Kpathsea library

Karl Berry
Olaf Weber
Taco Hoekwater
http://tug.org/kpathsea

This file documents the Kpathsea library for path searching.

Copyright © 1996–2014 Karl Berry & Olaf Weber.

Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of this manual into another language, under the above conditions for modified versions, except that this permission notice may be stated in a translation approved by the TeX Users Group.

Table of Contents

1 Introduction

This manual corresponds to version 6.2.0 of the Kpathsea library, released in May 2014.

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 is maintained in parallel, uses this library:

- Dviljk (see the 'dvilj' man page)
- Dvipsk (see *Dvips: A DVI driver*)
- GNU font utilities (see *GNU font utilities*)
- Web2c (see *Web2c: A T_EX implementation*)
- Xdvik (see the 'xdvi' man page)

Other software that we do not maintain also uses it.

Kpathsea is now maintained as part of the T_EX Live distribution (`http://tug.org/texlive`). For information on configuration, building, installing, and more, see *Building T_EX Live*.

The library is still actively maintained. If you have comments or suggestions, please send along (see Chapter 8 [Reporting bugs], page 35).

The Kpathsea library is distributed 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 accompanying files for the text of the GNU licenses, or `http://www.gnu.org/licenses`.

If you know enough about T_EX to be reading this manual, then you (or your institution) should consider joining the T_EX Users Group (if you're already a member, thanks!). TUG produces the periodical *TUGboat*, sponsors an annual meeting and publishes the proceedings, and arranges courses on T_EX for all levels of users throughout the world. See `http://tug.org` for information.

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 T_EX" paper by Don Knuth, published in his book *Digital Typography*, among other places.

[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. When I started, 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 *T_EX 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. That is, I tried to go with the flow in each program, rather than changing the program's calling sequences to conform to new routines.

Then, as bugs inevitably appeared, I found I was fixing the same thing three times (Web2c and fontutils were already sharing code, since I maintained both of 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 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.

[Karl writes again.] Indeed, time goes on. As of 2006 or so, Olaf's available time for Kpathsea became rather reduced, and I started taking overall care of it again, although I did not do any significant new development. In 2009, Taco Hoekwater made a major rearrangement to make the library suitable for use within the MetaPost library (see Section 7.1 [Programming overview], page 31). Also, for some years now, Peter Breitenlohner has made many improvements to the infrastructure and kept up-to-date with respect to the overall TeX Live build, where Kpathsea is now maintained.

2 `unixtex.ftp`: Obtaining TeX

This chapter is `ftp://ftp.tug.org/tex/unixtex.ftp`, last updated 13 June 2010. Also available as `http://www.tug.org/unixtex.ftp`. Email `tex-k@tug.org` with comments or questions.

The principal free TeX distribution for Unix-like systems is TeX Live, on the web at `http://tug.org/texlive`. The pages there describe many ways to acquire and build TeX, over the Internet or on physical media, both the sources and precompiled binaries for many systems, either standalone or as part of various operating system distributions.

Web2C, Kpathsea, Dvips, and Dviljk are no longer released as a separate packages. Their sources are now maintained as part of TeX Live.

The host `ftp.cs.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 undertake the very significant project of building your TeX installation from scratch, it is far more reliable and less work to retrieve these files as part of a larger package

In any case, the Stanford ftp site is not the canonical source for anything except what was created as part of Knuth's original TeX, so do not rely on any other files available there being up-to-date. The best place to check for up-to-date files is CTAN (the Comprehensive TeX Archive Network), `http://www.ctan.org`.

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 6.5.1 [mktex configuration], page 27.

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 6.5.1 [mktex configuration], page 27.

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.

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 document is available from `http://www.gnu.org/prep/standards`. The TDS document is available from `http://www.mirror.ctan.org/tds` (see Chapter 2 [unixtex.ftp], page 3).

