Kpathsea library

for version 3.3.1
May 1999

K. Berry (kb@mail.tug.org)
O. Weber (infovore@xs4all.nl)

1 Introduction

This manual corresponds to version 3.3.1 of the Kpathsea library, released in May 1999.

The library's fundamental purpose is to return a filename from a list of directories specified by the user, similar to what shells do when looking up program names to execute.

The following software, all of which we maintain, uses this library:

- Dviljk (see the '`dvilj`' man page)
- Dvipsk (see section "Introduction" in *Dvips: A DVI driver*)
- GNU font utilities (see section "Introduction" in *GNU font utilities*)
- Web2c (see section "Introduction" in *Web2c: A TeX implementation*)
- Xdvik (see the '`xdvi`' man page)

Other software that we do not maintain also uses it.

We are still actively maintaining the library (and probably always will be, despite our hopes). If you have comments or suggestions, please send them to us (see Section 2.6 [Reporting bugs], page 20).

We distribute the library under the GNU Library General Public License (LGPL). In short, this means if you write a program using the library, you must (offer to) distribute the source to the library, along with any changes you have made, and allow anyone to modify the library source and distribute their modifications. It does not mean you have to distribute the source to your program, although we hope you will. See the files '`COPYING`' and '`COPYING.LIB`' for the text of the GNU licenses.

If you know enough about TeX to be reading this manual, then you (or your institution) should consider joining the TeX Users Group (if you're already a member, great!). TUG produces the periodical *TUGboat*, sponsors an annual meeting and publishes the proceedings, and arranges courses on TeX for all levels of users throughout the world. Anyway, here is the address:

TeX Users Group
P.O. Box 2311
Portland OR 97208-2311
USA
phone: +1 503 223-9994
fax: +1 503 223-3960
email: `tug@tug.org`

1.1 History

(This section is for those people who are curious about how the library came about.) (If you like to read historical accounts of software, we urge you to seek out the GNU Autoconf manual and the "Errors of TeX" paper by Don Knuth, published in *Software—Practice and Experience* 19(7), July 1989.)

[Karl writes.] My first ChangeLog entry for Web2c seems to be February 1990, but I may have done some work before then. In any case, Tim Morgan and I were jointly maintaining it for a time. (I should mention here that Tim had made Web2c into a real distribution long before I had ever used it or even heard of it, and Tom Rokicki did the original implementation. I was using `pxp` and `pc` on VAX 11/750's and the hot new Sun 2 machines.)

It must have been later in 1990 and 1991 that I started working on *TEX for the Impatient*. Dvips, Xdvi, Web2c, and the GNU fontutils (which I was also writing at the time) all used different environment variables, and, more importantly, had different bugs in their path searching. This became extremely painful, as I was stressing everything to the limit working on the book. I also desperately wanted to implement subdirectory searching, since I couldn't stand putting everything in one big directory, and also couldn't stand having to explicitly specify '`cm`', '`pandora`', . . . in a path.

In the first incarnation, I just hacked separately on each program—that was the original subdirectory searching code in both Xdvi and Dvips, though I think Paul Vojta has completely rewritten Xdvi's support by now. That is, I tried to go with the flow in each program, rather than changing the program's calling sequences to conform to common routines.

Then, as bugs inevitably appeared, I found I was fixing the same thing three times (Web2c and fontutils were always sharing code, since I maintained those—there was no Dvipsk or Xdvik or Dviljk at this point). After a while, I finally started sharing source files. They weren't yet a library, though. I just kept things up to date with shell scripts. (I was developing on a 386 running ISC 2.2 at the time, and so didn't have symbolic links. An awful experience.)

The ChangeLogs for Xdvik and Dvipsk record initial releases of those distributions in May and June 1992. I think it was because I was tired of the different configuration strategies of each program, not so much because of the path searching. (Autoconf was being developed by David MacKenzie and others, and I was adapting it to TEX and friends.)

I started to make a separate library that other programs could link with on my birthday in April 1993, according to the ChangeLog. I don't remember exactly why I finally took the time to make it a separate library; a conversation with david zuhn that initiated it. Just seemed like it was time.

Dviljk got started in March 1994 after I bought a Laserjet 4. (Kpathsea work got suspended while Norm Walsh and I, with Gustaf Neumann's help, implemented a way for TEX to get at all those neat builtin LJ4 fonts . . . such a treat to have something to typeset in besides Palatino!)

By spring of 1995, I had implemented just about all the path-searching features in Kpathsea that I plan to, driven beyond my initial goals by Thomas Esser and others. I then started to integrate Web2c with Kpathsea. After the release of a stable Web2c, I hope to be able to stop development, and turn most of my attention back to making fonts for GNU. (Always assuming Micros**t hasn't completely obliterated Unix by then, or that software patents haven't stopped software development by anybody smaller than a company with a million-dollar-a-year legal budget. Which is actually what I think is likely to happen, but that's another story. . .)

[Olaf writes.] At the end of 1997, UNIX is still alive and kicking, individuals still develop software, and Web2c development still continues. Karl had been looking for some time for someone to take up part of the burden, and I volunteered.

2 Installation

(A copy of this chapter is in the distribution file '`kpathsea/INSTALL`'.)

The procedure for Kpathsea (and Web2c, etc.) configuration and installation follows. If you encounter trouble, see Section 2.6.5 [Common problems], page 24, a copy of which is in the file '`kpathsea/BUGS`'.

2.1 Simple installation

Installing TEX and friends for the first time can be a daunting experience. Thus, you may prefer to skip this whole thing and just get precompiled executables: see Section 2.5 [unixtex.ftp], page 17.

This section explains what to do if you wish to take the defaults for everything, and generally to install in the simplest possible way. Most steps here refer to corresponding subsection in the next section which explains how to override defaults and generally gives more details.

By default everything will be installed under '`/usr/local`' and the following discussion assumes this. However, if you already have TEX installed, its location is used to derive the directory under which everything is to be installed.

1. Be sure you have enough disk space: approximately 8 megabytes for the compressed archives, 15MB for sources, 50MB for compilation, 40MB for the (initial) installed system (including library files). See Section 2.2.1 [Disk space], page 4.
2. Retrieve these distribution archives:

   `ftp://ftp.tug.org/tex/texk.tar.gz`
      These are the sources, which you will be compiling.

   `ftp://ftp.tug.org/tex/texklib.tar.gz`
      This is a basic set of input files. You should unpack it in the directory '`/usr/local/share`'; doing so will create a '`texmf`' subdirectory there.

   These archives are mirrored on the CTAN hosts, in the '`systems/web2c`' directory.

   See Section 2.2.2 [Kpathsea application distributions], page 5.
3. When using the default search paths, there is no need to edit any distribution files. See Section 2.2.3 [Changing search paths], page 5.
4. At the top level of the distribution, run '`sh configure`'. (If you have the GNU Bash shell installed, run '`bash configure`'.) See Section 2.2.4 [Running configure], page 7.
5. '`make`'. See Section 2.2.5 [Running make], page 10. If you are using a BSD 4.4 system such as FreeBSD or NetBSD, you may have to use GNU make (often installed in '`/usr/local/bin`'), not the BSD make.
6. '`make install`'. See Section 2.2.6 [Installing files], page 10.
7. '`make distclean`'. See Section 2.2.7 [Cleaning up], page 11.
8. Set up a cron job to rebuild the filename database that makes searching faster. This line will rebuild it every midnight:

```
0 0 * * * cd /usr/local/share/texmf && /bindir/mktexlsr
```

See Section 2.2.8 [Filename database generation], page 11, and Section 3.4 [Filename database], page 34.

9. If you're installing Dvips, you also need to set up configuration files for your printers and make any additional PostScript fonts available. See section "Installation" in *Dvips*. If you have any color printers, see section "Color device configuration" in *Dvips*.
10. The first time you run a DVI driver, a bunch of PK fonts will be built by Metafont via `mktexpk` (and added to the filename database). This will take some time. Don't be alarmed; they will created only this first time (unless something is wrong with your path definitions).

    By default, `mktexpk` will create these fonts in a hierarchy under '`/var/tmp/texfonts`'; it simply assumes that '`/var/tmp`' exists and is globally writable. If you need a different arrangement, see Section 2.2.9.1 [mktex configuration], page 12.

    See Section 2.2.9 [mktex scripts], page 12.
11. For some simple tests, try '`tex story \\bye`' and '`latex sample2e`'. Then run '`xdvi story`' or '`dvips sample2e`' on the resulting DVI files to preview/print the documents. See Section 2.2.10 [Installation testing], page 14.

2.2 Custom installation

Most sites need to modify the default installation procedure in some way, perhaps merely changing the prefix from '`/usr/local`', perhaps adding extra compiler or loader options to work around `configure` bugs. This section explains how to override default choices. For additional distribution-specific information:

- '`dviljk/INSTALL`'.
- See section "Installation" in *Dvips*.
- See section "Installation" in *Web2c*.
- '`xdvik/INSTALL`'.

These instructions are for Unix systems. Other operating-system specific distributions have their own instructions. The code base itself supports Amiga, DOS, OS/2, and VMS.

Following are the same steps as in the previous section (which describes the simplest installation), but with much more detail.

2.2.1 Disk space

Here is a table showing the disk space needed for each distribution (described in the next section). The '(totals)' line reflects the '`texk`' source distribution and '`texklib`'; the individual distributions don't enter into it. Sizes are in megabytes. All numbers are approximate.

Distribution	.tar.gz	Unpacked	Compiled	Installed
dviljk	.9	3.8		
dvipsk	.9	3.2		
xdvik	.7	2.5		
web2c	1.3	5.0		

web	1.9	6.5	-	-
texk	3.8	14.1	43.1	23.5
texklib	3.8	15.0	-	15.0
(totals)	7.6	29.1	43.1	38.5

2.2.2 Kpathsea application distributions

The archive `ftp://ftp.tug.org/tex/texk.tar.gz` contains all of the Kpathsea applications I maintain, and the library itself. For example, since NeXT does not generally support X11, you'd probably want to skip '`xdvik`' (or simply remove it after unpacking '`texk.tar.gz`'. If you are not interested in all of them, you can also retrieve them separately:

'`dviljk.tar.gz`'
: DVI to PCL, for LaserJet printers.

'`dvipsk.tar.gz`'
: DVI to PostScript, for previewers, printers, or PDF generation.

'`web2c.tar.gz`'
: The software needed to compile TEX and friends.

'`web.tar.gz`'
: The original WEB source files, also used in compilation.

'`xdvik.tar.gz`'
: DVI previewing under the X window system.

If you want to use the Babel LaTEX package for support of non-English typesetting, you may need to retrieve additional files. See the file '`install.txt`' in the Babel distribution.

2.2.3 Changing search paths

If the search paths for your installation differ from the standard TEX directory structure (see section "Introduction" in *A Directory Structure for TEX files*), edit the file '`kpathsea/texmf.in`' as desired, before running `configure`. For example, if you have all your fonts or macros in one big directory.

You may also wish to edit the file '`mktex.cnf`', either before or after installation, to control various aspects of `mktexpk` and friends. See Section 2.2.9.1 [mktex configuration], page 12.

You do not need to edit '`texmf.in`' to change the default top-level or other installation *directories* (only the paths). You can and should do that when you run `configure` (next step).

You also do not need to edit '`texmf.in`' if you are willing to rely on '`texmf.cnf`' at runtime to define the paths, and let the compile-time default paths be incorrect. Usually there is no harm in doing this.

The section below explains default generation in more detail.

2.2.3.1 Default path features

The purpose of having all the different files described in the section above is to avoid having the same information in more than one place. If you change the installation directories or top-level prefix at `configure`-time, those changes will propagate through the whole sequence. And if you change the default paths in '`texmf.in`', those changes are propagated to the compile-time defaults.

The Make definitions are all repeated in several `Makefile`'s; but changing the top-level '`Makefile`' should suffice, as it passes down all the variable definitions, thus overriding the submakes. (The definitions are repeated so you can run Make in the subdirectories, if you should have occasion to.)

By default, the bitmap font paths end with '`/$MAKETEX_MODE`', thus including the device name (usually a Metafont mode name such as '`ljfour`'). This distinguishes two different devices with the same resolution—a write/white from a write/black 300 dpi printer, for example.

However, since most sites don't have this complication, Kpathsea (specifically, the `kpse_init_prog` function in '`kpathsea/proginit.c`') has a special case: if the mode has not been explicitly set by the user (or in a configuration file), it sets `MAKETEX_MODE` to `/`. This makes the default PK path, for example, expand into `.../pk//`, so fonts will be found even if there is no subdirectory for the mode (if you arranged things that way because your site has only one printer, for example) or if the program is mode-independent (e.g., `pktype`).

To make the paths independent of the mode, simply edit '`texmf.in`' before installation, or the installed '`texmf.cnf`', and remove the '`$MAKETEX_MODE`'.

See Section 2.2.9.3 [mktex script arguments], page 14, for how this interacts with `mktexpk`.

See Section 2.4 [TeX directory structure], page 15, for a description of the default arrangement of the input files that comprise the TeX system. The file '`kpathsea/HIER`' is a copy of that section.

2.2.3.2 Default path generation

This section describes how the default paths are constructed.

You may wish to ignore the whole mess and simply edit '`texmf.cnf`' after it is installed, perhaps even copying it into place beforehand so you can complete the installation, if it seems necessary.

To summarize the chain of events that go into defining the default paths:

1. '`configure`' creates a '`Makefile`' from each '`Makefile.in`'.
2. When Make runs in the '`kpathsea`' directory, it creates a file '`texmf.sed`' that substitutes the Make value of `$(var)` for a string `@var@`. The variables in question are the one that define the installation directories.
3. '`texmf.sed`' (together with a little extra magic—see '`kpathsea/Makefile`') is applied to '`texmf.in`' to generate '`texmf.cnf`'. This is the file that will eventually be installed and used.

