Kpathsea library

for version 2.6
January 1995

Karl Berry

1 Introduction

This manual corresponds to version 2.6 of the Kpathsea library, released in January 1995.

The library's fundamental purpose is to look up a file in 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 I maintain, uses this library:

- Dviljk
- Dvipsk (see section "Introduction" in *Dvips: A DVI driver*)
- GNU font utilities (see section "Introduction" in *GNU font utilities*)
- Web2c (see section "Introduction" in *Web2c: A TEX implementation*)
- Xdvik

The library is still under development (and probably always will be, despite my hopes). I do not promise to keep the interface unchanged. If you have comments or suggestions, please send them to me (see Section 2.4 [Reporting bugs], page 11).

Currently, I distribute the library under the GNU General Public License (see Appendix A [Copying], page 33). In short, this means if you write a program using the library, you must (offer to) distribute the source, and allow anyone to modify the source and distribute their modifications.

If you have a problem with this, contact me. I would consider putting the library under the GNU Library General Public License, which would permit you to distribute the source only to the library, not to your program using it. But I will only do this if someone actually says they will not use the library under the GPL conditions, and would use it under the LGPL.

If you know enough about TEX to be reading this manual, then you (or perhaps your institution) should consider joining the TEX Users Group (if you're already a member, great!). TUG produces a periodical called *TUGboat*, sponsors an annual meeting (the proceedings of which are published in *TUGboat*), and arranges courses on TEX for all levels of users. Given sufficient funding (which your joining will help) TUG could sponsor more projects that will benefit the TEX community, such as a successor to TEX π . Anyway, here is the address:

TEX Users Group
P.O. Box 869
Santa Barbara, CA 93102 USA

phone: (805) 899-4673
email: 'tug@tug.org'

1.1 History

(This section is for those people who are curious about how this came about.) (If you like to read historical accounts of software, I urge you to seek out the GNU Autoconf manual and, even more fun, the "Errors of TEX" paper that Don Knuth published in *Software—Practice and Experience*.)

My first ChangeLog entry for Web2c seems to be February 1990, but I may have done some stuff before then. In any case, Tim Morgan and I were sort of jointly maintaining it for a time. (I should say 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.)

It must have been later in 1990 and 1991 that I started working on *TeX for the Impatient* and Dvips, Xdvi, Web2c, and the GNU fontutils (which I was also writing at the time) using different environment variables, and, even more importantly, having different bugs in their path searching became extremely painful. 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 'pandora', 'cm' in a path.

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

Then, as bugs inevitably appeared, I found I was fixing the same thing in each of three (Web2c and fontutils were always sharing code, since I maintained those—there was no Dvipsk or Xdvik or Dviljk at this point). After a while, I finally started sharing source files. They weren't 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.)

Things kept on like this for quite a while. The 'ChangeLog's 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 starting to make it 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; I think it was a conversation with david zuhn that led to doing 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!)

At this point (October 1994), I've implemented just about all the path-searching features in Kpathsea that I ever intended to (and some I didn't intend ...). After the next stable release of Web2c, I figure I'll be able to stop development, and turn most of my attention back to making fonts for GNU. (Always assuming Microsoft 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...)

2 Installation

Here are the basic steps for configuration and installation:

1. Edit the file 'make/paths.make' if you want to make changes to the installation directories or paths that will have effect across different runs of configure. Alternatively, override the Make variables on the command line when you run Make.

   Exception: to reliably change the top-level 'prefix', you must give 'configure' the option '-prefix=*prefix*', instead of changing the value in 'paths.make'.

2. Edit 'kpathsea/texmf.cnf.in' to change the local paths to match your local setup. See section "Default paths" in *Kpathsea*, for more details on changing the paths. A copy is in 'kpathsea/INSTALL'. See 'kpathsea/HIER' for an explanation of the default setup.

   If the paths do not match where the files actually are, the programs will probably start up Very, Very, Slowly, and/or not be able to find the fonts or other input files.

3. sh configure (in the top-level directory). This makes system-dependent '#define's' in '*/c-auto.h' (from the corresponding 'c-auto.h.in') and creates a 'Makefile' (from the corresponding 'Makefile.in', by doing '@*var*@' and 'ac_include' substitutions).

   Perhaps the most common desire is to compile with optimization instead of or as well as debugging. You can change the options passed to the compiler by changing 'CFLAGS', either for configure or make. For example:

   ```
   prompt$ env CFLAGS="-g -O" configure
   prompt$ make
   ```

   or

   ```
   prompt$ configure
   prompt$ make CFLAGS="-g -O"
   ```

   See section "Running configure scripts" in *Autoconf*, for detailed configure options. (A copy is in 'kpathsea/CONFIGURE'.)

4. make (still in the top-level directory). Barring configuration and compiler bugs, this will compile all the programs. See section "Common problems" in *Kpathsea*, for system-dependent problems (this section is also in 'kpathsea/INSTALL').

   This also creates the 'texmf.cnf' and 'paths.h' files that define the default search paths.

1. make install. This installs the library, header files, and documentation. Or make install-data to just install the architecture-independent files. Or make install-exec to just install the (binary) archive library file.

   Since I only distribute Kpathsea as part of another package, you will probably be doing the above in a top-level directory that contains a 'Makefile', 'kpathsea', and the other package. But you can do the installation in 'kpathsea' itself, if you only want to install the library, not the other package.

