CCLRC / Rutherford Appleton Laboratory
Particle Physics & Astronomy Research Council
Starlink Project
Starlink System Note 71

SSN/71.0

Norman Gray
14 June 1999

Dvi2bitmap -- convert DVI files to bitmap images

User's Guide -- Software Version 0.13

Abstract

This application processes a DVI file produced by TeX, converting each page to a single bitmap. The conversion is done directly, rather than through a chain of intermediate file formats, making the process extremely fast. It can produce output as XBM, XPM, GIF and PNG files.

Document information

Document date	14 June 1999
Last revised	19 August 2005
Version number	0
Distribution ID	0.13
Software version	Software Version 0.13

Contents

1 Introduction

It is sometimes useful to convert the typeset output of TeX into a bitmap image viewable on the web. This is most often the case when TeX or LaTeX are being used to typeset the mathematics in a paper being conveted to HTML. It is possible to do this with a chain of general-purpose tools, for example going from DVI to postscript to PNM files to GIFs, but this is generally slow. For an overview of maths and SGML/HTML, see *Appendix A*.

The tool dvi2bitmap does this processing in a single step, reading the DVI file and font files, and emitting a bitmap. It can, at present, generate XBM, XPM, GIF and, if the relevant library is installed, PNG files.

See *Section 2* for usage instructions, and *Section 5* for installation instructions.

The `dvi2bitmap` application is available for download at `<http://www.astro.gla.ac.uk/users/norman/star/dvi2bitmap/>`.

This document matches version 0.13 of the program (you can see what version you have with the command `dvi2bitmap -V`). This should currently be regarded as beta software.

2 Usage

Synopsis:

```
dvi2bitmap [flags] dvi-file
```

This program is intended to conform to the DVI processing standard.

The `dvi-file` argument is the name of a DVI file to be converted to a bitmap. dvi2bitmap looks for the file both with and without the default extension `.dvi`.

You may also read the DVI file from the standard input by giving the DVI file as `"-"`, thus

```
cat myfile.dvi | dvi2bitmap -
```

is an alternative way of reading the file (rather pointless in this case, but it shows the principle; see *Section 3.2* for further discussion). For more arcane purposes, the DVI file may also be specified as <`osfile`>`dvi-file` (which is entirely equivalent), or <`osfd`>`integer`, where the given integer specifies an open file descriptor; specifying `"-"` as the input file is equivalent to <`osfd`>`0`

The motivation for this program was the need for a helper program to produce small bitmaps for inclusion in web pages. Accordingly, the program's underlying usage model is that one would generate a file of TeX or LaTeX material, convert it to a DVI file using TeX, and convert the result to a collection of bitmap files. The input text will typically be equations, but any other TeX material will work as well. For example, the processor which generates the HTML could spit out a file such as

```
\documentclass{article}
\pagestyle{empty}
\begin{document}
$E=mc^2$
\newpage
% etc...
\end{document}
```

and then this program can scoot through it turning each page into a bitmap. I had thought about some complicated scheme to delimit areas on the page, but realised that since the file being processed would typically be generated on the fly specifically for processing by a tool like this, this wasn't really necessary. See *Section 6* for a script which can help with this.

I hope that the program is (or can be made to be) flexible enough to support other modes of use.

2.1 Options

Various of the options below have common syntax features.

`[keyword-value-list]`
: This indicates a sequence of `keyword=value` pairs, separated by commas. Not all keywords necessarily have a value.

`[boolean]`
: This can be 'yes', 'true' or 'on' indicating *true*; or 'no', 'false' or 'off' indicating *false*.

The options are as follows:

`-c, --crop=[keyword-value-pairs]`
: The `--crop` option allows you to control how the generated bitmaps are cropped before they are written. The keywords are 'left', 'right', 'top', 'bottom' and 'all', and the value in each case is the number of pixels to leave as a margin. If the keywords 'relative' (default) or 'absolute' are present, they refer to all of the keywords following: if the crop is specified as 'relative', then the values specify the number of pixels to leave around the blackened pixels of the text; if the crop is 'absolute', then it specifies the position of the crop (in pixels) from the left or top edge of the 'page'. The specification `----crop=absolute,all=dimen`, which would set all the crops to the same position, is silly, and so is forbidden.