4. The definitions in '`texmf.cnf`' are recast as C `#define`'s in '`paths.h`'. These values will be the compile-time defaults; they are not used at runtime unless no '`texmf.cnf`' file can be found.

   (That's a lie: the compile-time defaults are what any extra `:`'s in '`texmf.cnf`' expand into; but the paths as distributed have no extra `:`'s, and there's no particular reason for them to.)

2.2.4 Running `configure`

Run `sh configure` *options* (in the top-level directory, the one containing '`kpathsea/`'), possibly using a shell other than `sh` (see Section 2.2.4.1 [configure shells], page 7).

`configure` adapts the source distribution to the present system via `#define`'s in '`*/c-auto.h`', which are created from the corresponding '`c-auto.in`'. It also creates a '`Makefile`' from the corresponding '`Makefile.in`', doing '`@`*var*`@`' and '`ac_include`' substitutions).

`configure` is the best place to control the configuration, compilation, and installed location of the software, either via command-line options, or by setting environment variables before invoking it. For example, you can disable `mktexpk` by default with the option '`--disable-mktexpk`'. See Section 2.2.4.2 [configure options], page 7.

2.2.4.1 `configure` shells

Considerable effort has gone into trying to ensure that the `configure` scripts can be run by most Bourne shell variants. Should `sh` run into trouble, your best bet is to use Bash, the GNU shell (see section "Top" in *Bash Features*).

Bourne shell variants for which problems have been reported in the past are:

`ksh`
: Old versions of the Korn shell may fail to handle the scripts. The Korn shell may be installed as '`/bin/sh`' on AIX, in which case '`/bin/bsh`' may serve instead.

`ash`
: Old versions of ash are unable to handle the scripts. Ash is sometimes installed as '`/bin/sh`' on NetBSD, FreeBSD, and Linux systems. '`/bin/bash`' should be available for those systems, but might not be part of a default installation.

Ultrix `/bin/sh`
: '`/bin/sh`' under Ultrix is a DEC-grown shell that is notably deficient in many ways. '`/bin/sh5`' may be necessary.

2.2.4.2 `configure` options

For a complete list of all `configure` options, run '`configure --help`' or see section "Running `configure` scripts" in *Autoconf*, (a copy is in the file '`kpathsea/README.CONFIGURE`'). The generic options are listed first in the '`--help`' output, and the package-specific options come last. The environment variables `configure` pays attention to are listed below.

Options particularly likely to be useful are '`--prefix`', '`--datadir`', and the like; see Section 2.2.4.4 [configure scenarios], page 9.

This section gives pointers to descriptions of the '`--with`' and '`--enable`' options to `configure` that Kpathsea-using programs accept.

'`--without-mktexmf-default`'
'`--without-mktexpk-default`'
'`--without-mktextfm-default`'
'`--with-mktextex-default`'
: Enable or disable the dynamic generation programs. See Section 2.2.9.1 [mktex configuration], page 12.

'`--enable-shared`'
: Build Kpathsea as a shared library, and link against it. Also build the usual static library. See Section 2.2.4.5 [Shared library], page 9.

'`--disable-static`'
: Build only the shared library. Implies '`--enable-shared`'.

'`--enable-maintainer-mode`'
: Enables make targets that are useful for the maintainer and likely to be a pain for anyone else; the makefiles created when this option is enabled may not work at all for you. You have been warned.

2.2.4.3 `configure` environment

`configure` uses the value of the following environment variables in determining your system's characteristics, and substitutes for them in `Makefile`'s:

'`CC`'
: The compiler to use: default is `gcc` if it's installed, otherwise `cc`.

'`CFLAGS`'
: Options to give the compiler: default is '`-g -O2`' for `gcc`, '`-g`' otherwise. `CFLAGS` comes after any other options. You may need to include `-w` here if your compilations commonly have useless warnings (e.g., `NULL redefined`), or `configure` may fail to detect the presence of header files (it takes the messages on standard error to mean the header file doesn't exist).

'`CPPFLAGS`'
: Options to pass to the compiler preprocessor; this matters most for configuration, not the actual source compilation. The `configure` script often does only preprocessing (e.g., to check for the existence of `#include` files), and `CFLAGS` is not used for this. You may need to set this to something like '`-I/usr/local/include/wwwhatever`' if you have the libwww library installed for hyper-xdvik (see '`xdvik/INSTALL`').

'`DEFS`'
: Additional preprocessor options, but not used by `configure`. Provided for enabling or disabling program features, as documented in the various program-specific installation instructions. `DEFS` comes before any compiler options included by the distribution '`Makefile`'s or by `configure`.

'`LDFLAGS`'
: Additional options to give to the loader. `LDFLAGS` comes before any other linker options.

'`LIBS`'
: Additional libraries to link with.

2.2.4.4 `configure` scenarios

Here are some common installation scenarios:

- Including X support in Metafont. This is disabled by default, since many sites have no use for it, and it's a leading cause of configuration problems.

  ```
  configure --with-x
  ```

- Putting the binaries, TEX files, GNU info files, etc. into a single TEX hierarchy, say '`/here/texmf`', requires overriding defaults in `configure`:

  ```
  configure --prefix=/here/texmf --datadir=/here
  ```

- You can compile on multiple architectures simultaneously either by building symbolic link trees with the `lndir` script from the X11 distribution, or with the '`--srcdir`' option:

  ```
  configure --srcdir=srcdir
  ```

- If you are installing binaries for multiple architectures into a single hierarchy, you will probably want to override the default '`bin`' and '`lib`' directories, something like this:

  ```
  configure --prefix=texmf --datadir=texmf \
    --bindir=texmf/arch/bin --libdir=texmf/arch/lib
  make texmf=texmf
  ```

  (Unless you make provisions for architecture-specific files in other ways, e.g., with Depot or an automounter.)

- To compile with optimization (to compile without debugging, remove the '`-g`'):

  ```
  env CFLAGS="-g -O" sh configure ...
  ```

  For a potential problem if you optimize, see Section 2.6.5.4 [TEX or Metafont failing], page 26.

2.2.4.5 Shared library

You can compile Kpathsea as a shared library on a few systems, by specifying the option '`--enable-shared`' when you run '`configure`'.

The main advantage in doing this is that the executables can then share the code, thus decreasing memory and disk space requirements.

On some systems, you can record the location of shared libraries in a binary, usually by giving certain options to the linker. Then individual users do not need to set their system's environment variable (e.g., `LD_LIBRARY_PATH`) to find shared libraries. If you want to do this, you will need to add the necessary options to `LDFLAGS` yourself; for example, on Solaris, include something like '`-R${prefix}/lib`', on IRIX or Linux, use '`-rpath${prefix}/lib`'. (Unfortunately, making this happen by default is very difficult, because of interactions with an existing installed shared library.)

Currently, shared library support is implemented only on Linux, SunOS 4 (Solaris 1), SunOS 5 (Solaris 2), IRIX 5, and IRIX 6. If you're interested and willing in adding support for other systems, please see the '`configure`' mode in the '`klibtool`' script, especially the host-specific case statement around line 250.

2.2.5 Running `make`

`make` (still in the top-level directory). This also creates the '`texmf.cnf`' and '`paths.h`' files that define the default search paths, and (by default) the '`plain`' and '`latex`' TeX formats.

You can override directory names and other values at `make`-time. '`make/paths.make`' lists the variables most commonly reset. For example, '`make default_texsizes=600`' changes the list of fallback resolutions.

You can also override each of `configure`'s environment variables (see Section 2.2.4.3 [configure environment], page 8). The Make variables have the same names.

Finally, you can supply additional options via the following variables. (`configure` does not use these.)

'`XCPPFLAGS`'
'`XDEFS`' Preprocessor options.

'`XCFLAGS`' Compiler options.

'`XLDFLAGS`'
: Loader options (included at beginning of link commands).

'`XLOADLIBES`'
: More loader options (included at end of link commands).

'`XMAKEARGS`'
: Additional Make arguments passed to all sub-`make`'s. You may need to include assignments to the other variables here via `XMAKEARGS`; for example: '`make XMAKEARGS="CFLAGS=-O XDEFS=-DA4"`'.

It's generally a bad idea to use a different compiler ('`CC`') or libraries (`LIBS`) for compilation than you did for configuration, since the values `configure` determined may then be incorrect.

Adding compiler options to change the "universe" you are using (typically BSD vs. system V) is generally a cause of trouble. It's best to use the native environment, whatever that is; `configure` and the software usually adapt best to that. In particular, under Solaris 2.x, you should not use the BSD-compatibility library ('`libucb`') or include files ('`ucbinclude`').

If you want to use the Babel LaTeX package for support of non-English typesetting, you need to modify some files before making the LaTeX format. See the file '`install.txt`' in the Babel distribution.

2.2.6 Installing files

The basic command is the usual `make install`. For security issues, see Section 2.3 [Security], page 15.

The first time you install any manual in the GNU Info system, you should add a line (you choose where) to the file '`dir`' in your '`$(infodir)`' directory. Sample text for this is given near the top of the Texinfo source files ('`kpathsea/kpathsea.texi`', '`dvipsk/dvips.texi`', and '`web2c/doc/web2c.texi`'). If you have a recent version of the GNU Texinfo distribution installed (`ftp://prep.ai.mit.edu/pub/gnu/texinfo-3.9.tar.gz` or later), this should happen automatically.

On the offchance that this is your first Info installation, the '`dir`' file I use is included in the distribution as '`etc/dir-example`'.

You may wish to use one of the following targets, especially if you are installing on multiple architectures:

- `make install-exec` to install in architecture-dependent directories, i.e., ones that depend on the `$(exec_prefix)` Make variable. This includes links to binaries, libraries, etc., not just "executables".
- `make install-data` to install in architecture-independent directories, such as documentation, configuration files, pool files, etc.

If you use the Andrew File System, the normal path (e.g., *prefix*/bin) only gets you to a read-only copy of the files, and you must specify a different path for installation. The best way to do this is by setting the '`prefix`' variable on the `make` command line. The sequence becomes something like this:

```
configure --prefix=/whatever
make
make install prefix=/afs/.system.name/system/1.3/@sys/whatever
```

With AFS, you will definitely want to use relative filenames in '`ls-R`' (see Section 3.4 [Filename database], page 34), not absolute filenames. This is done by default, but check anyway.

2.2.7 Cleaning up

The basic command is `make distclean`. This removes all files created by the build.

Alternatively,

- `make mostlyclean` if you intend to compile on another architecture. For Web2C, since the generated C files are portable, they are not removed. If the `lex` vs. `flex` situation is going to be different on the next machine, `rm web2c/lex.yy.c`.
- `make clean` to remove files created by compiling, but leave configuration files and Makefiles.
- `make maintainer-clean` to remove everything that the Makefiles can rebuild. This is more than '`distclean`' removes, and you should only use it if you are thoroughly conversant with (and have the necessary versions of) Autoconf.
- `make extraclean` to remove other junk, e.g., core files, log files, patch rejects. This is independent of the other '`clean`' targets.

2.2.8 Filename database generation

You will probably want to set up a `cron` entry on the appropriate machine(s) to rebuild the filename database nightly or so, as in:

```
0 0 * * * cd texmf && /bindir/mktexlsr
```

See Section 3.4 [Filename database], page 34.

Although the `mktex...` scripts make every effort to add newly-created files on the fly, it can't hurt to make sure you get a fresh version every so often.

2.2.9 'mktex' scripts

If Kpathsea cannot otherwise find a file, for some file types it is configured by default to invoke an external program to create it dynamically (see Section 2.2.9.1 [mktex configuration], page 12). This is most useful for fonts (bitmaps, TFM's, and arbitrarily-sizable Metafont sources such as the Sauter and EC fonts), since any given document can use fonts never before referenced. Trying to build all fonts in advance is therefore impractical, if not impossible.

The script is passed the name of the file to create and possibly other arguments, as explained below. It must echo the full pathname of the file it created (and nothing else) to standard output; it can write diagnostics to standard error.

2.2.9.1 'mktex' configuration

The following file types can run an external program to create missing files: 'pk', 'tfm', 'mf', 'tex'; the scripts are named 'mktexpk', 'mktextfm', 'mktexmf', and 'mktextex'.

In the absence of `configure` options specifying otherwise, everything but 'mktextex' will be enabled by default. The `configure` options to change the defaults are:

```
--without-mktexmf-default
--without-mktexpk-default
--without-mktextfm-default
--with-mktextex-default
```

The `configure` setting is overridden if the environment variable or configuration file value named for the script is set; e.g., 'MKTEXPK' (see Section 2.2.9.3 [mktex script arguments], page 14).

As distributed, all the scripts source a file 'texmf/web2c/mktex.cnf' if it exists, so you can override various defaults. See 'mktex.opt', for instance, which defines the default mode, resolution, some special directory names, etc. If you prefer not to change the distributed scripts, you can simply create 'mktex.cnf' with the appropriate definitions (you do not need to create it if you have nothing to put in it). 'mktex.cnf' has no special syntax; it's an arbitrary Bourne shell script. The distribution contains a sample 'mktex.cnf' for you to copy and modify as you please (it is not installed anywhere).

In addition, you can configure a number of features with the `MT_FEATURES` variable, which you can define:

- in 'mktex.opt', as just mentioned;
- by editing the file 'mktex.opt', either before 'make install' (in the source hierarchy) or after (in the installed hierarchy);
- or in the environment.

If none of the options below are enabled, `mktexpk`, `mktextfm`, and `mktexmf` follow the following procedure to decide where fonts should be installed. Find the tree where the font's sources are, and test the permissions of the 'fonts' directory of that tree to determine whether it is writable. If it is, put the files in the tree in appropriate locations. If it isn't writable, see whether the tree is a system tree (named in `SYSTEXMF`). If so, the `VARTEXFONTS` tree is used. In all other cases the working directory is used.