2. The first time you install any manual in Info, you have to add a line (you choose where) to the 'dir' file in your '$(infodir)' directory. A sample line to add is given near the top of the Texinfo source files ('kpathsea/kpathsea.texi' and 'dvipsk/dvips.texi').
3. `make distclean`. This removes all files created by the build.

See Section 4.7 [Filename database], page 20, for a description of an externally-generated database that can help speed searches.

See Chapter 3 [Debugging], page 12, for runtime debugging support that may help track down problems.

Do not attempt to use any version of Kpathsea with any program except the version that the program came with, unless you are a glutton for punishment.

2.1 Default paths

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

1. 'configure' creates a 'Makefile' from each 'Makefile.in'.
2. When Make runs in the 'kpathsea' directory, it creates a file 'texmf.sed' that substitutes the Make value of `$(var)` for a string `@var@`. The variables in question are the one that define the installation directories.
3. 'texmf.sed' (and a little extra magic—see 'kpathsea/Makefile') is applied to 'texmf.cnf.in' to generate 'texmf.cnf'. This is the file that will eventually be installed and used by the programs to look up programs.
4. The definitions in 'texmf.cnf' are changed into the form of C '#define''s, producing 'paths.h'. These values will be the compile-time defaults; they are not used unless no 'texmf.cnf' file can be found at runtime.

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

The purpose of this elaborate sequence is to avoid having the same information in more than one place. If you change the installation directories or top-level prefix before running 'configure', those changes will propagate through the whole sequence. If you change the default paths in 'texmf.cnf.in', those changes are propagated to the compile-time defaults.

Alternatively, you can ignore the whole mess and edit '`texmf.cnf`' after it is installed. Maybe even copying it into place beforehand so you can complete the installation, if TeX or Metafont is having trouble finding their input files.

Unfortunately, editing '`Makefile.in`' *does not work* in one common case—changing the `prefix` or `exec_prefix` variables. For these, you must use the '`-prefix`' or '`-exec-prefix`' options to `configure`. See section "Running `configure` scripts" in *Autoconf*. (That's another partial lie: editing does work, as long as a program named `tex` is not in your `PATH`.)

See Chapter 6 [TeX directory structure], page 29, for a description of some ways to arrange the TeX library files, and some features of the distributed paths that may not be obvious. The file '`kpathsea/HIER`' is a copy of that section.

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

2.2 Common problems

Some common problems with compilation, linking, or execution are described below.

2.2.1 Unable to find files

If a program complains it cannot find fonts (or other input files), any of several things might be wrong:

- You don't have the fonts (or whatever) installed. Nothing will automatically generate TFM files or TeX and Metafont sources for you (by default). See section "Obtaining Web2c" in *Web2c*.

  You can, however, configure TeX and Metafont to run a script to generate these input files, if you have (or write) such scripts. See section "'`MakeTeX`'... invocation" in *Web2c*.
- 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, thus overriding what you carefully set in '`texmf.cnf`'. See Section 5.1 [TeX environment variables], page 23.

- Your files reside in a directory that is only pointed to via a symbolic link, in a leaf directory.

  Unfortunately, Kpathsea's subdirectory searching has a (congenital) 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 4.6 [Subdirectory expansion], page 19, for an explanation of why this happens.

  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. This is since presumably you would not have asked for the directory to be searched for subdirectories if you didn't want it to be.

- There is a bug in the library. See Section 2.4 [Reporting bugs], page 11.

In any case, you may find the debugging options helpful in determining precisely where the fonts (or whatever) are being looked for. See the program's documentation for its debugging options, and also see Chapter 3 [Debugging], page 12.

2.2.2 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 4.7 [Filename database], page 20. (Kpathsea always uses '`ls-R`' if it's present; there's no need to recompile or reinstall any of the programs.)

- Your recursively-searched directories (e.g., '`/usr/local/lib/tex/fonts//`'), contain a mixture of files and directories. This prevents Kpathsea from using a useful optimization (see Section 4.6 [Subdirectory expansion], page 19).

  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 directories where the dozens of TFM, PK, or whatever files reside.

- Finally, one simple-to-fix (but unlikely) cause: If you recursively search '`$HOME`' or '`~`', and you are running as '`root`', you will search every directory on the system. This typically takes quite some time!

In any case, you may find the debugging options helpful in determining precisely when the disk or network is being pounded. See Chapter 3 [Debugging], page 12, and also see the program's documentation.

2.2.3 XtInherit

On DEC OSF/1 1.x systems, the loader has a bug that manifests itself in the following error (all on one line, but for the sake of the paper width it's broken here):

```
xdvik/xdvi: /sbin/loader: Fatal Error: search_for_undefineds:
     symbol _XtInherit should not have any relocation entry
```

According to Michael Rickabaugh ‘<mjr@quarry.enet.dec.com>’:

> This is a bug fixed in DEC OSF/1 2.0.
>
> If you know how, installing ‘/sbin/loader’ from a 2.0 system onto a 1.3 system will work. Make sure that ‘/usr’ is *not* mounted when you do this. (If you forget about umounting /usr, it is possible most of your filesystems will become corrupted.)
>
> Otherwise, I suggest getting a hold of a 2.0 CD and running ‘/usr/sbin/installupdate’.

Alternatively, you may be able to use the freely available X11 libraries that come with the MIT distribution (on ‘ftp.x.org’, for example).

Linking statically, perhaps only with some of the X libraries, may also work. (if you find the definitive workaround, please let me know.)