  The conversion from points to pixels takes account of the magnification set in the `--magnification` option, if that's been specified already, but it doesn't notice if that's set after this option, and it takes no account of any magnification in the DVI file.

  See *Section 2.2* for TeX `\special` commands which set this within the TeX file.

  See below for TeX `\special` commands which set this within the TeX file.

`-F, --font-search=[keyword-value-list]`
: Specifies how `dvi2bitmap` is to find the fonts it needs. Keywords are as follows:

  `path[=list]`: use the given list of filesystem paths to search for PK fonts, or enable using the default path, if `=path` is missing. The default path is given by the environment variable `DVI2BITMAP_PK_PATH`. See also the discussion of font searching below.

  `kpathsea`: enable using the `kpathsea` library to find fonts. If the program was not built against the `kpathsea` library, this option has no effect.

  `command[=script]`: enable using the given script to find fonts. If the argument is missing, this enables using the script configured into the program at compile-time. This script is any program which will search the filesystem and produce a single line on output, giving the full path to the specified font. For example, this might be given as `kpsewhich pk %f.%dpk` to run the `kpsewhich` program. The command name is a font-string template, as described elsewhere.

  If the program does not find a font using whichever methods have been enabled then, following the pattern of `dvips` and other DVIware, it writes a file `missfont.log` in the current directory, containing commands which you can use to generate the fonts immediately or later.

  `none`: disable all font-searching. The result is that only the `missfont.log` file is written.

  Each of the keywords can be prefixed by 'no' to turn off the corresponding option -- thus `--font-search=nopath,nokpathsea,nocommand` has the same effect as `--font-search=none`.

`-G, --font-gen=[boolean], --font-gen=command[=script]`
Switch automatic generation of fonts off and on. If `--font-gen=command` is given, then the command specified at compile time is used to generate fonts. If, further, a font-generation script is specified, then it will be used instead of the default. The specified script is a font-string template, as described below. The default for automatic font generation is set at compile time.

`-g, --debug=[spec]`
Switch on debugging. The `[spec]` is a list of letters indicating what to debug, as follows. You may trace DVI file parsing ('d'), PK file parsing ('p'), font rasterdata parsing ('r'), input ('i'), bitmap generation ('b') or the main program ('m'). Adding an extra 'g' increases still further the amount of debugging code. The debugging information may be uninformative or unintelligible; it might even crash the program (mention that to me).

`-h, --height=size; -w, --width=size`
Specify the height and width of the canvas on which the output bitmap is painted. The program tries to make an estimate of this based on information within the DVI file, but it can't efficiently get all the information it needs, so sometimes the estimate is wrong. If you get a warning message like `Warning: p.12: bitmap too big: occupies (1183,1072)...(4134,6255). Requested 4100x6200` then you'll need to specify a bitmap size. The numbers `(1183,1072)...(4134,6255)` are the coordinates of the top-left and bottom-right of the bitmap: in this case `--height=6300 --width=4200` would suffice. At some point, I'd like to make the bitmap 'expandable', obviating the need for these options.

`--help`
Display outline help for the options on stderr, and exit

`-l, --end-page`
See option `--start-page`

`-M, --font-mode=[mode]`
Set the MetaFont mode which is used for generating font files. The default is `ibmvga`. If you set this, you will probably have to set the resolution to a consistent number.

`-m, --magnification=[number]`
The TeX magnification parameter which is used when processing the DVI file. It is a float, with 1.0 corresponding to no magnification (the default). This interacts with the resolution as follows: if you specify a resolution of 100, and a magnification of 2, then `dvi2bitmap` will search for PK files at 200 dpi.

`-n, --nodvi`
Do not actually process the DVI file, but read the DVI pre- and postamble. Useful in conjunction with the `--query` options. If this option is present, then the program returns non-zero if any fonts were missing (see also *Section 2.3*). The `-n` is for brevity and consistency with other tools -- the behaviour can be alternatively specified as `--process=nodvi`

`-o, --output=[filename-pattern]`
Choose the output filename pattern. The value is a 'printf' formatting string, with a single `%d` formatting descriptor, which will be replaced in output filenames with the page number. If there is no such descriptor, the filename 'pattern' is used as-is for the first

filename output, after which the program reverts to the default filename pattern. The filename pattern can be overridden on a per-page basis by a TeX `\special` embedded in the DVI file (see the outputfile special in *Section 2.2*). If there is no file extension, or if it does not match the output type, a suitable file extension will be added.