In short, 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')
        dpinn/               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
```

5 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.

5.1 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 5.4 [Filename database], page 12) 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 might very well 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 not 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 any read permission (for example, a lost+found directory). 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 6.2 [File lookup], page 24, and Section 6.3 [Glyph lookup], page 25).

5.2 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 8.3 [Debugging], page 36).

These sources may be combined via default expansion (see Section 5.3.1 [Default expansion], page 9).

5.2.1 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`.

If Kpathsea cannot find any `texmf.cnf` file, it reports a warning including all the directories it checked. If you don't want to see this warning, set the environment variable `KPATHSEA_WARNING` to the single character 'O' (zero, not oh).

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 '%', either at the beginning of a line or preceded by whitespace, and continue to the end of the line. That is, as with most shells, a '%' in the "middle" of a value does not start a comment. Examples:

        % this is a comment
        var = a%b % but the value of var will be "a%b".

- 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,bat,cmd,...}`. Most notably, this allows different flavors of TeX to have different search paths.

- *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 Windows.

- 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
```

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. However, there seemed little advantage in 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`.

5.3 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.

5.3.1 Default expansion

If the highest-priority search path (see Section 5.2 [Path sources], page 8) 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
```

Put another way, default expansion works on "formats" (search paths), and not directly on environment variables. Example, showing the trailing ':' ignored in the first case and expanded in the second:

```
$ env TTFONTS=/tmp: kpsewhich --expand-path '$TTFONTS'
/tmp
$ env TTFONTS=/tmp: kpsewhich --show-path=.ttf
/tmp:.:/home/olaf/texmf/fonts/truetype//:...
```

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 8.3 [Debugging], page 36). Default expansion is implemented in the source file `kpathsea/kdefault.c`.

5.3.2 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 5.2 [Path sources], page 8).

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

5.3.3 Tilde expansion

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

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`.

5.3.4 Brace expansion

'`x{a,b}y`' expands to '`xay:xby`'. 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:xB2y`'.

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 instead 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`.

5.3.5 `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 6.5 [mktex scripts], page 27, because these change the working directory. You should not ever define it yourself.

5.3.6 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 `ST_NLINK_TRICK` 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`.

5.4 Filename database (`ls-R`)

Kpathsea goes to some lengths to minimize disk accesses for searches (see Section 5.3.6 [Subdirectory expansion], page 11). Nevertheless, in practice searching each possible directory in typical TeX installations takes an excessively long time.

Therefore, Kpathsea can use an externally-built *filename 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`.

5.4.1 `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 5.4.3 [Database format], page 13.

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.

5.4.2 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.

5.4.3 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
...
```

5.5 `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.

5.5.1 Path searching options

Kpsewhich looks up each non-option argument on the command line as a filename, and returns the first file found.

Various options alter the path searching behavior:

'`--all`'   Report all matches found, one per line. By default, if there is more than one match, just one will be reported (chosen effectively at random).

'`--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.

'`--engine=`*name*'

> Set the engine name to *name*. By default it is not set. The engine name is used in some search paths to allow files with the same name but used by different engines to coexist.
>
> In particular, since the memory dump files (`.fmt`/`.base`/`.mem`) are now stored in subdirectories named for the engine (`tex`, `pdftex`, `xetex`, etc.), you must specify an engine name in order to find them. For example, `cont-en.fmt` typically exists for both `pdftex` and `xetex`. With the default path settings, you can use '`--engine=/`' to look for any dump file, regardless of engine; if a dump file exists for more than one engine, it's indeterminate which one is returned. (The '`/`' ends up specifying a normal recursive search along the path where the dumps are stored, namely '`$TEXMF/web2c{/$engine,}`'.)