2.2.4 wchar_t

The upshot of all the following is that if you get error messages regarding wchar_t, try defining NO_FOIL_X_WCHAR_T (for Web2c) or FOIL_X_WCHAR_T (for everything else).

wchar_t has caused infinite trouble. None of my code ever uses wchar_t; all I want to do is include X header files and various system header files, possibly compiling with GCC. This seems an impossible task!

The X11 header '<Xlib.h>' and GCC's '<stddef.h>' have conflicting definitions for wchar_t.

The particulars: '<X11/Xlib.h>' from MIT X11R5 defines `wchar_t` if `X_WCHAR` is defined, which is defined if `X_NOT_STDC_ENV` is defined, and we define *that* if `STDC_HEADERS` is not defined ('`configure`' decides if STDC_HEADERS gets defined). But when compiling with gcc on SunOS 4.1.x, `STDC_HEADERS` is not defined ('`string.h`' doesn't declare the '`mem`'* functions), so we do get X's `wchar_t`—and we also get gcc's `wchar_t` from its '<stddef.h>'. Conflict.

On the other hand, SunOS 4.1.1 with some other X configurations actually needs GCC to define `wchar_t`, and fails otherwise.

My current theory is to define `wchar_t` to a nonsense symbol before the X include files are read; that way its definition (if any) will be ignored by other system include files. Going along with that, define `X_WCHAR` to tell X not to use '<stddef.h>', that we've already included, but instead to make its own definition.

But this is not the end of the story. The X11 include files distributed with DG/UX 5.4.2 for the Aviion have been modified to include '<_int_wchar_t.h>' if `X_WCHAR`, so our `#define` will not have any typedef to change—but the uses of `wchar_t` in the X include files will be changed to reference this undefined symbol. So there's nothing to foil in this case. I don't know how to detect this automatically, so it's up to you to define `NO_FOIL_X_WCHAR_T` yourself.

2.2.5 `ShellWidgetClass`

This section is adapted from question 47 from the '`comp.sys.sun.admin`' FAQ.

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

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

- Get the Openwindows patches that apply to this problem.
- Statically link the `Xmu` library into the executable.
- Avoid using `Xmu` at all. For this last, if you are compiling Metafont, see section "Online Metafont graphics" in *web2c*. If you are compiling Xdvi, see the `-DNOTOOL` option in '`xdvik/INSTALL`'.

- Ignore the errors. The binary runs fine regardless.

Here is the information for getting the two patches:

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

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

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

```
-Bstatic -lXmu -Bdynamic
```

If you are using `gcc`, include '`-static`' in '`LDFLAGS`'; this will link all libraries statically. If you want to link only `Xmu` statically and everything else dynamically, you have to do it by hand: run `gcc -v`, grab the `ld` line, and add the '`-B`''s given above around `-lXmu`.

The reason is that gcc moves all linker options to the front of the `ld` command line. So you can't specify different options for different libraries. When I reported this to the GCC maintainers, the reply was that they would happily merge in the changes, but they didn't want to take the time to do it themselves.

2.2.6 Pointer combination warnings

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

In general, 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 bug list.)

2.3 Shared library

You can compile Kpathsea as a shared library. The advantage in doing this is that the different executables can then share the code, decreasing memory usage. (The other advantage in general of shared libraries is that it's possible to update the library and programs independently. But since the Kpathsea interface is not and can not be frozen, that doesn't apply here.)

Under Solaris, use '`-K pic -xstrconst`' if you compile with a Sun compiler, '`-fpic`' if you use GCC. Also add '`-L`*$(libdir)* `-R`*$(libdir)*' to '`LDFLAGS`' when you link the binaries, so that the library can be found, and users do not have set '`LD_LIBRARY_PATH`'.

(If you know how to make Kpathsea shared on other systems, please send a message to the bug address in the next section.)

2.4 Reporting bugs

If you encounter problems, please report them to '`tex-k@cs.umb.edu`'. Include the version number of the library, the system you are using, and enough information to reproduce the bug in your report. To get on this mailing list yourself, email '`tex-k-request@cs.umb.edu`' with a message whose body contains a line

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

To avoid wasted effort and time (both mine and yours), I strongly advise applying the principles given in the GNU C manual (see section "Reporting Bugs" in *The GNU CC manual*) to your bug reports.

Please also report bugs in this documentation—not only factual errors, but unclear explanations, typos, wrong fonts, ...

3 Debugging

Kpathsea provides a number of runtime debugging options, detailed below by their names (and corresponding numeric values). 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, except for Dviljk, are different from those below).

You can also set the environment variable KPATHSEA_DEBUG. In this case, you should use the numbers below. Also use the numbers below if you run the program under a debugger and set the the variable 'kpathsea_debug' yourself.

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

KPSE_DEBUG_STAT
: (1). Reports 'stat'(2) calls. This is useful for verifying that your directory structure is not forcing Kpathsea to do many additional file tests (see Section 2.2.2 [Slow path searching], page 7 and see Section 4.6 [Subdirectory expansion], page 19). If you are using an up-to-date 'ls-R' database (see Section 4.7 [Filename database], page 20), this should produce no output unless a nonexistent file is searched for.

KPSE_DEBUG_HASH
: (2). Reports lookups in all hash tables, including 'ls-R' (see Section 4.7 [Filename database], page 20), font aliases (see Section 5.2.2 [Fontmap], page 25), and config file values (see Section 4.2.1 [Config files], page 16). 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). Reports file openings and closings. Especially useful when your system's file table is full, for seeing if some files have been opened but never closed. In case you want to set breakpoints: this works by redefining 'fopen' ('fclose') to be 'kpse_fopen_trace' ('kpse_fclose_trace').

