX documentation: the search path for LATEX.

Which is the best directory to keep `.sty` files where MiKTEX can find them?
Each new LATEX package should be installed in the folder `tex\latex\package` relative to the local `TEXMF` folder (usually `C:\localtexmf\`).
Example: suppose you want to install the package `thesis-ex` which consists of the style file `thesis-es.sty`:
Create a new folder:

```
C:\> mkdir C:\localtexmf\tex\latex\thesis-ex
```

Copy all files (only one in our case) to the new folder:

```
C:\> copy thesis-ex.sty C:\localtexmftex\latex\thesis-ex
```

Refresh the file name database so that MiKTEX finds the new files:

```
C:\> initexmf -u
```

Figure 5: From http://www.miktex.org/faq/index.html

TeX Directory Structure

- http://www.tug.org/tds/
- http://www.ctan.org/tex-archive/tds/standard/tds/tds.html (`.pdf`, `.dvi`, etc)

Filesystem Hierarchy Standard
http://www.pathname.com/fhs/

Web2c documentation

- `/usr/local/teTeX/info/web2c.info` (command-line options!)
- http://www.tug.org/web2c/manual

`Kpathsea` **documentation**

- `/usr/local/teTeX/info/kpathsea.info` (search path syntax!)
- http://www.tug.org/kpathsea/

teTEX: See `TETEXDOC.*` and `teTeX-FAQ`:

```
TEXMF/doc/tetex/teTeX-FAQ
TEXMF/doc/tetex/TETEXDOC.dvi
```

dvips: `dvips.info`

xdvi: man page for `xdvik`

Obtaining a TEX system for your computer
See http://www.tug.org/interest.html.

- **fpTEX** is a free system for Windows 95/98/ME/NT/2000/XP computers that is based directly on **web2c**. http://www.fptex.org/ (Thank you, Fabrice Popineau.)
- **MikTEX** is another one often used on Windows computers. http://www.miktex.org/ (Thank you, Christian Schenk.)
- **Join the TEX Users Group**. Then you will get a TEX Live CD in the mail every year containing several free TEX systems, including the two above, in a ready-to-run or easy-to-install form. (This is also true for most of the other Local User Groups such as Dante, UKTUG, etc.) See http://www.tug.org/texlive.html.

Appendix R: Some Miscellaneous Remarks

- These programs use `kpathsea`: TEX METAPOST, BIBTEX, `dvips [dvipsk]`, `xdvi [xdvik]`, and quite a few others.
- Some programs that don't use `kpathsea`: `ghostscript`, `ghostview`, `latex2html`, `info`, and `man`.
- Drawback of finely subdivided trees as recommended in the TDS spec: It's harder to do group operations across the boundaries. Advantage: It's easier to do operations on the natural subsets.
- I like to put the `.tex`, `x.tex`, etc., files from latex into the generic tree.
- `kpathsea` changes the generic `;` path separator into the Unix equivalent `:` when printing messages to the screen, if running on a Unix system.
- If a directory is listed in `TEXMFDBS` in the `texmf.cnf` then the disk is never searched — I think. Need to be wary of complications here.

Appendix X: xdvi

xdvi search paths The following variables are significant for xdvi. General setup:

`TEXMFCNF` to find `texmf.cnf` and read search path info

`TEXMFDBS` to search for `ls-R` database files

`MIMELIBDIR` for mime types

`MAILCAPDIR` for mailcap file

For dealing with a particular font (e.g., `cmtt10`):

`VFFONTS` search for `cmtt10.vf`

`MKTEXPK` use mktexpk script? 0 false, 1 true

`PKFONTS` search for `cmtt10.300pk`

Generating PK files Here is my weak and superficial understanding of the process, in case it may be of some small help. I have not done a great deal of experimentation with PK file generation.

The config files that affect the generation of PK files are

```
...teTeX/texmf/web2c/texmf.cnf
```

and

```
...teTeX/fontname/*
```

Note that the latter is *outside of the texmf hierarchy.*

If the MF sources were found in one texmf tree, the PK files are written into that same texmf tree. If that `TEXMF` tree is not writable, the generated fonts go into `VARTEXFONTS`. The files in the fontname directory control which subdirectory of the PK tree the PK files go into.