'`--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 *name*s.
>
> 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 6.1 [Supported file formats], page 21, for more information on each of these.
>
> The strings in parentheses are abbreviations recognized only by `kpsewhich` (not the underlying library calls). They are provided when it would otherwise require an argument containing a space to specify the format, to simplify quoting of calls from shells.

```
gf: gf
pk: pk
bitmap font (bitmapfont):
tfm: .tfm
afm: .afm
base: .base
bib: .bib
bst: .bst
cnf: .cnf
ls-R: ls-R ls-r
fmt: .fmt
map: .map
mem: .mem
mf: .mf
mfpool: .pool
mft: .mft
mp: .mp
mppool: .pool
MetaPost support (mpsupport):
```

```
ocp: .ocp
ofm: .ofm .tfm
opl: .opl  .pl
otp: .otp
ovf: .ovf .vf
ovp: .ovp  .vpl
graphic/figure:  .eps .epsi
tex: .tex  .sty .cls .fd .aux .bbl .def .clo .ldf
TeX system documentation (doc):
texpool: .pool
TeX system sources (source):  .dtx .ins
PostScript header:  .pro
Troff fonts (trofffont):
type1 fonts: .pfa .pfb
vf: .vf
dvips config (dvipsconfig):
ist: .ist
truetype fonts: .ttf .ttc .TTF .TTC .dfont
type42 fonts: .t42 .T42
web2c files (web2c):
other text files (othertext):
other binary files (otherbin):
misc fonts (miscfont):
web: .web  .ch
cweb: .w .web  .ch
enc files: .enc
cmap files (cmap):
subfont definition files: .sfd
opentype fonts: .otf
pdftex config (pdftexconfig):
lig files: .lig
texmfscripts:
lua: .lua .luatex .luc .luctex .texlua .texluc .tlu
font feature files: .fea
cid maps: .cid .cidmap
mlbib: .mlbib .bib
mlbst: .mlbst .bst
clua: .dll .so
ris: .ris
bltxml: .bltxml
```

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*. Usual values for
> *filetype* are '`pk`', '`mf`', '`tex`', and '`tfm`'. By default, all are off in Kpsewhich,
> even if they are enabled for TeX. This option implies setting `--must-exist`.
> See Section 6.5 [mktex scripts], page 27.

'`--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 6.5.3 [mktex script arguments],
> page 30.

'`--must-exist`'

> Do everything possible to find the files, notably including searching the disk
> and running the '`mktex`' scripts. 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 5.3 [Path expansion], page 9). 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`' and '`--show-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 5.2.1
> [Config files], page 8).

'`--safe-in-name=`*name*'
'`--safe-out-name=`*name*'

> Exit successfully if *name* is safe to open for reading or writing, respectively, else
> unsuccessfully. No output is written. These tests take account of the related
> Kpathsea configuration settings (see Section 7.2 [Calling sequence], page 31).

'`--subdir=`*string*'

> Report only those matches whose directory part *ends* with *string* (compared
> literally, except case is ignored on a case-insensitive operating system). For
> example, suppose there are two matches for a given name:

```
kpsewhich foo.sty
⇒ /some/where/foo.sty
/another/place/foo.sty
```

> Then we can narrow the result to what we are interested in with `--subdir`:

```
kpsewhich --subdir=where foo.sty
⇒ /some/where/foo.sty

kpsewhich --subdir=place foo.sty
⇒ /another/place/foo.sty
```

The string to match must be at the end of the directory part of the match, and it is taken literally, with no pattern matching:

```
kpsewhich --subdir=another foo.sty
⇒
```

The string to match may cross directory components:

```
kpsewhich --subdir=some/where foo.sty
⇒ /some/where/foo.sty
```

`--subdir` implies `--all`; if there is more than one match, they will all be reported (in our example, both '`where`' and '`place`' end in '`e`'):

```
kpsewhich --subdir=e
⇒ /some/where/foo.sty
/another/place/foo.sty
```

Because of the above rules, the presence of a leading '`/`' is important, since it "anchors" the match to a full component name:

```
kpsewhich --subdir=/lace foo.sty
⇒
```

However, a trailing '`/`' is immaterial (and ignored), since the match always takes place at the end of the directory part:

```
kpsewhich --subdir=lace/ foo.sty
⇒ /another/place/foo.sty
```

The purpose of these rules is to make it convenient to find results only within a particular area of the tree. For instance, a given script named `foo.lua` might exist within both `texmf-dist/scripts/pkg1/` and `texmf-dist/scripts/pkg2/`. By specifying, say, '`--subdir=/pkg1`', you can be sure of getting the one you are interested in.

We only match at the end because a site might happen to install TeX in `/some/coincidental/pkg1/path/`, and we wouldn't want to match `texmf-dist/scripts/pkg2/` that when searching for '`/pkg1`'.