The 'appendonlydir' option is enabled by default.

'appendonlydir'
: Tell mktexdir to create directories append-only, i.e., set their sticky bit (see section "Mode Structure" in *GNU File Utilities*). This feature is silently ignored on non-Unix platforms (e.g. Windows/NT and MS-DOS) which don't support similar functionality. This feature is enabled by default.

'dosnames'
: Use 8.3 names; e.g., 'dpi600/cmr10.pk' instead of 'cmr10.600pk'. Note that this feature only affects filenames that would otherwise clash with other TeX-related filenames; 'mktex' scripts do nothing about filenames which exceed the 8+3 MS-DOS limits but remain unique when truncated (by the OS) to these limits, and nether do the scripts care about possible clashes with files which aren't related with TeX. For example, 'cmr10.600pk' would clash with 'cmr10.600gf' and is therefore changed when 'dosnames' is in effect, but 'mf.pool' and 'mp.base' don't clash with any TeX-related files and are therefore unchanged.

  This feature is turned on by default on MS-DOS. If you do not wish 'dosnames' to be set on an MS-DOS platform, you need to set the MT_FEATURES environment variable to a value that doesn't include 'dosnames'. You can also change the default setting by editing 'mktex.opt', but only if you use the 'mktex' shell scripts; the emulation programs don't consult 'mktex.opt'.

'fontmaps'
: Instead of deriving the location of a font in the destination tree from the location of the sources, the aliases and directory names from the Fontname distribution are used. (see section "Introduction" in *Fontname*).

'nomfdrivers'
: Let mktexpk and mktextfm create metafont driver files in a temporary directory. These will be used for just one metafont run and not installed permanently.

'nomode'
: Omit the directory level for the mode name; this is fine as long as you generate fonts for only one mode.

'stripsupplier'
: Omit the font supplier name directory level.

'striptypeface'
: Omit the font typeface name directory level.

'strip'
: Omit the font supplier and typeface name directory levels. This feature is deprecated in favour of 'stripsupplier' and 'striptypeface'.

'varfonts'
: When this option is enabled, fonts that would otherwise be written in system texmf tree go to the VARTEXFONTS tree instead. The default value in 'kpathsea/Makefile.in' is '/var/tmp/texfonts'. The *Linux File System Standard* recommends '/var/tex/fonts'.

  The 'varfonts' setting in MT_FEATURES is overridden by the USE_VARTEXFONTS environment variable: if set to '1', the feature is enabled, and if set to '0', the feature is disabled.

2.2.9.2 'mktex' script names

The following table shows the default name of the script for each possible file types. (The source is the variable kpse_make_specs in 'kpathsea/tex-make.c'.)

'mktexpk' Glyph fonts.

'mktextex'
TeX input files.

'mktexmf' Metafont input files.

'mktextfm'
TFM files.

These names are overridden by an environment variable specific to the program—for example, DVIPSMAKEPK for Dvipsk.

If a mktex... script fails, the invocation is appended to a file 'missfont.log' (by default) in the current directory. You can then execute the log file to create the missing files after fixing the problem.

If the current directory is not writable and the environment variable or configuration file value TEXMFOUTPUT is set, its value is used. Otherwise, nothing is written. The name 'missfont.log' is overridden by the MISSFONT_LOG environment variable or configuration file value.

2.2.9.3 'mktex' script arguments

The first argument to a 'mktex' script is always the name of the file to be created.

In the default 'mktexpk' implementation, additional arguments may also be passed:

'--dpi *num*'
Sets the resolution of the generated font to *num*.

'--mfmode *name*'
Sets the Metafont mode to *name*.

'--bdpi *num*'
Sets the the "base dpi" for the font. This must match the mode being used.

'--mag *string*'
A "magstep" string suitable for the Metafont mag variable. This must match the combination of *bdpi* and *dpi* being used.

'--destdir *string*'
A directory name. If the directory is absolute, it is used as-is. Otherwise, it is appended to the root destination directory set in the script.

2.2.10 Installation testing

Besides the tests listed in Section 2.1 [Simple installation], page 3, you can try running 'make check'. This includes the torture tests (trip, trap, and mptrap) that come with Web2c (see section "Triptrap" in *Web2c*).

2.3 Security

None of the programs in the TEX system require any special system privileges, so there's no first-level security concern of people gaining illegitimate root access.

A TEX document, however, can write to arbitrary files, e.g., '`~/.rhosts`', and thus an unwitting user who runs TEX on a random document is vulnerable to a trojan horse attack. This loophole is closed by default, but you can be permissive if you so desire in '`texmf.cnf`'. See section "tex invocation" in *Web2c*. MetaPost has the same issue.

Dvips, Xdvi, and TEX can also execute shell commands under some circumstances. To disable this, see the '`-R`' option in section "Option details" in *Dvips*, the xdvi man page, and section "tex invocation" in *Web2c*, respectively.

Another security issue arises because it's very useful—almost necessary—to make arbitrary fonts on user demand with `mktexpk` and friends. Where do these files get installed? By default, the `mktexpk` distributed with Kpathsea assumes a world-writable '`/var/tmp`' directory; this is a simple and convenient approach, but it may not suit your situation because it means that a local cache of fonts is created on every machine.

To avoid this duplication, many people consider a shared, globally writable font tree desirable, in spite of the potential security problems. To do this you should change the value of `VARTEXFONTS` in '`texmf.cnf`' to refer to some globally known directory. See Section 2.2.9.1 [mktex configuration], page 12.

The first restriction you can apply is to make newly-created directories under '`texmf`' be append-only with an option in '`mktex.cnf`'. See Section 2.2.9.1 [mktex configuration], page 12.

Another approach is to establish a group (or user) for TEX files, make the '`texmf`' tree writable only to that group (or user), and make `mktexpk` et al. setgid to that group (or setuid to that user). Then users must invoke the scripts to install things. (If you're worried about the inevitable security holes in scripts, then you could write a C wrapper to exec the script.)

The '`mktex...`' scripts install files with the same read and write permissions as the directory they are installed in. The executable, sgid, suid, and sticky bits are always cleared.

Any directories created by the '`mktex...`' scripts have the same permissions as their parent directory, unless the `appendonlydir` feature is used, in which case the sticky bit is always set.

2.4 TEX directory structure

This section describes the default installation hierarchy of the distribution. It conforms to both the GNU coding standards and the TEX directory structure (TDS) standard. For rationale and further explanation, please see those documents. The GNU standard is available as `ftp://prep.ai.mit.edu/pub/gnu/standards/standards.texi` and mirrors. The TDS document is available from '*CTAN:*`/tex-archive/tds`' (see Section 2.5 [unixtex.ftp], page 17).

You can change the default paths in many ways (see Section 2.2.3 [Changing search paths], page 5). One common desire is to put everything (binaries and all) under a single top-level directory such as '/usr/local/texmf' or '/opt/texmf'—in the terms used below, make *prefix* and *texmf* the same. For specific instructions on doing that, see Section 2.2.4.4 [configure scenarios], page 9.

Here is a skeleton of the default directory structure, extracted from the TDS document:

prefix/	installation root ('/usr/local' by default)
bin/	executables
man/	man pages
include/	C header files
info/	GNU info files
lib/	libraries ('libkpathsea.*')
share/	architecture-independent files
texmf/	TDS root
bibtex/	BibTEX input files
bib/	BibTEX databases
base/	base distribution (e.g., 'xampl.bib')
misc/	single-file databases
pkg/	name of a package
bst/	BibTEX style files
base/	base distribution (e.g., 'plain.bst', 'acm.bst')
misc/	single-file styles
pkg/	name of a package
doc/	additional documentation
dvips/	'.pro', '.ps', 'psfonts.map'
fonts/	font-related files
type/	file type (e.g., 'tfm', 'pk')
mode/	type of output device (types 'pk' and 'gf' only)
supplier/	name of a font supplier (e.g., 'public')
typeface/	name of a typeface (e.g., 'cm')
dpi*nnn*/	font resolution (types 'pk' and 'gf' only)
metafont/	Metafont (non-font) input files
base/	base distribution (e.g., 'plain.mf')
misc/	single-file packages (e.g., 'modes.mf')
pkg/	name of a package (e.g., 'mfpic')
metapost/	MetaPost input files
base/	base distribution (e.g., 'plain.mp')
misc/	single-file packages
pkg/	name of a package
support/	support files for MetaPost-related utilities (e.g., 'trfonts.map')
mft/	'MFT' inputs (e.g., 'plain.mft')
tex/	TEX input files
format/	name of a format (e.g., 'plain')
base/	base distribution for *format* (e.g., 'plain.tex')
misc/	single-file packages (e.g., 'webmac.tex')
local/	local additions to or local configuration files for *format*
pkg/	name of a package (e.g., 'graphics', 'mfnfss')
generic/	format-independent packages
hyphen/	hyphenation patterns (e.g., 'hyphen.tex')
images/	image input files (e.g., Encapsulated PostScript)
misc/	single-file format-independent packages (e.g., 'null.tex').

pkg/	name of a package (e.g., '`babel`')
`web2c/`	implementation-dependent files ('`.pool`', '`.fmt`', '`texmf.cnf`', etc.)

Some concrete examples for most file types:

```
/usr/local/bin/tex
/usr/local/man/man1/xdvi.1
/usr/local/info/kpathsea.info
/usr/local/lib/libkpathsea.a
/usr/local/share/texmf/bibtex/bst/base/plain.bst
/usr/local/share/texmf/fonts/pk/ljfour/public/cm/cmr10.600pk
/usr/local/share/texmf/fonts/source/public/pandora/pnr10.mf
/usr/local/share/texmf/fonts/tfm/public/cm/cmr10.tfm
/usr/local/share/texmf/fonts/type1/adobe/utopia/putr.pfa
/usr/local/share/texmf/metafont/base/plain.mf
/usr/local/share/texmf/metapost/base/plain.mp
/usr/local/share/texmf/tex/plain/base/plain.tex
/usr/local/share/texmf/tex/generic/hyphen/hyphen.tex
/usr/local/share/texmf/web2c/tex.pool
/usr/local/share/texmf/web2c/tex.fmt
/usr/local/share/texmf/web2c/texmf.cnf
```

2.5 '`unixtex.ftp`': Obtaining TEX

This chapter is `ftp://ftp.tug.org/tex/unixtex.ftp`, last updated 18 March 1998. Also available as `http://www.tug.org/unixtex.ftp`. The IP address is currently `[158.121.106.10]`, and the canonical host name is currently '`tug.org`'. It is also in Kpathsea source distributions as '`etc/unixtex.ftp`' (although the network version is usually newer). Mail `kb@mail.tug.org` with comments or questions.

Following are general instructions for Unix or other sites who wish to acquire the Web2c distribution, (plain) TEX, LaTEX (2e), BibTEX, Metafont, MetaPost, DVI processors for the X window system, PostScript, the PCL language in the HP LaserJet, and related programs. They are oriented towards building from the original sources, though some information on alternative packages is included in the last section. See also `http://www.tug.org/web2c`, the Web2c and Kpathsea home page.

Please consider joining the TEX Users Group (TUG) to help support the maintenance and development of the programs you retrieve. Email `office@tug.org` or see `http://www.tug.org` for information and a membership form.

For actual installation instructions after obtaining the necessary sources, see Chapter 2 [Installation], page 3. A copy is in the distribution file '`kpathsea/INSTALL`'.

2.5.1 Electronic distribution

In many places we refer to *CTAN:*. This is *both* a host name and a directory name. Here are some primary locations:

`ftp://ctan.tug.org/tex-archive`	(California, USA)
`ftp://ftp.dante.de/tex-archive`	(Germany)
`ftp://ftp.tex.ac.uk/tex-archive`	(England)

CTAN has many mirrors worldwide; see the top-level file '`README.mirrors`' from one of the sites above, or finger `ctan@ftp.tug.org`, or see `http://www.tug.org/CTAN.sites`. A list current as of the time of distribution is in the top-level file '`./MIRROR`'.

You can also access CTAN via the World Wide Web, Gopher, electronic mail, or NFS. The same '`README.mirrors`' file explains how.

You will need to retrieve some or all of the following archives, depending on your needs (don't forget to set binary mode for file transfers):

'*CTAN:*`/systems/web2c/texmflib.tar.gz`'
: A basic collection of fonts (TFM files only) and macro packages (including Texinfo and LaTeX 2e). It unpacks into '`texmf/`'; if you change the structure of this hierarchy, you will also have to change the default search paths (see Section 2.2.3 [Changing search paths], page 5). It is required unless you already have these files, in which case you should change the default paths as necessary to find them.

'*CTAN:*`/systems/web2c/web.tar.gz`'
: The original WEB source files, written mostly by Don Knuth. Required unless you already have this '`web`' version. (The WEB sources change irregularly with respect to Web2c itself.) Unpacks into '`web2c-`*version*'.

'*CTAN:*`/systems/web2c/web2c.tar.gz`'
: The Web2c system. Required. Also unpacks into '`web2c-`*version*'.

'*CTAN:*`/systems/web2c/web2c-etex.tar.gz`'
: Additions to the Web2c system for building e-TeX. Optional. Unpacks into '`web2c-`*version*'.

'*CTAN:*`/systems/web2c/etexlib.tar.gz`'
: Additions to the texmf tree needed to build e-TeX. Optional. Unpacks into '`texmf/`'.

'*CTAN:*`/systems/web2c/etexdoc.tar.gz`'
: Documentation for e-TeX as an addition to the texmf tree. Optional. Unpacks into '`texmf/`'.

'*CTAN:*`/systems/web2c/web2c-omega.tar.gz`'
: Additions to the Web2c system for building Omega. Optional. Unpacks into '`web2c-`*version*'.

'*CTAN:*`/systems/web2c/omegalib.tar.gz`'
: Additions to the texmf tree needed to build Omega. Optional. Unpacks into '`texmf/`'.