KPSE_DEBUG_PATHS
: (8). Reports 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', the compile-time default, an environment variable, etc. This is the contents of a structure defined in 'tex-file.h'.

KPSE_DEBUG_EXPAND
: (16). Reports the directory list corresponding to each path element Kpathsea searches in. This is only relevant when Kpathsea is searching the disk, since 'ls-R' searches

don't look through directory lists in this way (they go straight to the file using the hash table).

`KPSE_DEBUG_SEARCH`
: (32). Reports on each file search Kpathsea attempts: 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.

Debugging output from Kpathsea is always written to standard error, and begins with '`kdebug:`'. (Except for hash table buckets, which just start with the number.)

3.1 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 disk is full.

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.

Each successful search turns into one line in the log file, with 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/lib/texmf/dvips/config.ps
774455887 /usr/local/lib/texmf/dvips/psfonts.map
774455888 /usr/local/lib/texmf/dvips/texc.pro
774455888 /usr/local/lib/texmf/fonts/public/cm/pk/ljfour/cmbx10.600pk
774455889 /usr/local/lib/texmf/fonts/public/cm/pk/ljfour/cmsl10.600pk
774455889 /usr/local/lib/texmf/fonts/public/cm/pk/ljfour/cmr10.600pk
774455889 /usr/local/lib/texmf/dvips/texc.pro
```

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

4 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.

4.1 Searching overview

A *search path* is a colon-separated list of path elements, which are directory names with some 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`', (typically) returning the first one that exists.

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 path element *e*, Kpathsea first sees if a prebuilt database (see below) 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, contains no matches, the filesystem is searched. Kpathsea constructs the list of directories that correspond to this path element, and then checks in them 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.)

Each path element is checked in turn: first the database, then the disk. Once a match is found, the searching stops and the result is returned. This avoids possibly-expensive processing of path specifications that are never needed on a particular run.

Although the simplest and most common path element is a directory name, Kpathsea supports additional features in search paths: layers of 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 getting rid of all the magic specifications and getting down to 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 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; it is not looked for along any paths.

4.2 Path sources

A search path can come from many sources. In priority order (meaning Kpathsea will use whichever it finds first):

1. A user-set environment variable, e.g., 'TEXINPUTS'.
2. A program-specific configuration file, e.g., an 'S /a:/b' line in Dvips' 'config.ps'.
3. A line in a Kpathsea configuration file 'texmf.cnf', e.g., 'TEXINPUTS=/c:/d'. See section below.
4. The compile-time default (specified in 'kpathsea/paths.h').

In any case, once the path specification to use is determined, its evaluation is independent of its source. These sources may also be combined via default expansion. See the next section.

You can see each of these values for a given search path by using the debugging options of Kpathsea or your program. See Chapter 3 [Debugging], page 12.

4.2.1 Config files

As mentioned above, Kpathsea reads *runtime configuration files* named 'texmf.cnf' for search path definitions. The path used to search for them 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, for technical reasons).

The environment variable used is 'TEXMFCNF'.

Kpathsea reads *all* 'texmf.cnf' files in the search path, not just the first one found; it uses the first definition of each variable encountered. Thus, with the (default) search path of '.:$TEXMF', values from './texmf.cnf' override those from '$TEXMF/texmf.cnf'.

Here is the format for 'texmf.cnf' files:

- Anything after a '%' or '#' is ignored; this is for comments.
- Blank lines are ignored.
- Each remaining nonblank line must look like

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

  where the '=' and surrounding whitespace is optional.
- The *variable* name may contain any characters except whitespace, '=', or '.' characters, but sticking to 'A-Za-z_' is safest.
- If the '.*progname*' is present, the definition only applies if the program that is running is named (i.e., the last component of argv[0]) *progname*. This allows (for example) different flavors of TeX to have different search paths.
- The *value* may contain any characters except whitespace, '%', and '@'. (These restrictions are necessary because of the various sed and other processing done on 'texmf.cnf' at build time.)
- All definitions are read before anything is expanded, so you can use variables before they are defined (like make, unlike most everything else).

Here is the fragment from the distributed file 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 = .:$TEXMF/tex/latex2e//:$TEXMF/tex//
TEXINPUTS = .:$TEXMF/tex//
TEXINPUTS.latex209 = $latex209_inputs
TEXINPUTS.latex2e = $latex2e_inputs
TEXINPUTS.latex = $latex2e_inputs
```

Although this format has obvious similarities to Bourne shell scripts—change the comment character to #, disallow spaces around the =, and get rid of the .*program* convention, and it could be run through the shell. But there seemed little advantage to doing this, since all the information would have to passed back (with echo's, presumably) 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'.

4.3 Default expansion

If the highest-priority search path (in the list in the previous section) contains an *extra colon* (i.e., leading, trailing, or doubled), Kpathsea inserts the next-highest-priority search path that is set at that point. If that search path has an extra colon, the same happens with the next-highest.

(An extra colon in the compile-time default value has unpredictable results, and may cause the program to crash, so installers beware.)

For example, given

```
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
```

You can trace this by debugging "paths" (see Chapter 3 [Debugging], page 12).

Minor technical point: 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 (where they will eventually be effectively equivalent to '.', i.e., the current directory). It checks first for a leading ':', then a trailing ':', then a doubled ':'.

4.4 Variable expansion

'$foo' or '${foo}' in a path element is replaced by (1) the value of an environment variable 'foo' (if it is set); (2) the value of 'foo' from 'texmf.cnf' (if any such exists); (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 are not balanced!). Otherwise, Kpathsea gives a warning and ignores the '$' and its following character.