`--pipe`
Indicates that the dvi-file argument is a non-seekable file, such as a named or unnamed pipe. This is automatically the case if you specify the DVI file as stdin, "-".

`-p, --start-page=num, -l, --end-page=num, -P, --page-range=[spec]`
These select page ranges, using a slight extension of the notation used by `dvips` (and the same option letters, except that `dvips` uses `-pp` instead of `-P`).

The `--start-page=snum` and `--end-page=enum` options take single page numbers; if either of these is given, then the program will process pages from page `snum` to page `enum`, with the defaults being the corresponding extremes. The `[spec]` consists of a comma-separated sequence of page numbers and ranges (`a-b`); only those pages, and the pages falling in those ranges (inclusive of the end pages) are processed. Any of these specifications may be prefixed by either `=` or `:n:`. In the former case, DVI page numbers are used rather than the TeX `\count`*0* register; in the latter case, the program examines the `\count`*n* register rather than the default `\count`*0*

You can specify both of these prefixes one or more times, but you cannot mix the `--start-page` and `--end-page` options with the `--page-range` option. The program will respect only the last `--start-page` and `--end-page` options, but the `--page-range` options are cumulative. There may be no spaces in the `pagelist`. The page numbers may be negative.

Examples:

```
dvi2bitmap --page-range=3,6-10 ...
```

process only the specified pages

```
dvi2bitmap --page-range=:2:1 ...
```

process only pages where `\count2` was 1.

`-Q, --query=[keyword-list]`
Query various things. The available possibilities are as given below. The results of each of the queries is printed on a line by itself, prefixed by a 'Q', the keyword and a space, so that, for example, each of the lines produced by the `--query=missing-fonts` option would start `Qmissing-fonts cmbx10 110 ...`

Some of these options (`--query=missing-fonts` and `--query=missing-fontgen`) are probably most useful with the `-n` or `--process=options` options, to investigate a DVI file before processing. Others (`--query=types` and `--query=paper`) are probably useful only with `--process=options`. The option `--query=bitmaps` is only useful if you do actually generate bitmaps. For consistency (and so you don't have to remember which ones do which), the appropriate `--process` option is *not* implied in any of them, and you have to give it explicitly.

`--query=bitmaps`
: Prints on stdout a line for each bitmap it generates, giving the filename, horizontal size, and vertical size, in pixels. This also reports the position of any 'mark' in the bitmap -- see item '`mark`'.

`-Qf, --query=missing-fonts`
: Show missing fonts. The program writes on standard output one line per missing font, starting with `Qf` or `Qmissing-fonts` (depending on which of the variants was given -- the shorter ones are less mnemonic, but more convenient to parse in scripts), then five fields: the font name, the DPI value it was looking for, the base-DPI of the font, the magnification factor, and a dummy metafont mode. This output might be massaged for use with the mktexpk (TeXLive) or MakeTeXPK (teTeX) scripts to generate the required fonts, but `--query=missing-fontgen` is more straightforward.

`-QF, --query=all-fonts`
: As for `--query=missing-fonts` except that found fonts are also listed, all prefixed by `Qall-fonts`

`-Qg, --query=missing-fontgen`
: As for `--query=missing-fonts`, except that the output consists of the string `Qmissing-fontgen` followed by a `mktexpk` or `MakeTeXPK` command which can be used to generate the font.

`-QG, --query=all-fontgen`
: As for `--query=missing-fonts`, except that font-generation commands for found fonts are also listed, prefixed by `Qall-fontgen.`

  Note: Only one of `--query=missing-fonts`, `--query=all-fonts`, `--query=missing-fontgen` and `--query=all-fontgen` should be specified -- if more than one appears, only the last one is respected -- if more than one appears, only the last one is respected. In each of these four cases, plus their short forms, font-generation is automatically suppressed. This is probably what you want (it's not obvious why you're querying this otherwise), but if you do not want this, then you can reenable font generation with `--font-gen=true`

`--query=paper`
: Show the list of paper sizes which are predefined for the `--paper-size` option.