'*CTAN:*`/systems/web2c/omegadoc.tar.gz`'
: Documentation for Omega as an addition to the texmf tree. Optional. Unpacks into '`texmf/`'.

'*CTAN:*`/systems/web2c/web2c-pdftex.tar.gz`'
: Additions to the Web2c system for building pdfTeX. Optional. Unpacks into '`web2c-`*version*'.

'*CTAN*:/systems/web2c/pdftexlib.tar.gz'
: Additions to the texmf tree needed to build pdfTEX. Optional. Unpacks into 'texmf/'.

'*CTAN*:/systems/web2c/pdftexdoc.tar.gz'
: Unpacks into 'texmf/'. Documentation for pdfTEX as an addition to the texmf tree. Optional. Unpacks into 'texmf/'.

'*CTAN*:/systems/web2c/texk.tar.gz'
: The web and web2c sources, plus the additions for web2c, plus the dvi drivers mentioned below, plus extra dvi drivers and tools not packaged separately. Recommended if you want to build a complete system. Unpacks into 'texk-*version*'.

'*CTAN*:/systems/web2c/texklib.tar.gz'
: The basic texmf tree, plus the additions for e-TEX, Omega, and pdfTEX. The match for texk.tar.gz. Unpacks into 'texmf/'.

'*CTAN*:/dviware/dvipsk/dvipsk.tar.gz'
: DVI-to-PostScript translator. Unpacks into 'dvipsk-*version*'. Optional.

'*CTAN*:/dviware/xdvik/xdvik.tar.gz'
: X window system DVI previewer. Unpacks into 'xdvik-*version*'. Optional.

'*CTAN*:/dviware/dviljk/dviljk.tar.gz'
: DVI-to-PCL (HP LaserJet) translator. Unpacks into 'dviljk-*version*'. Optional.

All that said, the originating host for the software above is 'ftp.tug.org'. You can retrieve these distributions (but not much else) from the 'tex/' directory on that host.

2.5.2 CD-ROM distribution

Numerous organizations distribute various TEX CD-ROM's:

- TUG, UK TUG, and GUTenberg (French-speaking TEX user group) collaborated to produce the *TEX Live* CD-ROM, based on teTEX, which in turn is based on Web2c; email tex-live@mail.tug.org or see http://www.tug.org/tex-live.html.
- Dante (the German-speaking TEX user group) has produced a CD-ROM. See http://www.dante.de/dante/DANTE-CTAN-CD-ROM.html, and http://www.dante.de/tex-informationen/CD-ROMs.html for information about TEX CD's in general. Both are in German.
- The Free Software Foundation's *Source Code CD-ROM* contains the minimal TEX source distribution described in the previous section (i.e., enough to print GNU documentation); email gnu@prep.ai.mit.edu.
- The Gateway! CD-ROM set contains a runnable NetBSD/Amiga distribution and sources; see http://www.netbsd.org/Sites/cdroms.html.
- The InfoMagic CD-ROM is a copy of CTAN (see previous section); see http://www.infomagic.com.
- NTG (Dutch-speaking TEX user group) produced the 4allTEX CD-ROM; email ntg@nic.surfnet.nl, or see http://www.ntg.nl/4allcd/. This is a runnable system.

- The Prime Time TEXcetera CD-ROM is also a copy of CTAN; email `ptf@ctcl.com` or see `http://www.ptf.com/ptf/`.
- Walnut Creek's TEX CD-ROM is also a copy of CTAN; email `info@cdrom.com` or see `http://www.cdrom.com:/titles/tex.html`.
- Most Linux distributions include some TEX package based on Web2c; see the Linux documentation file '`Distribution-HOWTO`' for a comparison of Linux distributions, available (for example) via `http://www.linux.org`.

If you know of additional TEX CD-ROM distributions to add to this list, please inform `kb@mail.tug.org`.

2.5.3 Other TEX packages

Many other TEX implementations are available in '*CTAN:*`/systems`', including ready-to-run distributions for Unix, Amiga, Acorn, VMS, Macintosh, DOS, and Windows (in various forms). Although Web2c has support in the source code for many operating systems, and in fact some of the other distributions are based on it, it's unlikely to work as distributed on anything but Unix. (Please contribute improvements!)

The Unix distribution alluded to above is the teTEX distribution. This includes both complete sources and precompiled binaries for many popular Unix variants, including Linux. It is based on Web2c, and contains many other TEX-related programs as well.

The host `labrea.stanford.edu` is the original source for the files for which Donald Knuth is directly responsible: '`tex.web`', '`plain.tex`', etc. However, unless you want to build your TEX library tree ab initio, it is more reliable and less work to retrieve these files as part of the above packages. In any case, `labrea` is not the canonical source for anything except what was created by Stanford TEX project, so do not rely on all the files available at that ftp site being up-to-date.

2.6 Reporting bugs

(A copy of this chapter is in the file '`kpathsea/BUGS`'.)

If you have problems or suggestions, please report them to `tex-k@mail.tug.org` using the bug checklist below.

Please report bugs in the documentation; not only factual errors or inconsistent behavior, but unclear or incomplete explanations, typos, wrong fonts, . . .

2.6.1 Bug checklist

Before reporting a bug, please check below to be sure it isn't already known (see Section 2.6.5 [Common problems], page 24).

Bug reports should be sent via electronic mail to `tex-k@mail.tug.org`, or by postal mail to 135 Center Hill Road / Plymouth, MA 02360 / USA.

The general principle is that a good bug report includes all the information necessary for reproduction. Therefore, to enable investigation, your report should include the following:

- The version number(s) of the program(s) involved, and of Kpathsea itself. You can get the former by giving a sole option '`--version`' to the program, and the latter by running '`kpsewhich --version`'. The '`NEWS`' and '`ChangeLog`' files also contain the version number.
- The hardware, operating system (including version number), compiler, and `make` program you are using (the output of `uname -a` is a start on the first two, though often incomplete). If the bug involves the X window system, include X version and supplier information as well (examples: X11R6 from MIT; X11R4 from HP; OpenWindows 3.3 bundled with SunOS 4.1.4).
- Any options you gave to `configure`. This is recorded in the '`config.status`' files.

  If you are reporting a bug in '`configure`' itself, it's probably system-dependent, and it will be unlikely the maintainers can do anything useful if you merely report that thus-and-such is broken. Therefore, you need to do some additional work: for some bugs, you can look in the file '`config.log`' where the test that failed should appear, along with the compiler invocation and source program in question. You can then compile it yourself by hand, and discover why the test failed. Other '`configure`' bugs do not involve the compiler; in that case, the only recourse is to inspect the `configure` shell script itself, or the Autoconf macros that generated `configure`.
- The log of all debugging output, if the bug is in path searching. You can get this by setting the environment variable `KPATHSEA_DEBUG` to '`-1`' before running the program. Please look at the log yourself to make sure the behavior is really a bug before reporting it; perhaps "old" environment variable settings are causing files not to be found, for example.
- The contents of any input files necessary to reproduce the bug. For bugs in DVI-reading programs, for example, this generally means a DVI file (and any EPS or other files it uses)—TeX source files are helpful, but the DVI file is necessary, because that's the actual program input.

  GNU `shar`, available from `ftp://prep.ai.mit.edu/pub/gnu` is a convenient way of packaging multiple (possibly binary) files for electronic mail. If you feel your input files are too big to send by email, you can ftp them to `ftp://ftp.tug.org/incoming` (that directory is writable, but not readable).
- If you are sending a patch (do so if you can!), please do so in the form of a context diff ('`diff -c`') against the original distribution source. Any other form of diff is either not as complete or harder for me to understand. Please also include a '`ChangeLog`' entry.
- If the bug involved is an actual crash (i.e., core dump), it is easy and useful to include a stack trace from a debugger (I recommend the GNU debugger GDB, available from `ftp://prep.ai.mit.edu/pub/gnu`). If the cause is apparent (a `NULL` value being dereferenced, for example), please send the details along. If the program involved is TeX or Metafont, and the crash is happening at apparently-sound code, however, the bug may well be in the compiler, rather than in the program or the library (see Section 2.6.5.4 [TeX or Metafont failing], page 26).
- Any additional information that will be helpful in reproducing, diagnosing, or fixing the bug.

2.6.2 Mailing lists

Web2c and Kpathsea in general are discussed on the mailing list `tex-k@mail.tug.org`. To join, email `tex-k-request@mail.tug.org` with a line consisting of

`subscribe` *you@your.preferred.email.address*

in the body of the message.

You do not need to join to submit a report, nor will it affect whether you get a response. There is no Usenet newsgroup equivalent (if you can be the one to set this up, email '`tex-k-request`'). Traffic on the list is fairly light, and is mainly bug reports and enhancement requests to the software. The best way to decide if you want to join or not is read some of the archives from `ftp://ftp.tug.org/mail/archives/tex-k/`.

Be aware that large data files are sometimes included in bug reports. If this is a problem for you, do not join the list.

If you only want announcements of new releases, not bug reports and discussion, join `tex-archive@math.utah.edu` (via mail to `tex-archive-request@math.utah.edu`).

If you are looking for general TEX help, such as how to use LaTEX, please use the mailing list `info-tex@shsu.edu` mailing list, which is gatewayed to the '`comp.text.tex`' Usenet newsgroup (or post to the newsgroup; the gateway is bidirectional).

2.6.3 Debugging

Kpathsea provides a number of runtime debugging options, detailed below by their names and corresponding numeric values. When the files you expect aren't being found, the thing to do is enable these options and examine the output.

You can set these with some runtime argument (e.g., '`-d`') to the program; in that case, you should use the numeric values described in the program's documentation (which, for Dvipsk and Xdvik, are different than those below). It's best to give the '`-d`' (or whatever) option first, for maximal output. Dvipsk and Xdvik have additional program-specific debugging options as well.

You can also set the environment variable `KPATHSEA_DEBUG`; in this case, you should use the numbers below. If you run the program under a debugger and set the variable `kpathsea_debug`, also use the numbers below.

In any case, by far the simplest value to use is '`-1`', which will turn on all debugging output. This is usually better than guessing which particular values will yield the output you need.

Debugging output always goes to standard error, so you can redirect it easily. For example, in Bourne-compatible shells:

```
dvips -d -1 ... 2>/tmp/debug
```

It is sometimes helpful to run the standalone Kpsewhich utility (see Section 3.5 [Invoking kpsewhich], page 36), instead of the original program.

In any case, you can *not* use the *names* below; you must always use somebody's numbers. (Sorry.) To set more than one option, just sum the corresponding numbers.

KPSE_DEBUG_STAT (1)
: Report 'stat'(2) calls. This is useful for verifying that your directory structure is not forcing Kpathsea to do many additional file tests (see Section 2.6.5.2 [Slow path searching], page 25, and see Section 3.3.6 [Subdirectory expansion], page 33). If you are using an up-to-date 'ls-R' database (see Section 3.4 [Filename database], page 34), this should produce no output unless a nonexistent file that must exist is searched for.

KPSE_DEBUG_HASH (2)
: Report lookups in all hash tables: 'ls-R' and 'aliases' (see Section 3.4 [Filename database], page 34); font aliases (see Section 4.3.2 [Fontmap], page 44); and config file values (see Section 3.2.1 [Config files], page 30). Useful when expected values are not being found, e.g.., file searches are looking at the disk instead of using 'ls-R'.

KPSE_DEBUG_FOPEN (4)
: Report file openings and closings. Especially useful when your system's file table is full, for seeing which files have been opened but never closed. In case you want to set breakpoints in a debugger: this works by redefining 'fopen' ('fclose') to be 'kpse_fopen_trace' ('kpse_fclose_trace').

KPSE_DEBUG_PATHS (8)
: Report general path information for each file type Kpathsea is asked to search. This is useful when you are trying to track down how a particular path got defined—from 'texmf.cnf', 'config.ps', an environment variable, the compile-time default, etc. This is the contents of the kpse_format_info_type structure defined in 'tex-file.h'.

KPSE_DEBUG_EXPAND (16)
: Report the directory list corresponding to each path element Kpathsea searches. This is only relevant when Kpathsea searches the disk, since 'ls-R' searches don't look through directory lists in this way.

KPSE_DEBUG_SEARCH (32)
: Report on each file search: the name of the file searched for, the path searched in, whether or not the file must exist (when drivers search for 'cmr10.vf', it need not exist), and whether or not we are collecting all occurrences of the file in the path (as with, e.g., 'texmf.cnf' and 'texfonts.map'), or just the first (as with most lookups). This can help you correlate what Kpathsea is doing with what is in your input file.

KPSE_DEBUG_VARS (64)
: Report the value of each variable Kpathsea looks up. This is useful for verifying that variables do indeed obtain their correct values.

GSFTOPK_DEBUG (128)
: Activates debugging printout specific to gsftopk program.

MAKETEX_DEBUG (512)
: If you use the optional mktex programs instead of the traditional shell scripts, this will report the name of the site file ('mktex.cnf' by default) which is

read, directories created by mktexdir, the full path of the 'ls-R' database built by mktexlsr, font map searches, MT_FEATURES in effect, parameters from mktexnam, filenames added by mktexupd, and some subsidiary commands run by the programs.

MAKETEX_FINE_DEBUG (1024)
: When the optional mktex programs are used, this will print additional debugging info from functions internal to these programs.

Debugging output from Kpathsea is always written to standard error, and begins with the string 'kdebug:'. (Except for hash table buckets, which just start with the number, but you can only get that output running under a debugger. See comments at the hash_summary_only variable in 'kpathsea/db.c'.)

2.6.4 Logging

Kpathsea can record the time and filename found for each successful search. This may be useful in finding good candidates for deletion when your filesystem is full, or in discovering usage patterns at your site.

To do this, define the environment or config file variable TEXMFLOG. The value is the name of the file to append the information to. The file is created if it doesn't exist, and appended to if it does.