Remember to quote the '$''s and braces as necessary for your shell.

Shell variable values cannot be seen by Kpathsea.

For example, given

```
setenv TEXMF /home/tex
setenv TEXINPUTS .:$TEXMF:${TEXMF}new
```

the final 'TEXINPUTS' path is the three directories:

```
.:/home/tex:/home/texnew
```

You can trace this by debugging "paths" (see Chapter 3 [Debugging], page 12).

4.5 Tilde expansion

A leading '~' or '~*user*' in a path element is replaced by the current or *user*'s home directory, respectively.

If *user* is invalid, or the home directory cannot be determined, Kpathsea uses '.' instead.

For example,

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

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

4.6 Subdirectory expansion

A '//' 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 contain 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'.

I should mention one related implementation trick, which I stole from GNU find. Matthew Farwell '<dylan@ibmpcug.co.uk>' suggested it, and David MacKenzie '<djm@gnu.ai.mit.edu>' implemented it (as far as I know).

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 *one* subdirectory and five hundred files, `st_nlink` will be 3, and Kpathsea has to stat every one of those 501 entries. Therein lies slowness.

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

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

4.7 Filename database (`ls-R`)

Kpathsea goes to some lengths to minimize disk accesses for searches (see Section 4.6 [Subdirectory expansion], page 19). Nevertheless, at installations with enough directories, doing a linear search of each possible directory for a given file can take an excessively long time ("excessive" depending on the speed of the disk, whether it's NFS-mounted, how patient you are, etc.). In practice, the union of font directories from the Dvips(k) and Dviljk distributions is large enough for searching to be noticeably slow on typical machines these days.

Therefore, Kpathsea can use an externally-built "database" that maps files to directories, thus avoiding the need to exhaustively search the disk. By fiat, you must name the file '`ls-R`', and put it at the root of the TeX installation hierarchy ('`$TEXMF`' by default). Kpathsea does variable expansion on the '`$TEXMF`', naturally, so you can use different '`ls-R`''s for different trees, if you are testing new ones. However, one and only one '`ls-R`' is read; it is not searched for along any paths.

You can build '`ls-R`' with the command

`ls -R` */your/root/dir* `>ls-R`

if your `ls` produces the right output format (see the section below). GNU `ls`, for example, outputs in this format. It is probably best to do this via `cron`, so changes in the installed files will be automatically reflected (albeit with some delay) in the database.

If your system uses symbolic links, the command `ls -LR` will be more reliable than plain `ls -R`. The former follows the symbolic links to the real files, which is what Kpathsea needs.

Kpathsea warns you if it finds an '`ls-R`' file, but the file does not contain any usable entries. The usual culprit is using just `ls -R` to generate the '`ls-R`' file instead of `ls -R` */your/dir*. Kpathsea looks for lines starting with '`/`', to improve reliability with unusual filenames (specifically, those ending with a '`:`').

Because the database may be out-of-date for a particular run (e.g., if a font was just built with `MakeTeXPK`), 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 greatly surprise users ("I see the font '`foo.tfm`' when I do an `ls`; why can't Dvips find it?"), I do not recommend using this feature.

4.7.1 Database format

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

- Blank lines are ignored.
- If a line begins with '`/`' and ends with a colon, it's the name of a directory.
- All other lines name 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`' on my system:

```
bibtex
dvips
fonts
ini
ls-R
mf
```

```
tex

/usr/local/lib/texmf/bibtex:
bib
bst
doc

/usr/local/lib/texmf/bibtex/bib:
asi.bib
bibshare
btxdoc.bib
```

On my system, 'ls-R' is about 30K bytes.

5 TEX searching

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 features. Indeed, many of the supposedly generic path searching features were provided because they seemed useful in that conTEXt (font lookup, particularly).

5.1 TEX environment variables

Kpathsea defines a sequence of environment variables to search for each file type it supports. This makes it easy for different programs to check the same environment variables, in the same order.

The following table lists the environment variables searched for each file type in the order they are searched (and a brief description of the file type). That is, only if the first variable is unset is the second variable checked, and so on. If none are set, various other things are checked; see Section 4.2 [Path sources], page 16.

'`.base`' (Metafont memory dump) '`MFBASES`'

'`.bib`' (BibTEX bibliography source) '`BIBINPUTS`'

'`.bst`' (BibTEX style file) '`BSTINPUTS`', '`TEXINPUTS`'

'`.cnf`' (Kpathsea runtime configuration files) '`TEXMFCNF`'

'`.eps`' (Encapsulated PostScript figures) '`TEXPICTS`', '`TEXINPUTS`'

'`.fmt`' (TEX memory dump) '`TEXFORMATS`'

'`gf`' (generic font bitmap) '*program*`FONTS`', '`GFFONTS`', '`GLYPHFONTS`', '`TEXFONTS`'

'`.mf`' (Metafont source) '`MFINPUTS`'

'`mf.pool`' (Metafont program strings) '`MFPOOL`'

'`.pict`' (Other kinds of figures) Same as '`.eps`'.