5.5.2 Specially-recognized files for `kpsewhich`

`kpsewhich` recognizes a few special filenames on the command line and defaults to using the 'known' file formats for them, merely to save the time and trouble of specifying the format. This is only a feature of `kpsewhich`; when using the Kpathsea library itself, none of these special filenames are recognized, and it's still up to the caller to specify the desired format.

Here is the list of special filenames to `kpsewhich`, along with their corresponding format:

`config.ps`

> dvips config

`dvipdfmx.cfg`

> 'other text files'

`fmtutil.cnf`

> 'web2c files'

`glyphlist.txt`

        'map'

`mktex.cnf`

        'web2c files'

`pdfglyphlist.txt`

        'map'

`pdftex.cfg`

        'pdftex config' (although `pdftex.cfg` is not used any more; look for the file
        `pdftexconfig.tex` instead.)

`texmf.cnf`

        'cnf'

`XDvi`   'other text files'

A user-specified format will override the above defaults.

Another useful configuration file in this regard is `tcfmgr.map`, found in `texmf/texconfig/tcfmgr.map`, which records various information about the above configuration files (among others).

5.5.3 Auxiliary tasks

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

- '`--debug=`*num*' sets the debugging options to *num*. See Section 8.3 [Debugging], page 36.
- '`--var-value=`*variable*' outputs the value of *variable*, expanding '`$`' (see Section 5.3.2 [Variable expansion], page 10 and '`~`' (see Section 5.3.3 [Tilde expansion], page 10) constructs, but not performing other expansions.
- '`--expand-braces=`*string*' outputs the variable and brace expansion of *string*. See Section 5.3 [Path expansion], page 9.
- '`--expand-var=`*string*' outputs the variable and tilde expansion of *string*. For example, the '`mktex...`' scripts run '`kpsewhich --expand-var='$TEXMF'`' to find the root of the TeX system hierarchy. See Section 5.3 [Path expansion], page 9.
- '`--expand-path=`*string*' outputs the complete expansion of *string*, with each element separated by the usual path separator on the current system ('`;`' on Windows, '`:`' otherwise). This may be useful to construct a custom search path for a format not otherwise supported. To retrieve the search path for a format that is already supported, see '`--show-path`', next.

  Nonexistent directories are culled from the output:

        $ kpsewhich --expand-path '/tmp'
        ⇒ /tmp
        $ kpsewhich --expand-path '/nonesuch'
        ⇒

  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.

5.5.4 Standard options

Kpsewhich accepts the standard GNU options:

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

6 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.

6.1 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 5.2.1 [Config files], page 8).

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, see Section "Basic BibTEX style files" in *Web2c*) `BSTINPUTS`; suffix '`.bst`'.

'`cmap`'   (character map files) `CMAPFONTS`; suffix '`.cmap`'.

'`cnf`'   (Runtime configuration files, see Section 5.2.1 [Config files], page 8) `TEXMFCNF`; suffix '`.cnf`'.

'`cweb`'   (CWEB input files) `CWEBINPUTS`; suffixes '`.w`', '`.web`'; additional suffix '`.ch`'.

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

'`enc files`'
         (encoding vectors) `ENCFONTS`; suffix '`.enc`'.

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

'font cid map'
               (CJK mapping) FONTCIDMAPS suffix '.cid'.

'font feature files'
               (primarily for OpenType font features) FONTFEATURES suffix '.fea'.

'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'.

'lig files'
               (ligature definition files) LIGFONTS; suffix '.lig'.

'ls-R'   (Filename databases, see Section 5.4 [Filename database], page 12) TEXMFDBS.

'map'   (Fontmaps, see Section 6.3.2 [Fontmap], page 25) 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'.

'misc fonts'
               (font-related files that don't fit the other categories) MISCFONTS

'mlbib'   (MlBibTEX bibliography source) MLBIBINPUTS, BIBINPUTS, TEXBIB; suffixes
               '.mlbib', '.mlbib'.

'mlbst'   (MlBibTEX style) MLBSTINPUTS, BSTINPUTS; suffixes '.mlbst', '.bst'.

'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.

`opentype fonts`
: (OpenType fonts) `OPENTYPEFONTS`.

`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`'.

`pdftex config`
: (PDFTeX-specific configuration files) `PDFTEXCONFIG`.