Each successful search turns into one line in the log file: two words separated by a space. The first word is the time of the search, as the integer number of seconds since "the epoch", i.e., UTC midnight 1 January 1970 (more precisely, the result of the time system call). The second word is the filename.

For example, after setenv TEXMFLOG /tmp/log, running Dvips on 'story.dvi' appends the following lines:

```
774455887 /usr/local/share/texmf/dvips/config.ps
774455887 /usr/local/share/texmf/dvips/psfonts.map
774455888 /usr/local/share/texmf/dvips/texc.pro
774455888 /usr/local/share/texmf/fonts/pk/ljfour/public/cm/cmbx10.600pk
774455889 /usr/local/share/texmf/fonts/pk/ljfour/public/cm/cmsl10.600pk
774455889 /usr/local/share/texmf/fonts/pk/ljfour/public/cm/cmr10.600pk
774455889 /usr/local/share/texmf/dvips/texc.pro
```

Only filenames that are absolute are recorded, to preserve some semblance of privacy.

2.6.5 Common problems

Here are some common problems with configuration, compilation, linking, execution, ...

2.6.5.1 Unable to find files

If a program complains it cannot find fonts (or other input files), any of several things might be wrong. In any case, you may find the debugging options helpful. See Section 2.6.3 [Debugging], page 22.

- Perhaps you simply haven't installed all the necessary files; the basic fonts and input files are distributed separately from the programs. See Section 2.5 [unixtex.ftp], page 17.
- You have (perhaps unknowingly) told Kpathsea to use search paths that don't reflect where the files actually are. One common cause is having environment variables set from a previous installation, thus overriding what you carefully set in '`texmf.cnf`' (see Section 4.1 [Supported file formats], page 40). System '`/etc/profile`' or other files such may be the culprit.
- Your files reside in a directory that is only pointed to via a symbolic link, in a leaf directory and is not listed in '`ls-R`'.

  Unfortunately, Kpathsea's subdirectory searching has an irremediable deficiency: If a directory *d* being searched for subdirectories contains plain files and symbolic links to other directories, but no true subdirectories, *d* will be considered a leaf directory, i.e., the symbolic links will not be followed. See Section 3.3.6 [Subdirectory expansion], page 33.

  You can work around this problem by creating an empty dummy subdirectory in *d*. Then *d* will no longer be a leaf, and the symlinks will be followed.

  The directory immediately followed by the '`//`' in the path specification, however, is always searched for subdirectories, even if it is a leaf. Presumably you would not have asked for the directory to be searched for subdirectories if you didn't want it to be.
- If the fonts (or whatever) don't already exist, `mktexpk` (or `mktexmf` or `mktextfm`) will try to create them. If these rather complicated shell scripts fail, you'll eventually get an error message saying something like '`Can't find font` *fontname*'. The best solution is to fix (or at least report) the bug in `mktexpk`; the workaround is to generate the necessary fonts by hand with Metafont, or to grab them from a CTAN site (see Section 2.5 [unixtex.ftp], page 17).
- There is a bug in the library. See Section 2.6 [Reporting bugs], page 20.

2.6.5.2 Slow path searching

If your program takes an excessively long time to find fonts or other input files, but does eventually succeed, here are some possible culprits:

- Most likely, you just have a lot of directories to search, and that takes a noticeable time. The solution is to create and maintain a separate '`ls-R`' file that lists all the files in your main TeX hierarchy. See Section 3.4 [Filename database], page 34. Kpathsea always uses '`ls-R`' if it's present; there's no need to recompile or reconfigure any of the programs.
- Your recursively-searched directories (e.g., '`/usr/local/share/texmf/fonts//`'), contain a mixture of files and directories. This prevents Kpathsea from using a useful optimization (see Section 3.3.6 [Subdirectory expansion], page 33).

  It is best to have only directories (and perhaps a '`README`') in the upper levels of the directory structure, and it's very important to have *only* files, and no subdirectories, in the leaf directories where the dozens of TFM, PK, or whatever files reside.

In any case, you may find the debugging options helpful in determining precisely when the disk or network is being pounded. See Section 2.6.3 [Debugging], page 22.

2.6.5.3 Unable to generate fonts

This can happen if either `mktexpk` hasn't been installed properly, or if the local installation of Metafont isn't correct.

If `mf` is a command not found by `mktexpk`, then you need to install Metafont (see Section 2.5 [unixtex.ftp], page 17).

If Metafont runs, but generates fonts at the wrong resolution, you need to be sure the '`M`' and '`D`' lines in your Dvips configuration file match (see section "Config files" in *Dvips*). For example, if `mktexpk` is generating 300 dpi fonts, but you need 600 dpi fonts, you should have:

```
M ljfour
D 600
```

If Metafont runs but generates fonts at a resolution of 2602 dpi (and prints out the name of each character as well as just a character number, and maybe tries to display the characters), then your Metafont base file probably hasn't been made properly. (It's using the default `proof` mode, instead of an actual device mode.) To make a proper '`plain.base`', assuming the local mode definitions are contained in a file '`modes.mf`', run the following command (assuming Unix):

```
inimf "plain; input modes; dump"
```

Then copy the '`plain.base`' file from the current directory to where the base files are stored on your system ('`/usr/local/share/texmf/web2c`' by default), and make a link (either hard or soft) from '`plain.base`' to '`mf.base`' in that directory. See section "inimf invocation" in *Web2c*.

2.6.5.4 TeX or Metafont failing

If TeX or Metafont get a segmentation fault or otherwise fail while running a normal input file, the problem is usually a compiler bug (unlikely as that may sound). Even if the trip and trap tests are passed, problems may lurk. Optimization occasionally causes trouble in programs other than TeX and Metafont themselves, too.

Insufficient swap space may also cause core dumps or other erratic behavior.

For a workaround, if you enabled any optimization flags, it's best to omit optimization entirely. In any case, the way to find the facts is to run the program under the debugger and see where it's failing.

Also, if you have trouble with a system C compiler, I advise trying the GNU C compiler. And vice versa, unfortunately; but in that case I also recommend reporting a bug to the GCC mailing list; see section "Bugs" in *Using and Porting GNU CC*.

To report compiler bugs effectively requires perseverance and perspicacity: you must find the miscompiled line, and that usually involves delving backwards in time from the point of error, checking through TeX's (or whatever program's) data structures. Things are not helped by all-too-common bugs in the debugger itself. Good luck.

One known cause of trouble is the way arrays are handled. Some of the Pascal arrays have a lower index other than 0, and the C code will take the pointer to the allocated memory, subtract the lower index, and use the resulting pointer for the array. While this trick often works, ANSI C doesn't guarantee that it will. It it known to fail on HP-UX

10 mchines when the native compiler is used, unless the '+u' compiler switch was specified. Using GCC will work on this platform as well.

2.6.5.5 Empty Makefiles

On some systems (NetBSD, FreeBSD, AIX 4.1, and Mach10), `configure` may fail to properly create the Makefiles. Instead, you get an error which looks something like this:

```
prompt$ ./configure
...
creating Makefile
sed: 1: "\\@^ac_include make/pat ...": \ can not be used as a string delimiter
```

So far as I know, the bug here is in `/bin/sh` on these systems. I don't have access to a machine running any of them, so if someone can find a workaround that avoids the quoting bug, I'd be most grateful. (Search for `ac_include` in the `configure` script to get to the problematic code.)

It should work to run `bash configure`, instead of using `/bin/sh`. You can get Bash from `ftp://prep.ai.mit.edu/pub/gnu` and mirrors.

Another possible cause (reported for NeXT) is a bug in the `sed` command. In that case the error may look like this:

```
Unrecognized command: \@^ac_include make/paths.make@r make/paths.make
```

In this case, installing GNU `sed` should solve the problem. You can get GNU `sed` from the same places as Bash.

2.6.5.6 `XtStrings`

You may find that linking X programs results in an error from the linker that '`XtStrings`' is undefined, something like this:

```
gcc -o virmf ...
.../x11.c:130: undefined reference to 'XtStrings'
```

This generally happens because of a mismatch between the X include files with which you compiled and the X libraries with which you linked; often, the include files are from MIT and the libraries from Sun.

The solution is to use the same X distribution for compilation and linking. Probably '`configure`' was unable to guess the proper directories from your installation. You can use the `configure` options '`--x-includes=`*path*' and '`--x-libraries=`*path*' to explicitly specify them.

2.6.5.7 `dlopen`

(This section adapted from the file '`dlsym.c`' in the X distribution.)

The `Xlib` library uses the standard C function `wcstombs`. Under SunOS 4.1, `wcstombs` uses the '`dlsym`' interface defined in '`libdl.so`'. Unfortunately, the SunOS 4.1 distribution does not include a static '`libdl.a`' library.

As a result, if you try to link an X program statically under SunOS, you may get undefined references to `dlopen`, `dlsym`, and `dlclose`. One workaround is to include these definitions when you link:

```
void *dlopen() { return 0; }
void *dlsym()  { return 0; }
int dlclose()  { return -1; }
```

These are contained in the '`dlsym.c`' file in the MIT X distribution.

2.6.5.8 `ShellWidgetClass`

(This section adapted from the `comp.sys.sun.admin` FAQ.)

If you are linking with Sun's OpenWindows libraries in SunOS 4.1.x, you may get undefined symbols `_get_wmShellWidgetClass` and `_get_applicationShellWidgetClass` when linking. This problem does not arise using the standard MIT X libraries under SunOS.

The cause is bugs in the `Xmu` shared library as shipped from Sun. There are several fixes:

- Install the free MIT distribution from '`ftp.x.org`' and mirrors.
- Get the OpenWindows patches listed below.
- Statically link the `Xmu` library into the executable.
- Avoid using `Xmu` at all. If you are compiling Metafont, see section "Online Metafont graphics" in *Web2c*. If you are compiling Xdvi, see the `-DNOTOOL` option in '`xdvik/INSTALL`'.
- Ignore the errors. The binary runs fine regardless.

Here is the information for getting the two patches:

Patch ID: 100512-02
Bug ID's: 1086793, 1086912, 1074766
Description: 4.1.x OpenWindows 3.0 `libXt` jumbo patch

Patch ID: 100573-03
Bug ID: 1087332
Description: 4.1.x OpenWindows 3.0 undefined symbols when using shared `libXmu`.

The way to statically link with `libXmu` depends on whether you are using a Sun compiler (e.g., `cc`) or `gcc`. If the latter, alter the `x_libs` Make variable to include

```
-static -lXmu -dynamic
```

If you are using the Sun compiler, use '`-Bstatic`' and '`-Bdynamic`'.

2.6.5.9 Pointer combination warnings

When compiling with old C compilers, you may get some warnings about "illegal pointer combinations". These are spurious; just ignore them. I decline to clutter up the source with casts to get rid of them.

3 Path searching

This chapter describes the generic path searching mechanism Kpathsea provides. For information about searching for particular file types (e.g., TeX fonts), see the next chapter.

3.1 Searching overview

A *search path* is a colon-separated list of *path elements*, which are directory names with a few extra frills. A search path can come from (a combination of) many sources; see below. To look up a file '`foo`' along a path '`.:/dir`', Kpathsea checks each element of the path in turn: first '`./foo`', then '`/dir/foo`', returning the first match (or possibly all matches).

The "colon" and "slash" mentioned here aren't necessarily '`:`' and '`/`' on non-Unix systems. Kpathsea tries to adapt to other operating systems' conventions.

To check a particular path element *e*, Kpathsea first sees if a prebuilt database (see Section 3.4 [Filename database], page 34) applies to *e*, i.e., if the database is in a directory that is a prefix of *e*. If so, the path specification is matched against the contents of the database.

If the database does not exist, or does not apply to this path element, or contains no matches, the filesystem is searched (if this was not forbidden by the specification with '`!!`' and if the file being searched for must exist). Kpathsea constructs the list of directories that correspond to this path element, and then checks in each for the file being searched for. (To help speed future lookups of files in the same directory, the directory in which a file is found is floated to the top of the directory list.)

The "file must exist" condition comes into play with VF files and input files read by the TeX '`\openin`' command. These files may not exist (consider '`cmr10.vf`'), and so it would be wrong to search the disk for them. Therefore, if you fail to update '`ls-R`' when you install a new VF file, it will never be found.

Each path element is checked in turn: first the database, then the disk. If a match is found, the search stops and the result is returned. This avoids possibly-expensive processing of path specifications that are never needed on a particular run. (Unless the search explicitly requested all matches.)

Although the simplest and most common path element is a directory name, Kpathsea supports additional features in search paths: layered default values, environment variable names, config file values, users' home directories, and recursive subdirectory searching. Thus, we say that Kpathsea *expands* a path element, meaning transforming all the magic specifications into the basic directory name or names. This process is described in the sections below. It happens in the same order as the sections.

Exception to all of the above: If the filename being searched for is absolute or explicitly relative, i.e., starts with '`/`' or '`./`' or '`../`', Kpathsea simply checks if that file exists.

Ordinarily, if Kpathsea tries to access a file or directory that cannot be read, it gives a warning. This is so you will be alerted to directories or files that accidentally lack read permission (for example, a '`lost+found`'). If you prefer not to see these warnings, include the value '`readable`' in the `TEX_HUSH` environment variable or config file value.

This generic path searching algorithm is implemented in '`kpathsea/pathsearch.c`'. It is employed by a higher-level algorithm when searching for a file of a particular type (see Section 4.2 [File lookup], page 42, and Section 4.3 [Glyph lookup], page 43).