'pk' (packed bitmap font) '*program*FONTS', 'PKFONTS', 'TEXPKS', 'GLYPHFONTS', 'TEXFONTS'

'.tex' (TEX source) 'TEXINPUTS'

'tex.pool'
(TEX program strings) 'TEXPOOL'

'.tfm' (TEX font metrics) 'TFMFONTS', 'TEXFONTS'

'.vf' (virtual font) 'VFFONTS', 'TEXFONTS'

For the font variables, the intent is that:

1. 'TEXFONTS' is the default for everything.
2. 'GLYPHFONTS' is the default for bitmap (or, more precisely, non-metric) files.
3. Each format has its own variable.
4. Each program can and should have its own font override path as well; e.g., 'DVIPSFONTS' for Dvipsk. Again, this is for bitmaps, not metrics.

If these environment variables are set, the corresponding 'texmf.cnf' definition won't be looked at (unless, as usual, the environment variable has an extra ':'). See Section 4.3 [Default expansion], page 17.

5.2 Glyph lookup

Kpathsea provides a routine (kpse_find_glyph_format in 'kpathsea/tex-glyph.c') which searches for a bitmap font in GF or PK format (or either) given a font name (e.g., 'cmr10') and a resolution (e.g., 300).

The search is based solely on filenames, not file contents—if a PK file is named 'cmr10.300gf', it will be found as a GF file.

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.dpi* 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 script (typically named MakeTeXPK) to generate the font.
4. Look for *fallback.dpi*, where *fallback* is some last-resort font (typically 'cmr10').

5.2.1 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, 'cmr10.300pk'. Kpathsea looks for a PK file first, then a GF file.

If that fails, Kpathsea looks for 'dpi*dpi*/*name*.*format*'; for example, 'dpi300/cmr10.pk'. This is how fonts are typically stored on filesystems (like DOS's) 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`, which is slightly more than the 0.2% allowed by the DVI standard.

5.2.2 Fontmap

If a bitmap font 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 are searched for along the usual glyph path.

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 'Adobe-Lucida-Bold-Sans=Typewriter' instead of 'plcbst', you can.
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: the first word on each line is the true filename; the second word is the alias; subsequent words are ignored. A *word* is a sequence of non-whitespace characters. Blank lines are ignored; comments start with '%' and continue to end-of-line.

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 exactly 'foo.tfm' is being searched for; but an alias 'foo' matches 'foo.vf', 'foo.300pk', etc.

As an example, here are the fontmap entries that make the circle fonts equivalent. These are in the distributed '`texfonts.map`' in the Web2C distribution.

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

5.2.3 '`MakeTeX`'... scripts

If Kpathsea cannot find a bitmap font, by either its original name or a fontmap alias, it can be configured to invoke an external program to create it. The same mechanism can be used for other nonexistent files.

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.

5.2.3.1 '`MakeTeX`'... script names

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

'`MakeTeXPK`'
: Glyph fonts.

'`MakeTeXTeX`'
: TEX input files.

'`MakeTeXMF`'
: Metafont input files.

'`MakeTeXTFM`'
: TFM files.

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

If a MakeTeX... script fails, the invocation is appended to a file 'missfont.log' in the current directory. If the current directory is not writable and the environment variable 'TEXMFOUTPUT' is set, its value is used. Otherwise, nothing is written.

5.2.3.2 'MakeTeX'... script arguments

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

For 'MakeTeXPK', three or four additional arguments are also passed, via corresponding environment variables:

1. The dpi to make the font at ('KPATHSEA_DPI').
2. The "base dpi" the program is operating at ('MAKETEX_BASE_DPI'), i.e., the assumed resolution of the output device.
3. A "magstep" string suitable for the Metafont mag variable ('MAKETEX_MAG').
4. Optionally, a Metafont mode name to assign to the Metafont mode variable ('MAKETEX_MODE'). Otherwise, (the default) MakeTeXPK guesses the mode from the resolution. See Chapter 6 [TeX directory structure], page 29.
5. Optionally, 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 (from environment variables DESTDIR or MTP_DESTDIR or a compile-time default). If this argument is not supplied, the mode name is appended to the root destination directory.

Kpathsea sets 'KPATHSEA_DPI' appropriately for each attempt at building a font. It's up to the program using Kpathsea to set the others. (See Section 7.2 [Calling sequence], page 30.)

You can change the specification for the arguments passed to the external script by setting the environment variable named as the script name, but all capitals—'MAKETEXPK', for example. If you've changed the script name by setting (say) 'DVIPSMAKEPK' to 'foo', then the spec is taken from the environment variable 'FOO'.

The spec can contain any variable references, to the above variables or any others you might have set. As an example, the default spec for MakeTeXPK is:

```
$KPATHSEA_DPI $MAKETEX_BASE_DPI $MAKETEX_MAG $MAKETEX_MODE
```

The convention of passing the name of the file to be created as the first argument cannot be changed.

5.2.4 Fallback font

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

Finally, if the font cannot be found even at the fallback resolutions, Kpathsea looks for a fallback font, typically '`cmr10`'. Programs must enable this feature by assigning to the global variable `kpse_fallback_font` or calling `kpse_init_prog` (see Section 7.2 [Calling sequence], page 30); the default is no such fallback font.

6 TEX directory structure

(This section obviously not really written yet; sorry. See 'kpathsea/HIER'.)

By default, the bitmap font paths end with $MAKETEX_MODE, thus including the device name (i.e., the Metafont mode) in the path. This is to make it possible to distinguish two different devices with the same resolution—write/white and write/black 300dpi printers, for example.

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

To make the paths independent of the mode, simply edit 'texmf.cnf.in' before installation, or the installed 'texmf.cnf'. See Section 2.1 [Default paths], page 5.

See Section 5.2.3.2 [MakeTeX... script arguments], page 27, for how this interacts with MakeTeXPK.