`pk`
: (packed bitmap fonts, see Section "Glyph files" in *Dvips*) `PROGRAMFONTS` (*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`'.

`subfont definition files`
: (subfont definition files) `SFDFONTS` suffix '`.sfd`'.

`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`.

`texmfscripts`
: (Architecture-independent executables distributed in the texmf trees) `TEXMFSCRIPTS`.

`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`' and '`.TTF`', '`.ttc`' and '`.TTC`', '`.dfont`'.

`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`'.

'web'   (WEB input files) `WEBINPUTS`; suffix '`.web`'; additional suffix '`.ch`'.

'`web2c files`'
            (files specific to the web2c implementation) `WEB2C`.

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 5.3.1 [Default expansion], page 9.

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.

6.2 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. Example: given '`foo.bar`', look for '`foo.bar.tex`'.

2. Search for *name*, and if necessary for alternative names found in the fontmaps. Example: given '`foo.bar`', we also look for '`foo.bar`'.

3. If the file format defines a program to invoke to create missing files, run it (see Section 6.5 [mktex scripts], page 27).

The order in which we search for "suffixed" name (item 1) or the "as-is" name (item 2) is controlled by the `try_std_extension_first` configuration value. The default set in `texmf.cnf` is true, since common suffixes are already recognized: '`babel.sty`' will only look for '`babel.sty`', not '`babel.sty.tex`', regardless of this setting.

When the suffix is unknown (e.g., '`foo.bar`'), both names are always tried; the difference is the order in which they are tried.

try_std_extension_first only affects names being looked up which *already* have an extension. A name without an extension (e.g., 'tex story') will always have an extension added first.

This algorithm is implemented in the function kpathsea_find_file in the source file kpathsea/tex-file.c. You can watch it in action with the debugging options (see Section 8.3 [Debugging], page 36).

6.3 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 6.5 [mktex scripts], page 27)

4. Look for *fallback.dpi*, where *fallback* is some last-resort font (typically 'cmr10').

This is implemented in kpathsea_find_glyph in kpathsea/tex-glyph.c.

6.3.1 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 '*dpidpi/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*).

6.3.2 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 old and new names of the LaTeX circle fonts equivalent.

```
circle10          lcircle10
circle10          lcirc10
lcircle10         circle10
lcircle10         lcirc10
lcirc10           circle10
lcirc10           lcircle10
...
```

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

6.3.3 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 calling kpathsea_init_prog (see Section 7.2 [Calling sequence], page 31); the default is no fallback font.

6.4 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.

'`none`'   Don't suppress any warnings.

'`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.

6.5 `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 6.5.1 [mktex configuration], page 27). These are collectively known as `mktex` *scripts*, since most of them are named `mktex....`

For example, this is 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. Building all fonts in advance is therefore impractical, if not impossible.

It is also useful for the TEX '`.fmt`' (and Metafont '`.base`' and Metapost '`.mem`' files, see Section "Memory dumps" in *web2c*), where pre-generating every format consumes a lot of both time and space.

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.

6.5.1 `mktex` configuration

The list of file types and program names that can run an external program to create missing files is listed in the next section. 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-mktexfmt-default
--without-mktexmf-default
--without-mktexocp-default
--without-mktexofm-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 6.5.3 [mktex script arguments], page 30).

`mktexfmt` reads a file `fmtutil.cnf`, typically located in `texmf/web2c/` to glean its configuration information. The rest of the files and features in this section are primarily intended for the font generation scripts.

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 Core 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` environ-

ment 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.

'texmfvar'

Force generated files that would go into a system tree (as defined by `SYSTEXMF`) into `TEXMFVAR`. Starting with teTeX-3.0, the variable `TEXMFVAR` is always set. The 'varfonts' feature takes precedence if also set.

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

6.5.2 `mktex` script names

The following table shows the default name of the script for each of the file types which support runtime generation.

mktexfmt   ('.fmt', '.base', '.mem') TeX/Metafont/MetaPost formats. This script is also named `fmtutil`, and reads `fmtutil.cnf` for configuration information.

mktexmf   ('.mf') Metafont input files.

mkocp   ('.ocp') Omega compiled process files.