3.2 Path sources

A search path can come from many sources. In the order in which Kpathsea uses them:

1. A user-set environment variable, e.g., `TEXINPUTS`. Environment variables with an underscore and the program name appended override; for example, `TEXINPUTS_latex` overrides `TEXINPUTS` if the program being run is named '`latex`'.
2. A program-specific configuration file, e.g., an '`S /a:/b`' line in Dvips' '`config.ps`' (see section "Config files" in *Dvips*).
3. A line in a Kpathsea configuration file '`texmf.cnf`', e.g., '`TEXINPUTS=/c:/d`' (see below).
4. The compile-time default (specified in '`kpathsea/paths.h`').

You can see each of these values for a given search path by using the debugging options (see Section 2.6.3 [Debugging], page 22).

These sources may be combined via default expansion (see Section 3.3.1 [Default expansion], page 31).

3.2.1 Config files

As mentioned above, Kpathsea reads *runtime configuration files* named '`texmf.cnf`' for search path and other definitions. The search path used to look for these configuration files is named `TEXMFCNF`, and is constructed in the usual way, as described above, except that configuration files cannot be used to define the path, naturally; also, an '`ls-R`' database is not used to search for them.

Kpathsea reads *all* '`texmf.cnf`' files in the search path, not just the first one found; definitions in earlier files override those in later files. Thus, if the search path is '`.:$TEXMF`', values from '`./texmf.cnf`' override those from '`$TEXMF/texmf.cnf`'.

While (or instead of) reading this description, you may find it helpful to look at the distributed '`texmf.cnf`', which uses or at least mentions most features. The format of '`texmf.cnf`' files follows:

- Comments start with '`%`' and continue to the end of the line.
- Blank lines are ignored.
- A '`\`' at the end of a line acts as a continuation character, i.e., the next line is appended. Whitespace at the beginning of continuation lines is not ignored.
- Each remaining line must look like

  variable [`.` *progname*] [`=`] *value*

  where the '`=`' and surrounding whitespace is optional.
- The *variable* name may contain any character other than whitespace, '`=`', or '`.`', but sticking to '`A-Za-z_`' is safest.
- If '`.`*progname*' is present, the definition only applies if the program that is running is named (i.e., the last component of `argv[0]` is) *progname* or '*progname*`.exe`'. This allows different flavors of TeX to have different search paths, for example.
- *value* may contain any characters except '`%`' and '`@`'. (These restrictions are only necessary because of the processing done on '`texmf.cnf`' at build time, so you can

stick those characters in after installation if you have to.) The '$*var.prog*' feature is not available on the right-hand side; instead, you must use an additional variable (see below for example). A ';' in *value* is translated to ':' if running under Unix; this is useful to write a single 'texmf.cnf' which can be used under both Unix and NT. (If you really want ';''s in your filenames, add '-DALLOW_SEMICOLON_IN_FILENAMES' to CFLAGS.)

- All definitions are read before anything is expanded, so you can use variables before they are defined (like Make, unlike most other programs).

Here is a configuration file fragment illustrating most of these points:

```
% TeX input files -- i.e., anything to be found by \input or \openin ...
latex209_inputs = .:$TEXMF/tex/latex209//:$TEXMF/tex//
latex2e_inputs = .:$TEXMF/tex/latex//:$TEXMF/tex//
TEXINPUTS = .:$TEXMF/tex//
TEXINPUTS.latex209 = $latex209_inputs
TEXINPUTS.latex2e = $latex2e_inputs
TEXINPUTS.latex = $latex2e_inputs
```

Although this format has obvious similarities to Bourne shell scripts—change the comment character to #, disallow spaces around the =, and get rid of the .*name* convention, and it could be run through the shell. But there seemed little advantage to doing this, since all the information would have to passed back to Kpathsea and parsed there anyway, since the sh process couldn't affect its parent's environment.

The implementation of all this is in 'kpathsea/cnf.c'.

3.3 Path expansion

Kpathsea recognizes certain special characters and constructions in search paths, similar to that in shells. As a general example: '~$USER/{foo,bar}//baz' expands to all subdirectories under directories 'foo' and 'bar' in $USER's home directory that contain a directory or file 'baz'. These expansions are explained in the sections below.

3.3.1 Default expansion

If the highest-priority search path (see Section 3.2 [Path sources], page 30) contains an *extra colon* (i.e., leading, trailing, or doubled), Kpathsea inserts at that point the next-highest-priority search path that is defined. If that inserted path has an extra colon, the same happens with the next-highest. (An extra colon in the compile-time default value has unpredictable results, so installers beware.)

For example, given an environment variable setting

```
setenv TEXINPUTS /home/karl:
```

and a TEXINPUTS value from 'texmf.cnf' of

```
.:$TEXMF//tex
```

then the final value used for searching will be:

```
/home/karl:.:$TEXMF//tex
```

Since Kpathsea looks for multiple configuration files, it would be natural to expect that (for example) an extra colon in './texmf.cnf' would expand to the path in

'`$TEXMF/texmf.cnf`'. Or, with Dvips' configuration files, that an extra colon in '`config.$PRINTER`' would expand to the path in '`config.ps`'. This doesn't happen. It's not clear this would be desirable in all cases, and trying to devise a way to specify the path to which the extra colon should expand seemed truly baroque.

Technicality: Since it would be useless to insert the default value in more than one place, Kpathsea changes only one extra '`:`' and leaves any others in place (they will eventually be ignored). Kpathsea checks first for a leading '`:`', then a trailing '`:`', then a doubled '`:`'.

You can trace this by debugging "paths" (see Section 2.6.3 [Debugging], page 22). Default expansion is implemented in the source file '`kpathsea/kdefault.c`'.

3.3.2 Variable expansion

'`$foo`' or '`${foo}`' in a path element is replaced by (1) the value of an environment variable '`foo`' (if defined); (2) the value of '`foo`' from '`texmf.cnf`' (if defined); (3) the empty string.

If the character after the '`$`' is alphanumeric or '`_`', the variable name consists of all consecutive such characters. If the character after the '`$`' is a '`{`', the variable name consists of everything up to the next '`}`' (braces may not be nested around variable names). Otherwise, Kpathsea gives a warning and ignores the '`$`' and its following character.

You must quote the `$`'s and braces as necessary for your shell. *Shell* variable values cannot be seen by Kpathsea, i.e., ones defined by `set` in C shells and without `export` in Bourne shells.

For example, given

```
setenv tex /home/texmf
setenv TEXINPUTS .:$tex:${tex}prev
```

the final `TEXINPUTS` path is the three directories:

```
.:/home/texmf:/home/texmfprev
```

The '`.`*progname*' suffix on variables and '`_`*progname*' on environment variable names are not implemented for general variable expansions. These are only recognized when search paths are initialized (see Section 3.2 [Path sources], page 30).

Variable expansion is implemented in the source file '`kpathsea/variable.c`'.

3.3.3 Tilde expansion

A leading '`~`' in a path element is replaced by the value of the environment variable `HOME`, or '`.`' if `HOME` is not set.

A leading '`~`*user*' in a path element is replaced by *user*'s home directory from the system '`passwd`' database.

For example,

```
setenv TEXINPUTS ~/mymacros:
```

will prepend a directory '`mymacros`' in your home directory to the default path.

As a special case, if a home directory ends in '`/`', the trailing slash is dropped, to avoid inadvertently creating a '`//`' construct in the path. For example, if the home directory of the user 'root' is '/', the path element '~root/mymacros' expands to just '/mymacros', not '//mymacros'.

Tilde expansion is implemented in the source file 'kpathsea/tilde.c'.

3.3.4 Brace expansion

'x{*a*,*b*}y' expands to 'x*a*y:x*b*y'. For example:

```
foo/{1,2}/baz
```

expands to 'foo/1/baz:foo/2/baz'. ':' is the path separator on the current system; e.g., on a DOS system, it's ';'.

Braces can be nested; for example, 'x{A,B{1,2}}y' expands to 'xAy:xB1y:xB2y'.

Multiple non-nested braces are expanded from right to left; for example, 'x{A,B}{1,2}y' expands to 'x{A,B}1y:x{A,B}2y', which expands to 'xA1y:xB1y:xA2y:xB2:y'.

This feature can be used to implement multiple TEX hierarchies, by assigning a brace list to $TEXMF, as mentioned in 'texmf.in'.

You can also use the path separator in stead of the comma. The last example could have been written 'x{A:B}{1:2}y'.

Brace expansion is implemented in the source file 'kpathsea/expand.c'. It is a modification of the Bash sources, and is thus covered by the GNU General Public License, rather than the Library General Public License that covers the rest of Kpathsea.

3.3.5 KPSE_DOT expansion

When KPSE_DOT is defined in the environment, it names a directory that should be considered the current directory for the purpose of looking up files in the search paths. This feature is needed by the 'mktex...' scripts Section 2.2.9 [mktex scripts], page 12, because these change the working directory. You should not ever define it yourself.

3.3.6 Subdirectory expansion

Two or more consecutive slashes in a path element following a directory *d* is replaced by all subdirectories of *d*: first those subdirectories directly under *d*, then the subsubdirectories under those, and so on. At each level, the order in which the directories are searched is unspecified. (It's "directory order", and definitely not alphabetical.)

If you specify any filename components after the '//', only subdirectories which match those components are included. For example, '/a//b' would expand into directories '/a/1/b', '/a/2/b', '/a/1/1/b', and so on, but not '/a/b/c' or '/a/1'.

You can include multiple '//' constructs in the path.

'//' at the beginning of a path is ignored; you didn't really want to search every directory on the system, did you?

I should mention one related implementation trick, which I took from GNU find. Matthew Farwell suggested it, and David MacKenzie implemented it.

The trick is that in every real Unix implementation (as opposed to the POSIX specification), a directory which contains no subdirectories will have exactly two links (namely, one for '.' and one for '..'). That is to say, the st_nlink field in the 'stat' structure will be two. Thus, we don't have to stat everything in the bottom-level (leaf) directories—we can just check `st_nlink`, notice it's two, and do no more work.

But if you have a directory that contains a single subdirectory and 500 regular files, `st_nlink` will be 3, and Kpathsea has to stat every one of those 501 entries. Therein lies slowness.

You can disable the trick by undefining `UNIX_ST_LINK` in '`kpathsea/config.h`'. (It is undefined by default except under Unix.)

Unfortunately, in some cases files in leaf directories are `stat`'d: if the path specification is, say, '`$TEXMF/fonts//pk//`', then files in a subdirectory '`.../pk`', even if it is a leaf, are checked. The reason cannot be explained without reference to the implementation, so read '`kpathsea/elt-dirs.c`' (search for '`may descend`') if you are curious. And if you can find a way to *solve* the problem, please let me know.

Subdirectory expansion is implemented in the source file '`kpathsea/elt-dirs.c`'.

3.4 Filename database (`ls-R`)

Kpathsea goes to some lengths to minimize disk accesses for searches (see Section 3.3.6 [Subdirectory expansion], page 33). Nevertheless, at installations with enough directories, searching each possible directory for a given file can take an excessively long time (depending on the speed of the disk, whether it's NFS-mounted, how patient you are, etc.).

In practice, a font tree containing the standard PostScript and PCL fonts is large enough for searching to be noticeably slow on typical systems these days. Therefore, Kpathsea can use an externally-built "database" file named '`ls-R`' that maps files to directories, thus avoiding the need to exhaustively search the disk.

A second database file '`aliases`' allows you to give additional names to the files listed in '`ls-R`'. This can be helpful to adapt to "8.3" filename conventions in source files.

The '`ls-R`' and '`aliases`' features are implemented in the source file '`kpathsea/db.c`'.

3.4.1 '`ls-R`'

As mentioned above, you must name the main filename database '`ls-R`'. You can put one at the root of each TeX installation hierarchy you wish to search (`$TEXMF` by default); most sites have only one hierarchy. Kpathsea looks for '`ls-R`' files along the `TEXMFDBS` path, so that should presumably match the list of hierarchies.

The recommended way to create and maintain '`ls-R`' is to run the `mktexlsr` script, which is installed in '`$(bindir)`' ('`/usr/local/bin`' by default). That script goes to some trouble to follow symbolic links as necessary, etc. It's also invoked by the distributed '`mktex...`' scripts.

At its simplest, though, you can build '`ls-R`' with the command

`cd` */your/texmf/root* `&& ls -LAR ./ >ls-R`

presuming your `ls` produces the right output format (see the section below). GNU `ls`, for example, outputs in this format. Also presuming your `ls` hasn't been aliased in a system file (e.g., '`/etc/profile`') to something problematic, e.g., '`ls --color=tty`'. In that case, you will have to disable the alias before generating '`ls-R`'. For the precise definition of the file format, see Section 3.4.3 [Database format], page 35.

Regardless of whether you use the supplied script or your own, you will almost certainly want to invoke it via `cron`, so when you make changes in the installed files (say if you install a new LaTeX package), '`ls-R`' will be automatically updated.

The '`-A`' option to `ls` includes files beginning with '`.`' (except for '`.`' and '`..`'), such as the file '`.tex`' included with the LaTeX tools package. (On the other hand, *directories* whose names begin with '`.`' are always ignored.)

If your system does not support symbolic links, omit the '`-L`'.

`ls -LAR` */your/texmf/root* will also work. But using '`./`' avoids embedding absolute pathnames, so the hierarchy can be easily transported. It also avoids possible trouble with automounters or other network filesystem conventions.

Kpathsea warns you if it finds an '`ls-R`' file, but the file does not contain any usable entries. The usual culprit is running plain '`ls -R`' instead of '`ls -LR ./`' or '`ls -R` */your/texmf/root*'. Another possibility is some system directory name starting with a '`.`' (perhaps if you are using AFS); Kpathsea ignores everything under such directories.

Because the database may be out-of-date for a particular run, if a file is not found in the database, by default Kpathsea goes ahead and searches the disk. If a particular path element begins with '`!!`', however, *only* the database will be searched for that element, never the disk. If the database does not exist, nothing will be searched. Because this can surprise users ("I see the font '`foo.tfm`' when I do an `ls`; why can't Dvips find it?"), it is not in any of the default search paths.