`--query=types`
: List the output image formats which the program can generate, on stdout, separated by whitespace. The first output format is the default.

`-r, --resolution=[number]`
: Specifies the output resolution, in pixels-per-inch. This is used when deciding which PK files to use. The default is 110, which matches the default `ibmvga` metafont mode.

`-R, --colours=[keyword-value-list], --colors=[keyword-value-pairs]`
: Specifies the foreground or background colours, as RGB triples. The keywords are either `foreground` or `background`, and the values are a triple of integers separated by slashes, for example `--colours=foreground=127/127/255`. The integers must be in the range [0,255], and can be specified in decimal, octal or hex (for example `127=0177=0x7f`), or else the whole spec may be of the form `#rrggbb`, where 'rr', 'gg' and 'bb' are each a pair of hex digits.

`-s, --scaledown=[number]`
Reduces the linear size of the output bitmap by the given factor (default 1).

`-T, --output=type=[type]`
Choose the output format, which can be `png`, `gif`, `xpm` or `xbm`. The program generates XBM bitmaps by default, and has simple support for XPM. The GIF and PNG options may not be available if they weren't selected when the program was configured.

`-t, --paper-size=papersize`
Set the initial size of the bitmap to be one of the paper sizes returned by `--query=paper`. This is useful either to make sure that there is enough room on the initial bitmap, to avoid the warning above, or, along with the `--process=nocrop` option, to force the output bitmap to be a certain size.

`-v, --verbose=[quiet|silent]`
Quiet mode suppresses some chatter, and silent mode suppresses chatter, and does not display warnings or errors either.

`-V, --version`
Display the version number and compilation options, and exit.

`-X, --process=[keyword-value-list]`
Specifies the processing to be done. Keywords are as follows:

`dvi` and `nodvi` : enable or disable processing of the DVI file. If disabled, we do not require a DVI file to be present on the command line. The `nodvi` option is useful with some of the `--query` options.

`postamble` and `nopostamble`: enable or disable processing of the DVI postamble. If dvi2bitmap is called to invoke a non-seekable device such as a pipe, you should disable processing of the postamble. Disabling the postamble processing is incompatible with the `--query` options which examine the fonts in the file. By default, both the DVI body and the postamble are processed.

`--process=options`: shorthand for `--process=nopreamble,nodvi,nopostamble`. Only the options are examined.

`blur` and `noblur`: if true, blurs the bitmap, making a half-hearted attempt to make a low-resolution bitmap look better. This really isn't up to much -- if you have the fonts available, or are prepared to wait for them to be generated, a better way is to use the `--magnification` option to magnify the DVI file, and then the `--scale` option to scale it back down to the correct size.

`transparent` and `notransparent`: if true, this makes the output bitmap have a transparent background, if that's supported by the particular format you choose using option `--output-type`

`crop` and `nocrop`: if true, specifies that you want the output bitmap to be cropped. This is true by default, so you'll most often use the `crop=false` to specify that you do not want the output cropped (for example, if you're using the `--paper-size` option and want the output to stay the specified size).

By default, bitmaps are not blurred, are cropped, and are transparent if possible.

For PNG files, the output bitmap uses a palette plus an alpha channel; these are calculated in such a way that if you display the resulting bitmap on the same colour background as

dvi2bitmap was using (which is white by default, but can be specified using the 'background' special) then the result should look identical to the result with no transparency information, but probably progressively worse the further the background moves from this. I suppose, but can't at present check, that this implies that you should choose a mid-grey background colour when making such transparent PNGs. I'd welcome advice on this point.

2.2 DVI specials

dvi2bitmap recognises several DVI special commands, and emits a warning if it finds any others.

The syntax of the special commands is

```
\special{dvi2bitmap <special-command>+ }
```

There may be one or more <special-command> sequences within a single special.

The <special-command> which the program recognises are:

`default`
: Makes other special-commands in this same special affect defaults. See those commands for details.

`outputfile <filename>`
: The output file used for the current page will be named `filename.gif` (if the output type were 'gif'). A filename extension will be added if none is present, or if it does not match the output type selected.