`mkofm`   ('`.ofm`') Omega font metric files.

`mktexpk`   ('`pk`') Glyph fonts.

`mktextex`   ('`.tex`') TEX input files (disabled by default).

`mktextfm`   ('`.tfm`') TFM files.

These names can be 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.

6.5.3 `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 "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.

7 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.

7.1 Programming overview

Aside from this manual, your best source of information is the source to the programs that 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.

When looking at these program sources, you should know that previous versions of the library had a different programming interface, to support re-entrancy. In that interface the library function names were prefixed with `kpse_` instead of `kpathsea_`, and they did not need an instance variable as first argument. This change was made in 2009. Some of the programs mentioned above may still be using the previous interface.

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. In view of the way the headers depend on each other, it is recommended to use `#include <kpathsea/kpathsea.h>`, which includes every Kpathsea header.

If you want to include only specific headers, you should still consider including `kpathsea/config.h` before including any other Kpathsea header, as it provides symbols used in the other headers. Note that `kpathsea/config.h` includes `kpathsea/c-auto.h`, which is generated by Autoconf.

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.

7.2 Calling sequence

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

1. Call `kpathsea_new` to create a new library instance. This variable must be passed as the first argument to all the following library functions. The rest of this manual will be using `kpse` as a placeholder for the name of this variable.

2. Call `kpathsea_set_program_name` with `argv[0]` as the second argument; the third argument is a string or `NULL`. The third argument is used by Kpathsea as the program name for the `.program` feature of config files (see Section 5.2.1 [Config files], page 8). If the third argument is `NULL`, the value of the second argument is used. This function must be called before any other use of the Kpathsea library.

   `kpathsea_set_program_name` always sets the variables `kpse->invocation_name` and `kpse->invocation_short_name`. These variables are used in the error message macros defined in `kpathsea/lib.h`. It sets the 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 environment 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. You can use '`kpsewhich --expand-var=\$SELFAUTOLOC`', etc., to see the values.

3. Set debugging options. See Section 8.3 [Debugging], page 36. If your program doesn't have a debugging option already, you can define one and set `kpse->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 `kpathsea_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 `kpathsea_find_file`. You can call `kpathsea_find_file` after doing only the first and second 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 routine is `kpathsea_find_glyph`, defined in `tex-glyph.h`. This returns 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 `kpathsea_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.

9. TeX can write output files, via the `\openout` primitive; this opens a security hole vulnerable to Trojan horse attack: an unwitting user could run a TeX program that

overwrites, say, `~/.rhosts`. Analogous security holes exist for many other programs. To alleviate this, there is a configuration variable `openout_any`, which selects one of three levels of security. When it is set to 'a' (for "any"), no restrictions are imposed. When it is set to 'r' (for "restricted"), filenames beginning with '.' are disallowed (except `.tex` because LaTeX needs it). When it is set to 'p' (for "paranoid") additional restrictions are imposed: an absolute filename must refer to a file in (a subdirectory) of `TEXMFOUTPUT`, and any attempt to go up a directory level is forbidden (that is, paths may not contain a '..' component). The paranoid setting is the default. (For backwards compatibility, 'y' and '1' are synonyms of 'a', while 'n' and '0' are synonyms for 'r'.) The function `kpathsea_out_name_ok`, with a filename as second argument, returns `true` if that filename is acceptable to be opend for output or `false` otherwise.

10. Similarly, the function `kpathsea_in_name_ok`, with a filename as second argument, returns `true` if that filename is acceptable to be opend for input or `false` otherwise, depending on the value of the configuration variable `openin_any` (with 'a' as default).

11. To close the kpathsea library instance you are using, call `kpathsea_finish`. This function closes any open log files and frees the memory used by the instance.

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 and probably Automake for configuring and building your software if you use Kpathsea; I strongly recommend using Autoconf and Automake regardless. They are available from `http://www.gnu.org/software`.

7.3 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 formats, 'kpse_program_text_format' and 'kpse_program_binary_format', 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 `kpathsea_open_file` will open the files it finds in text or binary mode.

7.4 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 `kpathsea_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 `kpathsea_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 `kpathsea_cnf_get` (declared in `kpathsea/cnf.h`) with the string *var*.