3.4.2 Filename aliases

In some circumstances, you may wish to find a file under several names. For example, suppose a TeX document was created using a DOS system and tries to read '`longtabl.sty`'. But now it's being run on a Unix system, and the file has its original name, '`longtable.sty`'. The file won't be found. You need to give the actual file '`longtable.sty`' an alias '`longtabl.sty`'.

You can handle this by creating a file '`aliases`' as a companion to the '`ls-R`' for the hierarchy containing the file in question. (You must have an '`ls-R`' for the alias feature to work.)

The format of '`aliases`' is simple: two whitespace-separated words per line; the first is the real name '`longtable.sty`', and second is the alias ('`longtabl.sty`'). These must be base filenames, with no directory components. '`longtable.sty`' must be in the sibling '`ls-R`'.

Also, blank lines and lines starting with '`%`' or '`#`' are ignored in '`aliases`', to allow for comments.

If a real file '`longtabl.sty`' exists, it is used regardless of any aliases.

3.4.3 Database format

The "database" read by Kpathsea is a line-oriented file of plain text. The format is that generated by GNU (and most other) `ls` programs given the '`-R`' option, as follows.

- Blank lines are ignored.

- If a line begins with '/' or './' or '../' and ends with a colon, it's the name of a directory. ('../' lines aren't useful, however, and should not be generated.)
- All other lines define entries in the most recently seen directory. /'s in such lines will produce possibly-strange results.
- Files with no preceding directory line are ignored.

For example, here's the first few lines of 'ls-R' (which totals about 30K bytes) on my system:

```
bibtex
dvips
fonts
ls-R
metafont
metapost
tex
web2c

./bibtex:
bib
bst
doc

./bibtex/bib:
asi.bib
btxdoc.bib
...
```

3.5 kpsewhich: Standalone path searching

The Kpsewhich program exercises the path searching functionality independent of any particular application. This can also be useful as a sort of find program to locate files in your TeX hierarchies, perhaps in administrative scripts. It is used heavily in the distributed 'mktex...' scripts.

Synopsis:

kpsewhich *option*... *filename*...

The options and filename(s) to look up can be intermixed. Options can start with either '-' or '--', and any unambiguous abbreviation is accepted.

3.5.1 Path searching options

Kpsewhich looks up each non-option argument on the command line as a filename, and returns the first file found. There is no option to return all the files with a particular name (you can run the Unix 'find' utility for that, see section "Invoking find" in *GNU find utilities*).

Various options alter the path searching behavior:

'`--dpi=`*num*'
: Set the resolution to *num*; this only affects '`gf`' and '`pk`' lookups. '`-D`' is a synonym, for compatibility with Dvips. Default is 600.

'`--format=`*name*'
: Set the format for lookup to *name*. By default, the format is guessed from the filename, with '`tex`' being used if nothing else fits. The recognized filename extensions (including any leading '`.`') are also allowable *names*.

  All formats also have a name, which is the only way to specify formats with no associated suffix. For example, for Dvips configuration files you can use '`--format="dvips config"`'. (The quotes are for the sake of the shell.)

  Here's the current list of recognized names and the associated suffixes. See Section 4.1 [Supported file formats], page 40, for more information on each of these.

```
gf: gf
pk: pk
bitmap font
afm: .afm
base: .base
bib: .bib
bst: .bst
cnf: .cnf
ls-R: ls-R
fmt: .fmt
map: .map
mem: .mem
mf: .mf
mfpool: .pool
mft: .mft
mp: .mp
mppool: .pool
MetaPost support
ocp: .ocp
ofm: .ofm .tfm
opl: .opl
otp: .otp
ovf: .ovf
ovp: .ovp
graphic/figure: .eps .epsi
tex: .tex
TeX system documentation
texpool: .pool
TeX system sources
PostScript header/font: .pro
Troff fonts
tfm: .tfm
type1 fonts: .pfa .pfb
vf: .vf
dvips config
```

```
ist: .ist
truetype fonts: .ttf .ttc
type42 fonts
web2c files
other text files
other binary files
```

This option and '`--path`' are mutually exclusive.

'`--interactive`'
: After processing the command line, read additional filenames to look up from standard input.

'`-mktex=`*filetype*'
'`-no-mktex=`*filetype*'
: Turn on or off the '`mktex`' script associated with *filetype*. The only values that make sense for *filetype* are '`pk`', '`mf`', '`tex`', and '`tfm`'. By default, all are off in Kpsewhich. See Section 2.2.9 [mktex scripts], page 12.

'`--mode=`*string*'
: Set the mode name to *string*; this also only affects '`gf`' and '`pk`' lookups. No default: any mode will be found. See Section 2.2.9.3 [mktex script arguments], page 14.

'`--must-exist`'
: Do everything possible to find the files, notably including searching the disk. By default, only the '`ls-R`' database is checked, in the interest of efficiency.

'`--path=`*string*'
: Search along the path *string* (colon-separated as usual), instead of guessing the search path from the filename. '`//`' and all the usual expansions are supported (see Section 3.3 [Path expansion], page 31). This option and '`--format`' are mutually exclusive. To output the complete directory expansion of a path, instead of doing a one-shot lookup, see '`--expand-path`' in the following section.

'`--progname=`*name*'
: Set the program name to *name*; default is '`kpsewhich`'. This can affect the search paths via the '`.`*prognam*' feature in configuration files (see Section 3.2.1 [Config files], page 30).

3.5.2 Auxiliary tasks

Kpsewhich provides some additional features not strictly related to path lookup:

- '`--debug=`*num*' sets the debugging options to *num*. See Section 2.6.3 [Debugging], page 22.
- '`--expand-braces=`*string*' outputs the variable and brace expansion of *string*. See Section 3.3 [Path expansion], page 31.
- '`--expand-var=`*string*' outputs the variable expansion of *string*. For example, the '`mktex...`' scripts run '`kpsewhich --expand-var='$TEXMF'`' to find the root of the TeX system hierarchy. See Section 3.3 [Path expansion], page 31.

- '`--expand-path=`*string*' outputs the complete expansion of *string* as a colon-separated path. This is useful to construct a search path for a program that doesn't accept recursive subdirectory specifications.

  For one-shot uses of an arbitrary (not built in to Kpathsea) path, see '`--path`' in the previous section.
- '`--show-path=`*name*' shows the path that would be used for file lookups of file type *name*. Either a filename extension ('`pk`', '`.vf`', etc.) or an integer can be used, just as with '`--format`', described in the previous section.

3.5.3 Standard options

Kpsewhich accepts the standard GNU options:

- '`--help`' prints a help message on standard output and exits.
- '`--version`' prints the Kpathsea version number and exits.

4 TEX support

Although the basic features in Kpathsea can be used for any type of path searching, it came about (like all libraries) with a specific application in mind: I wrote Kpathsea specifically for TEX system programs. I had been struggling with the programs I was using (Dvips, Xdvi, and TEX itself) having slightly different notions of how to specify paths; and debugging was painful, since no code was shared.

Therefore, Kpathsea provides some TEX-specific formats and features. Indeed, many of the supposedly generic path searching features were provided because they seemed useful in that conTEXt (font lookup, particularly).

Kpathsea provides a standard way to search for files of any of the supported file types; glyph fonts are a bit different than all the rest. Searches are based solely on filenames, not file contents—if a GF file is named '`cmr10.600pk`', it will be found as a PK file.

4.1 Supported file formats

Kpathsea has support for a number of file types. Each file type has a list of environment and config file variables that are checked to define the search path, and most have a default suffix that plays a role in finding files (see the next section). Some also define additional suffixes, and/or a program to be run to create missing files on the fly.

Since environment variables containing periods, such as '`TEXINPUTS.latex`', are not allowed on some systems, Kpathsea looks for environment variables with an underscore, e.g., '`TEXINPUTS_latex`' (see Section 3.2.1 [Config files], page 30).

The following table lists the above information.

'`afm`'
: (Adobe font metrics, see section "Metric files" in *Dvips*) `AFMFONTS`; suffix '`.afm`'.

'`base`'
: (Metafont memory dump, see section "Memory dumps" in *Web2c*) `MFBASES`, `TEXMFINI`; suffix '`.base`'.

'`bib`'
: (BibTEX bibliography source, see section "bibtex invocation" in *Web2c*) `BIBINPUTS`, `TEXBIB`; suffix '`.bib`'.

'`bst`'
: (BibTEX style file, see section "Basic BibTEX style files" in *Web2c*) `BSTINPUTS`; suffix '`.bst`'.

'`cnf`'
: (Runtime configuration files, see Section 3.2.1 [Config files], page 30) `TEXMFCNF`; suffix '`.cnf`'.

'`dvips config`'
: (Dvips '`config.*`' files, such as '`config.ps`', see section "Config files" in *Dvips*) `TEXCONFIG`.

'`fmt`'
: (TEX memory dump, see section "Memory dumps" in *Web2c*) `TEXFORMATS`, `TEXMFINI`; suffix '`.fmt`'.

'`gf`'
: (generic font bitmap, see section "Glyph files" in *Dvips*) *program*`FONTS`, `GFFONTS`, `GLYPHFONTS`, `TEXFONTS`; suffix '`gf`'.

'`graphic/figure`'
: (Encapsulated PostScript figures, see section "PostScript figures" in *Dvips*) `TEXPICTS`, `TEXINPUTS`; additional suffixes: '`.eps`', '`.epsi`'.

'ist' (makeindex style files) TEXINDEXSTYLE, INDEXSTYLE; suffix '.ist'.

'ls-R' (Filename databases, see Section 3.4 [Filename database], page 34) TEXMFDBS.

'map' (Fontmaps, see Section 4.3.2 [Fontmap], page 44) TEXFONTMAPS; suffix '.map'.

'mem' (MetaPost memory dump, see section "Memory dumps" in *Web2c*) MPMEMS, TEXMFINI; suffix '.mem'.

'MetaPost support'
(MetaPost support files, used by DMP; see section "dmp invocation" in *Web2c*) MPSUPPORT.

'mf' (Metafont source, see section "mf invocation" in *Web2c*) MFINPUTS; suffix '.mf'; dynamic creation program: mktexmf.

'mfpool' (Metafont program strings, see section "pooltype invocation" in *Web2c*) MFPOOL, TEXMFINI; suffix '.pool'.

'mft' (MFT style file, see section "mft invocation" in *Web2c*) MFTINPUTS; suffix '.mft'.

'mp' (MetaPost source, see section "mpost invocation" in *Web2c*) MPINPUTS; suffix '.mp'.

'mppool' (MetaPost program strings, see section "pooltype invocation" in *Web2c*) MPPOOL, TEXMFINI; suffix '.pool'.

'ocp' (Omega compiled process files) OCPINPUTS;
suffix '.ocp'; dynamic creation program: MakeOmegaOCP.

'ofm' (Omega font metrics) OFMFONTS, TEXFONTS;
suffixes '.ofm', '.tfm'; dynamic creation program: MakeOmegaOFM.

'opl' (Omega property lists) OPLFONTS, TEXFONTS; suffix '.opl'.

'otp' (Omega translation process files) OTPINPUTS; suffix '.otp'.

'ovf' (Omega virtual fonts) OVFFONTS, TEXFONTS; suffix '.ovf'.

'ovp' (Omega virtual property lists) OVPFONTS, TEXFONTS; suffix '.ovp'.

'pk' (packed bitmap fonts, see section "Glyph files" in *Dvips*) *PROGRAM*FONTS (*program* being 'XDVI', etc.), PKFONTS, TEXPKS, GLYPHFONTS, TEXFONTS; suffix 'pk'; dynamic creation program: mktexpk.

'PostScript header'
(downloadable PostScript, see section "Header files" in *Dvips*) TEXPSHEADERS, PSHEADERS; additional suffix '.pro'.

'tex' (TeX source, see section "tex invocation" in *Web2c*) TEXINPUTS; suffix '.tex'; additional suffixes: none, because such a list cannot be complete; dynamic creation program: mktextex.

'TeX system documentation'
(Documentation files for the TeX system) TEXDOCS.

'TeX system sources'
(Source files for the TeX system) TEXSOURCES.

'texpool' (TEX program strings, see section "pooltype invocation" in *Web2c*) TEXPOOL, TEXMFINI; suffix '.pool'.

'tfm' (TEX font metrics, see section "Metric files" in *Dvips*) TFMFONTS, TEXFONTS; suffix '.tfm'; dynamic creation program: mktextfm.

'Troff fonts'
(Troff fonts, used by DMP; see section "DMP invocation" in *Web2c*) TRFONTS.

'truetype fonts'
(TrueType outline fonts) TTFONTS; suffixes '.ttf', '.ttc'.

'type1 fonts'
(Type 1 PostScript outline fonts, see section "Glyph files" in *Dvips*) T1FONTS, T1INPUTS, TEXPSHEADERS, DVIPSHEADERS; suffixes '.pfa', '.pfb'.

'type42 fonts'
(Type 42 PostScript outline fonts) T42FONTS.

'vf' (virtual fonts, see section "Virtual fonts" in *Dvips*) VFFONTS, TEXFONTS; suffix '.vf'.

There are two special cases, because the paths and environment variables always depend on the name of the program: the variable name is constructed by converting the program name to upper case, and then appending 'INPUTS'. Assuming the program is called 'foo', this gives us the following table.

'other text files'
(text files used by 'foo') FOOINPUTS.

'other binary files'
(binary files used by 'foo') FOOINPUTS.

If an environment variable by these names are set, the corresponding 'texmf.cnf' definition won't be looked at (unless, as usual, the environment variable value has an extra ':'). See Section 3.3.1 [Default expansion], page 31.

For the font variables, the intent is that:

- TEXFONTS is the default for everything.
- GLYPHFONTS is the default for bitmap (or, more precisely, non-metric) files.
- Each font format has a variable of its own.
- Each program has its own font override path as well; e.g., DVIPSFONTS for Dvipsk. Again, this is for bitmaps, not metrics.