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. (If you do this, I'd appreciate a note, just to satisfy my curiousity.)

7.1 Programming overview

Aside from this manual, your best source of information is the source to the programs I've modified to use Kpathsea (also listed in the introduction). 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.

Beyond these of examples of use, 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.

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 obviously no program other than Dvips will 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 best for users if they can modify one configuration file for all kinds of paths.

Kpathsea does not open any files or parse any formats itself. Its purpose is only to return filenames. The GNU font utilities source does contain libraries to read TFM, GF, and PK files.

7.2 Calling sequence

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

1. Call `kpse_set_progname` with `argv[0]`; This is the only initialization that is mandatory to take full advantage of Kpathsea—specifically, for the *.program* feature of config files (see Section 4.2.1 [Config files], page 16).

kpse_set_progname sets the global variables program_invocation_name and program_invocation_short_name. It also initializes debugging flags based on the environment variable KPATHSEA_DEBUG, if that is set. set.

The GNU C library provides these two global variables itself; in this case, the call to kpse_set_program does nothing. But you (as a software author) most likely do not want to force people installing your program to have glibc.

2. Set debugging options. See Chapter 3 [Debugging], page 12. If your program doesn't have a debugging flag already, you can define one and set 'kpathsea_debug' to the number that the user supplies (see Dviljk), 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 (see Dvipsk and Xdvik).
3. 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.
4. Call kpse_init_prog (see 'proginit.c'). It's useful for the DVI drivers, at least, but for other programs it may be simpler to extract the parts of it that actually apply. This does not initialize any paths, it just looks for (and sets) certain environment variables and other random information. (A search path is always initialized at the first call to find a file of that type; this eliminates much useless work, e.g., initializing the BibTEX search paths in a DVI driver.)
5. The routine to actually find a file of type *format* is kpse_find_*format*, defined in 'tex-file.h'. These are macros that expand to a call to 'kpse_find_file'. You can call, say, kpse_find_tfm after doing only the first of the initialization steps above—Kpathsea will read the generic config file 'texmf.cmf', look for environment variables, and do the expansions at the first lookup.
6. To find PK and/or GF bitmap fonts, the routines are 'kpse_find_pk', 'kpse_find_gf' and 'kpse_find_glyph', defined in 'tex-glyph.h'. These return a structure in addition to the resultant filename, because fonts can be found in so many ways. See the documentation in the source.

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

The 'c-*.h' header files can also help your program adapt to many different systems. You will almost certainly want to use Autoconf for configuring your software if you use Kpathsea; I strongly recommend using Autoconf regardless. You can get it by ftp from 'prep.ai.mit.edu' in 'pub/gnu/autoconf-*.tar.gz', or from any of its mirrors.

7.3 Programming with config files

You can use the same `texmf.cnf` configuration file as Kpathsea for your program. This will help installers do all configuration in one place.

To retrieve a value *var*, the best way is to call `kpse_var_expand` 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 the empty string. This function is declared in '`kpathsea/variable.h`'.

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

No initialization calls are needed.

Appendix A GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

1. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.
2. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.
3. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:
   a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

4. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

5. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
6. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.
7. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.
8. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

9. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

10. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

    Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

11. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

12. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DE-

FECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

13. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
one line to give the program's name and a brief idea of what it does.
Copyright (C) 19yy  name of author