No initialization calls are needed.

8 Reporting bugs

If you have problems or suggestions, please report them to `tex-k@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, . . .

8.1 Bug checklist

Before reporting a bug, please check below to be sure it isn't already known (see Section 8.5 [Common problems], page 38).

Bug reports should be sent via electronic mail to `tex-k@tug.org`.

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), compiler, and `make` program you are using (the output of `uname -a` is a start on the first two, though incomplete).

- 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 required, because that's the actual program input.

- 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 (`http://www.gnu.org/software/gdb`). 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 8.5.4 [TeX or
Metafont failing], page 40).

- Any additional information that will be helpful in reproducing, diagnosing, or fixing
  the bug.

8.2 Mailing lists

Web2c and Kpathsea in general are discussed on the mailing list `tex-k@tug.org`. You can
subscribe and peruse the archives on the web `http://lists.tug.org/tex-k`.

You do not need to join to submit a report, nor will it affect whether you get a response.
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 are looking for general TeX help, such as how to install a full TeX system or how
to use LaTeX, please see `http://tug.org/begin.html`.

8.3 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 what-
ever) 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 instance variable
`kpse->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 5.5 [Invoking
kpsewhich], page 14), instead of the original program.

In any case, you cannot 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 struc-
> ture is not forcing Kpathsea to do many additional file tests (see Section 8.5.2
> [Slow path searching], page 39, and see Section 5.3.6 [Subdirectory expansion],
> page 11). If you are using an up-to-date `ls-R` database (see Section 5.4 [File-
> name database], page 12), 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 5.4 [Filename database], page 12); font aliases (see Section 6.3.2 [Fontmap], page 25); and config file values (see Section 5.2.1 [Config files], page 8). 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.)

8.4 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.

In addition to this Kpathsea-specific logging, pdftex provides an option -recorder to write the names of all files accessed during a run to the file *basefile*.fls.

Finally, most systems provide a general tool to output each system call, thus including opening and closing files. It might be named strace, truss, struss, or something else.

8.5 Common problems

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

8.5.1 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 8.3 [Debugging], page 36.

- Perhaps you simply haven't installed all the necessary files; the basic fonts and input files are distributed separately from the programs. See Chapter 2 [unixtex.ftp], page 3.
- 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 6.1 [Supported file formats], page 21). 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 5.3.6 [Subdirectory expansion], page 11.

  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 Chapter 2 [unixtex.ftp], page 3).

- There is a bug in the library. See Chapter 8 [Reporting bugs], page 35.

8.5.2 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 5.4 [Filename database], page 12. 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 5.3.6 [Subdirectory expansion], page 11).

  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 8.3 [Debugging], page 36.

8.5.3 Unable to generate fonts

Metafont outputs fonts in bitmap format, tuned for a particular device at a particular resolution, in order to allow for the highest-possible quality of output. Some DVI-to-whatever programs, such as Dvips, try to generate these on the fly when they are needed, but this generation may fail in several cases.

If `mktexpk` runs, but fails with this error:

```
mktexpk: Can't guess mode for nnn dpi devices.
mktexpk: Use a config file to specify the mode, or update me.
```

you need to ensure the resolution and mode match; just specifying the resolution, as in `-D 360`, is not enough.

You can specify the mode name with the `-mode` option on the Dvips command line, or in a Dvips configuration file (see Section "Config files" in *Dvips*), such as `config.ps` in your document directory, `~/.dvipsrc` in your home directory, or in a system directory (again named `config.ps`). (Other drivers use other files, naturally.)

For example, if you need 360 dpi fonts, you could include this in a configuration file:

```
D 360
M lqmed
```

If Metafont runs, but generates fonts at the wrong resolution or for the wrong device, most likely `mktexpk`'s built-in guess for the mode is wrong, and you should override it as above.

See `http://ctan.org/pkg/modes` for a list of resolutions and mode names for most devices (additional submissions are welcome).

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*.

If `mf` is a command not found at all by `mktexpk`, then you need to install Metafont (see Chapter 2 [unixtex.ftp], page 3).

8.5.4 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.

Index

T

U

V

W

X

Z