4.2 File lookup

This section describes how Kpathsea searches for most files (bitmap font searches are the exception, as described in the next section).

Here is the search strategy for a file *name*:

1. If the file format defines default suffixes, and the suffix of *name* name is not already a known suffix for that format, try the name with each default appended, and use alternative names found in the fontmaps if necessary. We postpone searching the disk

as long as possible. Example: given 'foo.sty', look for 'foo.sty.tex' before 'foo.sty'. This is unfortunate, but allows us to find 'foo.bar.tex' before 'foo.bar' if both exist and we were given 'foo.bar'.

2. Search for *name*, and if necessary for alternative names found in the fontmaps. Again we avoid searching the disk if possible. Example: given 'foo', we look for 'foo'.
3. If the file format defines a program to invoke to create missing files, run it (see Section 2.2.9 [mktex scripts], page 12).

Under win32, you can change the policy by setting the variable *allow_multiple_suffixes* to *false* either in your environment or in the file 'texmf.cnf'. In this case, if the name looked for has already a suffix (dot plus three characters), only this name will be looked for, no matter if the suffix is known or not. If the name has no suffix, the search reverts to the previous policy.

This is implemented in the routine `kpse_find_file` in 'kpathsea/tex-file.c'. You can watch it in action with the debugging options (see Section 2.6.3 [Debugging], page 22).

4.3 Glyph lookup

This section describes how Kpathsea searches for a bitmap font in GF or PK format (or either) given a font name (e.g., 'cmr10') and a resolution (e.g., 600).

Here is an outline of the search strategy (details in the sections below) for a file *name* at resolution *dpi*. The search stops at the first successful lookup.

1. Look for an existing file *name.dpiformat* in the specified format(s).
2. If *name* is an alias for a file *f* in the fontmap file 'texfonts.map', look for *f.dpi*.
3. Run an external program (typically named 'mktexpk') to generate the font (see Section 2.2.9 [mktex scripts], page 12)
4. Look for *fallback.dpi*, where *fallback* is some last-resort font (typically 'cmr10').

This is implemented in `kpse_find_glyph_format` in 'kpathsea/tex-glyph.c'.

4.3.1 Basic glyph lookup

When Kpathsea looks for a bitmap font *name* at resolution *dpi* in a format *format*, it first checks each directory in the search path for a file '*name*.*dpiformat*'; for example, 'cmr10.600pk'. Kpathsea looks for a PK file first, then a GF file.

If that fails, Kpathsea looks for 'dpi*dpi*/*name*.*format*'; for example, 'dpi600/cmr10.pk'. This is how fonts are typically stored on filesystems (such as DOS) that permit only three-character extensions.

If that fails, Kpathsea looks for a font with a close-enough *dpi*. "Close enough" is defined by the macro `KPSE_BITMAP_TOLERANCE` in 'kpathsea/tex-glyph.h' to be *dpi* `/ 500 + 1`. This is slightly more than the 0.2% minimum allowed by the DVI standard (*CTAN:*`/dviware/driv-standard/level-0`).

4.3.2 Fontmap

If a bitmap font or metric file is not found with the original name (see the previous section), Kpathsea looks through any *fontmap* files for an *alias* for the original font name. These files are named '`texfonts.map`' and searched for along the `TEXFONTMAPS` environment/config file variable. All '`texfonts.map`' files that are found are read; earlier definitions override later ones.

This feature is intended to help in two respects:

1. An alias name is limited in length only by available memory, not by your filesystem. Therefore, if you want to ask for '`Times-Roman`' instead of '`ptmr`', you can (you get '`ptmr8r`').
2. A few fonts have historically had multiple names: specifically, LaTEX's "circle font" has variously been known as '`circle10`', '`lcircle10`', and '`lcirc10`'. Aliases can make all the names equivalent, so that it no longer matters what the name of the installed file is; TEX documents will find their favorite name.

The format of fontmap files is straightforward:

- Comments start with '`%`' and continue to the end of the line.
- Blank lines are ignored.
- Each nonblank line is broken up into a series of *words*: a sequence of non-whitespace characters.
- If the first word is '`include`', the second word is used as a filename, and it is searched for and read.
- Otherwise, the first word on each line is the true filename;
- the second word is the alias;
- subsequent words are ignored.

If an alias has an extension, it matches only those files with that extension; otherwise, it matches anything with the same root, regardless of extension. For example, an alias '`foo.tfm`' matches only when '`foo.tfm`' is being searched for; but an alias '`foo`' matches '`foo.vf`', '`foo.600pk`', etc.

As an example, here is an excerpt from the '`texfonts.map`' in the Web2c distribution. It makes the circle fonts equivalent and includes automatically generated maps for most PostScript fonts available from various font suppliers.

```
circle10        lcircle10
circle10        lcirc10
lcircle10       circle10
lcircle10       lcirc10
lcirc10         circle10
lcirc10         lcircle10
...
include adobe.map
include apple.map
include bitstrea.map
...
```

Fontmaps are implemented in the file 'kpathsea/fontmap.c'. The Fontname distribution has much more information on font naming (see section "Introduction" in *Filenames for TEX fonts*).

4.3.3 Fallback font

If a bitmap font cannot be found or created at the requested size, Kpathsea looks for the font at a set of *fallback resolutions*. You specify these resolutions as a colon-separated list (like search paths). Kpathsea looks first for a program-specific environment variable (e.g., `DVIPSSIZES` for Dvipsk), then the environment variable `TEXSIZES`, then a default specified at compilation time (the Make variable `default_texsizes`). You can set this list to be empty if you prefer to find fonts at their stated size or not at all.

Finally, if the font cannot be found even at the fallback resolutions, Kpathsea looks for a fallback font, typically 'cmr10'. Programs must enable this feature by assigning to the global variable `kpse_fallback_font` or calling `kpse_init_prog` (see Section 5.2 [Calling sequence], page 46); the default is no fallback font.

4.4 Suppressing warnings

Kpathsea provides a way to suppress selected usually-harmless warnings; this is useful at large sites where most users are not administrators, and thus the warnings are merely a source of confusion, not a help. To do this, you set the environment variable or configuration file value `TEX_HUSH` to a colon-separated list of values. Here are the possibilities:

'all' Suppress everything possible.

'checksum'
: Suppress mismatched font checksum warnings.

'lostchar'
: Suppress warnings when a character is missing from a font that a DVI or VF file tries to typeset.

'readable'
: Suppress warnings about attempts to access a file whose permissions render it unreadable.

'special' Suppresses warnings about an unimplemented or unparsable '\special' command.

'tex-hush.c' defines the function that checks the variable value. Each driver implements its own checks where appropriate.

5 Programming

This chapter is for programmers who wish to use Kpathsea. See Chapter 1 [Introduction], page 1, for the conditions under which you may do so.

5.1 Programming overview

Aside from this manual, your best source of information is the source to the programs I've modified to use Kpathsea (see Chapter 1 [Introduction], page 1). Of those, Dviljk is probably the simplest, and hence a good place to start. Xdvik adds VF support and the complication of X resources. Dvipsk adds the complication of its own config files. Web2c is source code I also maintain, so it uses Kpathsea rather straightforwardly, but is of course complicated by the Web to C translation. Finally, Kpsewhich is a small utility program whose sole purpose is to exercise the main path-searching functionality.

Beyond these examples, the '.h' files in the Kpathsea source describe the interfaces and functionality (and of course the '.c' files define the actual routines, which are the ultimate documentation). 'pathsearch.h' declares the basic searching routine. 'tex-file.h' and 'tex-glyph.h' define the interfaces for looking up particular kinds of files. You may wish to use #include <kpathsea/kpathsea.h>, which includes every Kpathsea header.

The library provides no way for an external program to register new file types: 'tex-file.[ch]' must be modified to do this. For example, Kpathsea has support for looking up Dvips config files, even though no program other than Dvips will likely ever want to do so. I felt this was acceptable, since along with new file types should also come new defaults in 'texmf.cnf' (and its descendant 'paths.h'), since it's simplest for users if they can modify one configuration file for all kinds of paths.

Kpathsea does not parse any formats itself; it barely opens any files. Its primary purpose is to return filenames. The GNU font utilities does contain libraries to read TFM, GF, and PK files, as do the programs above, of course.

5.2 Calling sequence

The typical way to use Kpathsea in your program goes something like this:

1. Call kpse_set_program_name with argv[0] as the first argument; the second argument is a string or NULL. The second argument is used by Kpathsea as the program name for the *.program* feature of config files (see Section 3.2.1 [Config files], page 30). If the second argument is NULL, the value of the first argument is used. This function must be called before any other use of the Kpathsea library.

   If necessary, kpse_set_program_name sets the global variables program_invocation_name and program_invocation_short_name. These variables are used in the error message macros defined in 'kpathsea/lib.h'. It sets the global variable kpse_program_name to the program name it uses. It also initializes debugging options based on the environment variable KPATHSEA_DEBUG (if that is set). Finally, it sets the variables SELFAUTOLOC, SELFAUTODIR and SELFAUTOPARENT to the location, parent and grandparent directory of the executable, removing '.' and '..' path elements and resolving symbolic links. These are used in the default configuration file to allow people to invoke TeX from anywhere, specifically from a mounted CD-ROM. (You can use '--expand-var=\$SELFAUTOLOC', etc., to see the values finds.)

2. The kpse_set_progname is deprecated. A call to kpse_set_progname with argv[0] is equivalent to a call of kpse_set_program_name with first argument argv[0] and second argument NULL. The function is deprecated because it cannot ensure that the *.program* feature of config files will always work (see Section 3.2.1 [Config files], page 30).
3. Set debugging options. See Section 2.6.3 [Debugging], page 22. If your program doesn't have a debugging option already, you can define one and set kpathsea_debug to the number that the user supplies (as in Dviljk and Web2c), or you can just omit this altogether (people can always set KPATHSEA_DEBUG). If you do have runtime debugging already, you need to merge Kpathsea's options with yours (as in Dvipsk and Xdvik).
4. If your program has its own configuration files that can define search paths, you should assign those paths to the client_path member in the appropriate element of the kpse_format_info array. (This array is indexed by file type; see 'tex-file.h'.) See 'resident.c' in Dvipsk for an example.
5. Call kpse_init_prog (see 'proginit.c'). It's useful for the DVI drivers, at least, but for other programs it may be simpler to extract the parts of it that actually apply. This does not initialize any paths, it just looks for (and sets) certain environment variables and other random information. (A search path is always initialized at the first call to find a file of that type; this eliminates much useless work, e.g., initializing the BibTEX search paths in a DVI driver.)
6. The routine to actually find a file of type *format* is kpse_find_*format*, defined in 'tex-file.h'. These are macros that expand to a call to 'kpse_find_file'. You can call, say, kpse_find_tfm after doing only the first of the initialization steps above—Kpathsea automatically reads the 'texmf.cnf' generic config files, looks for environment variables, and does expansions at the first lookup.
7. To find PK and/or GF bitmap fonts, the routines are kpse_find_pk, kpse_find_gf and kpse_find_glyph, defined in 'tex-glyph.h'. These return a structure in addition to the resultant filename, because fonts can be found in so many ways. See the documentation in the source.
8. To actually open a file, not just return a filename, call kpse_open_file. This function takes the name to look up and a Kpathsea file format as arguments, and returns the usual FILE *. It always assumes the file must exist, and thus will search the disk if necessary (unless the search path specified '!!', etc.). In other words, if you are looking up a VF or some other file that need not exist, don't use this.

Kpathsea also provides many utility routines. Some are generic: hash tables, memory allocation, string concatenation and copying, string lists, reading input lines of arbitrary length, etc. Others are filename-related: default path, tilde, and variable expansion, stat calls, etc. (Perhaps someday I'll move the former to a separate library.)

The 'c-*.h' header files can also help your program adapt to many different systems. You will almost certainly want to use Autoconf for configuring your software if you use Kpathsea; I strongly recommend using Autoconf regardless. It is available from ftp://prep.ai.mit.edu/pub/gnu/.

5.3 Program-specific files

Many programs will need to find some configuration files. Kpathsea contains some support to make it easy to place them in their own directories. The Standard TeX directory structure (see section "Introduction" in *A Directory Structure for TeX files*), specifies that such files should go into a subdirectory named after the program, like '`texmf/ttf2pk`'.

Two special formats, '`kpse_program_text_format`' and '`kpse_program_binary_format`' exist, which use `.:$TEXMF/`*program*`//` as their compiled-in search path. To override this default, you can use the variable *PROGRAM*`INPUTS` in the environment and/or '`texmf.cnf`'. That is to say, the name of the variable is constructed by converting the name of the program to upper case, and appending `INPUTS`.

The only difference between these two formats is whether `kpse_open_file` will open the files it finds in text or binary mode.

5.4 Programming with config files

You can (and probably should) use the same `texmf.cnf` configuration file that Kpathsea uses for your program. This helps installers by keeping all configuration in one place.

To retrieve a value *var* from config files, the best way is to call `kpse_var_value` on the string *var*. This will look first for an environment variable *var*, then a config file value. The result will be the value found or '`NULL`'. This function is declared in '`kpathsea/variable.h`'. For an example, see the `shell_escape` code in '`web2c/lib/texmfmp.c`'.

The routine to do variable expansion in the context of a search path (as opposed to simply retrieving a value) is `kpse_var_expand`, also declared in '`kpathsea/variable.h`'. It's generally only necessary to set the search path structure components as explained in the previous section, rather than using this yourself.

If for some reason you want to retrieve a value *only* from a config file, not automatically looking for a corresponding environment variable, call `kpse_cnf_get` (declared in '`kpathsea/cnf.h`') with the string *var*.

No initialization calls are needed.

Index

(Index is nonexistent)

Table of Contents