  If the `default` command has been given, then this instead specifies the default filename pattern, and the 'filename' should contain a single instance of either `%d` or `#`; if there is no such instance, one will be implicitly added at the end.

  The `%d` is precisely analogous to the behaviour of the --output option. However it is actually rather tricky to get an unadorned percent character into a TeX special, unless you play catcode tricks, and this is why you may alternatively include a `#` character to indicate where the page number should go. In fact, since it is *also* rather tricky to get a single `#` character in a special, any immediately following `#` characters are ignored. Thus the recommended way of specifying this special is through something of the form

  ```
  \special{dvi2bitmap default outputfile myfile-#}
  ```

  using the `#` form, and letting the file extension be controlled by the output type which is actually used.

`absolute`
: Affects the `crop` command.

`crop <side> <dimen>`
: Crop the bitmap on the current page so that the specified edge of the bitmap is <dimen> points away from the bounding box of the blackened pixels. <side> may be one of 'left', 'right', 'top', 'bottom' or 'all', referring to the corresponding edge, or all four edges at once.

If the `default` command has been given in this special, then this pattern of cropping is additionally made the default for subsequent pages. If the `absolute` command has been given, then the crop position is set at <`dimen`> points from the appropriate edge of the 'paper'.

The `-c` and `-C` command-line options (*Section 2.1*) have the effect of setting initial defaults. In the absence of either of these, the initial crop is exactly at the bounding box.

`default imageformat <format>`

Set the default image format, which should be one of the keywords 'xbm', 'xpm' 'gif', 'png'. This is equivalent to specifying the image format through the `-t` option (section *Section 2.1*).

The keyword is just `imageformat`, but you must specify the `default` keyword when you specify `imageformat`; this is for consistency, and makes it clear that this is setting a default format rather than setting the format only for the next image (that's not implemented at present, but could be added).

`default foreground|background <red> <green> <blue>`

Sets the (default) foreground and background colours for text. This works, as long as you specify the colour change before any text is output, since you can't, at present, change the colours after that. Specifically, you can't change the colours for a fragment of text in the middle of a page; for this reason, and as with `imageformat` you should at present always include the `default` keyword when using this special. The integers must be in the range [0,255], and can be specified in decimal, octal or hex (ie, `127=0177=0x7f`).

`strut <left> <right> <top> <bottom>`

This places a 'strut' in the generated file. Using the usual TeX `\strut` won't work: that would leave the appropriate space when TeXing the file, but that space doesn't explicitly appear in the DVI file (which is just a bunch of characters and locations), so when `dvi2bitmap` fits its tight bounding box to the blackened pixels in the file, it knows nothing of the extra space you want.

The 'strut' special forces the bounding box to be at least 'left', 'right', 'top' and 'bottom' points away from the position in the file where this special appears. All the dimensions must be positive, and they are floats rather than integers.

If you wanted to set a page containing only the maths '`${}^\circ$`' (why, is another matter), `dvi2bitmap` would normally make a tight bounding box for the bitmap, so that you'd get an image containing only the circle (unless other crop options were in force). If, in this case, you put in a special such as `\special{dvi2bitmap strut 0 2 10 2.5}`, you would force the bounding box to come no closer than 0pt to the left of the position in the file where this special appears, 2pt to the right, 10pt above and 2.5pt below.

A useful bit of TeX magic is:

```
{\catcode`p=12 \catcode`t=12 \gdef\DB@PT#1pt{#1}}
\def\DBstrut{\strut\special{dvi2bitmap strut 0 0
\expandafter\DB@PT\the\ht\strutbox\space\expandafter\DB@PT\the\dp\strutbox}}
```

Once you've done that, the command `\DBstrut` will put an appropriate strut in the output.