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) 19yy name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type 'show c' for details.
```

The hypothetical commands '`show w`' and '`show c`' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than '`show w`' and '`show c`'; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
‘Gnomovision’ (which makes passes at compilers) written by James Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Appendix B Regain your programming freedom

Until a few years ago, programmers in the United States could write any program they wished. This freedom has now been taken away by two developments: software patents, which grant the patent holder an absolute monopoly on some programming technique, and user interface copyright, which forbid compatible implementations of an existing user interface.

In Europe, especially through the GATT treaty, things are rapidly approaching the same pass.

B.1 Software patents

The U.S. Patent and Trademark Office has granted numerous software patents on software techniques. Patents are an absolute monopoly—independent reinvention is precluded. This monopoly lasts for seventeen years, i.e., forever (with respect to computer science).

One patent relevant to TeX is patent 4,956,809, issued to the Mark Williams company on September 11, 1990, applied for in 1982, which covers (among other things)

> representing in a standardized order consisting of a standard binary structure file stored on auxiliary memory or transported on a communications means, said standardized order being different from a different order used on at least one of the different computers;
>
> Converting in each of the different computers binary data read from an auxiliary data storage or communications means from the standardized order to the natural order of the respective host computer after said binary data are read from said auxiliary data storage or communications means and before said binary data are used by the respective host computer; and
>
> Converting in each of the different computers binary data written into auxiliary data storage or communications means from the natural order of the respective host computer to the standardized order prior to said writing.

… in other words, storing data on disk in a machine-independent order, as the DVI, TFM, GF, and PK file formats specify. Even though TeX is “prior art” in this respect, the patent was granted (the patent examiners not being computer scientists, even less computer typographers). Since there is a strong presumption in the courts of a patent’s validity once it has been granted, there is a good chance that users or implementors of TeX could be successfully sued on the issue.

As another example, the X window system, which was intended to be able to be used freely by everyone, is now being threatened by two patents: 4,197,590 on the use of exclusive-or to redraw cursors, held by Cadtrak, a litigation company (this has been upheld twice in court); and 4,555,775, held by AT&T, on the use of backing store to redraw windows quickly.

Here is one excerpt from a recent mailing by the League for Programming Freedom (see Section B.3 [What to do?], page 44) which I feel sums up the situation rather well. It comes from an article in *Think* magazine, issue #5, 1990. The comments after the quote were written by Richard Stallman.

> "You get value from patents in two ways," says Roger Smith, IBM Assistant General Counsel, intellectual property law. "Through fees, and through licensing negotiations that give IBM access to other patents.
>
> "The IBM patent portfolio gains us the freedom to do what we need to do through cross-licensing—it gives us access to the inventions of others that are the key to rapid innovation. Access is far more valuable to IBM than the fees it receives from its 9,000 active patents. There's no direct calculation of this value, but it's many times larger than the fee income, perhaps an order of magnitude larger."

This information should dispel the belief that the patent system will "protect" a small software developer from competition from IBM. IBM can always find patents in its collection which the small developer is infringing, and thus obtain a cross-license.

However, the patent system does cause trouble for the smaller companies which, like IBM, need access to patented techniques in order to do useful work in software. Unlike IBM, the smaller companies do not have 9,000 patents and cannot usually get a cross-license. No matter how hard they try, they cannot have enough patents to do this.

Only the elimination of patents from the software field can enable most software developers to continue with their work.

The value IBM gets from cross-licensing is a measure of the amount of harm that the patent system would do to IBM if IBM could not avoid it. IBM's estimate is that the trouble could easily be ten times the good one can expect from one's own patents—even for a company with 9,000 of them.

B.2 User interface copyright

(This section is copied from the GCC manual, by Richard Stallman.)

> *This section is a political message from the League for Programming Freedom to the users of the GNU font utilities. It is included here as an expression of support for the League on my part.*

Apple, Lotus and Xerox are trying to create a new form of legal monopoly: a copyright on a class of user interfaces. These monopolies would cause serious problems for users and developers of computer software and systems.

Until a few years ago, the law seemed clear: no one could restrict others from using a user interface; programmers were free to implement any interface they chose. Imitating interfaces, sometimes with changes, was standard practice in the computer field. The interfaces we know evolved gradually in this way; for example, the Macintosh user interface drew ideas from the Xerox interface, which in turn drew on work done at Stanford and SRI. 1-2-3 imitated VisiCalc, and dBase imitated a database program from JPL.

Most computer companies, and nearly all computer users, were happy with this state of affairs. The companies that are suing say it does not offer “enough incentive” to develop their products, but they must have considered it “enough” when they made their decision to do so. It seems they are not satisfied with the opportunity to continue to compete in the marketplace—not even with a head start.

If Xerox, Lotus, and Apple are permitted to make law through the courts, the precedent will hobble the software industry:

- Gratuitous incompatibilities will burden users. Imagine if each car manufacturer had to arrange the pedals in a different order.
- Software will become and remain more expensive. Users will be “locked in” to proprietary interfaces, for which there is no real competition.
- Large companies have an unfair advantage wherever lawsuits become commonplace. Since they can easily afford to sue, they can intimidate small companies with threats even when they don’t really have a case.
- User interface improvements will come slower, since incremental evolution through creative imitation will no longer be permitted.
- Even Apple, etc., will find it harder to make improvements if they can no longer adapt the

good ideas that others introduce, for fear of weakening their own legal positions. Some users suggest that this stagnation may already have started.

- If you use GNU software, you might find it of some concern that user interface copyright will make it hard for the Free Software Foundation to develop programs compatible with the interfaces that you already know.

B.3 What to do?

(This section is copied from the GCC manual, by Richard Stallman.)

To protect our freedom from lawsuits like these, a group of programmers and users have formed a new grass-roots political organization, the League for Programming Freedom.

The purpose of the League is to oppose new monopolistic practices such as user-interface copyright and software patents; it calls for a return to the legal policies of the recent past, in which these practices were not allowed. The League is not concerned with free software as an issue, and not affiliated with the Free Software Foundation.

The League's membership rolls include John McCarthy, inventor of Lisp, Marvin Minsky, founder of the Artificial Intelligence lab, Guy L. Steele, Jr., author of well-known books on Lisp and C, as well as Richard Stallman, the developer of GNU CC. Please join and add your name to the list. Membership dues in the League are $42 per year for programmers, managers and professionals; $10.50 for students; $21 for others.

The League needs both activist members and members who only pay their dues.

To join, or for more information, phone (617) 492-0023 or write to:

League for Programming Freedom
1 Kendall Square #143
P.O. Box 9171
Cambridge, MA 02139

You can also send electronic mail to `league@prep.ai.mit.edu`.

Here are some suggestions from the League for things you can do to protect your freedom to write programs:

- Don't buy from Xerox, Lotus or Apple. Buy from their competitors or from the defendants they are suing.
- Don't develop software to work with the systems made by these companies.
- Port your existing software to competing systems, so that you encourage users to switch.
- Write letters to company presidents to let them know their conduct is unacceptable.
- Tell your friends and colleagues about this issue and how it threatens to ruin the computer industry.
- Above all, don't work for the look-and-feel plaintiffs, and don't accept contracts from them.
- Write to Congress to explain the importance of this issue.

  House Subcommittee on Intellectual Property
  2137 Rayburn Bldg
  Washington, DC 20515

  Senate Subcommittee on Patents, Trademarks and Copyrights
  United States Senate
  Washington, DC 20510

  (These committees have received lots of mail already; let's give them even more.)

Express your opinion! You can make a difference.

Index

U

V

W

X

Z

Table of Contents