`mark`

This sets a 'mark' in the generated file, which is reported when you specify `--query=bitmaps` (see item '`-Q, --query=[keyword-list]`'). Normally, `--query=bitmaps` writes out the horizontal and vertical size of the generated bitmap. If use of this special has placed a 'mark' in the bitmap, however, then the `--query` option also reports the position of that mark, as a position within the bitmap, such that the top-left corner of the bitmap has coordinates *(0,0)*. For example, after

```
\noindent\special{dvi2bitmap mark}Hello
```

the command line

```
dvi2bitmap --query=bitmaps foo
```

might report

```
Qbitmaps foo-page1.png 80 14 -1 10
```

indicating that the bitmap is 80 pixels wide by 14 high, and that the reference point, after cropping, is at position *(-1, 10)*. The '-1' is because the mark appears to the left of the 'H' of 'Hello' (and the 'H' probably has some negative offset), and the '10' indicates that the baseline of this text is 10 pixels from the top of the bitmap; this latter information might be useful when working out how to position this bitmap within a generated HTML file. See *Section 3.4* for a discussion of how to do that.

Both here and in the support for the 'strut' special, there is a great deal of scope for off-by-one errors; also it's unclear what is the best interface to this functionality, so it's possible that this might change in subsequent versions. The author welcomes comments.

`unit <u>`

The units in the 'strut' and 'crop' specials are by default in TeX points. You may switch to a different unit with the 'unit' special. The specifier 'u' gives a unit name, which may be selected from the set of units TeX knows about ('pt', 'bp', 'cm', and so on), plus 'pixels', and 'dvi' to select DVI file units (usually the same as 'sp'). If the 'default' qualifier is present, this setting applies to subsequent special strings as well.

For example, the pair of commands

```
\special{dvi2bitmap default outputfile trial-# unit pc crop all 5}
\special{dvi2bitmap absolute crop left 5}
```

will change the output filename pattern for the rest of the DVI file, and set a 5pc margin round the bounding box. The current page, however, will have a left-hand crop five points in from the left hand side. Remember that TeX's origin is one inch from the left and the top of the paper, and it is with respect to this origin that the program reckons the absolute distances for the cropping.

2.3 Exit value

Exits with a non-zero status if there were any processing errors. Having *no* fonts present counts as a processing error.

If there is at least one font present, then missing fonts will be replaced by the first `cmr10` font it finds, or a more-or-less randomly chosen alternative if that font is not used at all. The program will produce a warning if the `-q` option is not present, but it will return with a zero (success) status.

Exception: If the `-n` option (see *Section 2.1*) is present, then the program returns success only if *all* fonts are present.

2.4 Examples

Basic usage examples.

```
% dvi2bitmap --resolution=110 --magnification=2 --scale=2 \
    --output-type=gif hello.dvi
```

This converts the file `hello.dvi` to a GIF bitmap. It first sets the magnification factor to 2, so that the program uses a double-size font (eg, `.../cmr10.220pk`), then scales the bitmap down by a factor of 2 to obtain a bitmap of the correct size.

```
% dvi2bitmap -n -Qf --resolution=110 --magnification=1.5 \
    --verbose=quiet hello.dvi
Qf cmr10 165 110 1.5 localfont
```

This reads the DVI file to find out what fonts are required, but does not process it further. It then tries to find the fonts, fails, and produces a list of parameters which could be used to generate the font files.

See also *Section 2.5*, and see *Section 3* for more elaborate examples of use.

2.5 Finding and generating fonts

2.5.1 Finding fonts

The program searches for fonts using a number of mechanisms.

1. The `-fp` option (see above) specifies a colon-separated list of filename templates which should be searched for font PK files. If this is given on the command line, it overrides...
2. The `DVI2BITMAP_PK_PATH` environment variable, if defined, specifies a colon-separated list of filename templates which are to be searched for PK files.

3. If the program cannot find fonts using the environment variable, and if it was configured with support for the `kpathsea` library (see *Section 5.1*), then it should find PK files using the same mechanism other DVI processors use.

4. `dvi2bitmap` can be configured to use a script to find fonts. If the program was not configured to use `kpathsea` or the search fails, then the program invokes a script which knows where to find font files, given a search pattern, and which returns a single line containing a discovered font filename. See item '`--with-fontfinder`'.

The third method is the ideal -- you should build `dvi2bitmap` using the `kpathsea` library if possible (see *Section 5.1.1* for how to obtain it): it is because other DVI-processing programs like `dvips` and `xdvi` are built with the `kpathsea` library, that you normally never have to worry about where fonts live. The `kpathsea` library is generally integrated with the font-generation commands, and can be queried using the `kpsewhich` command.

There are one or two possible wrinkles with the third method. The path-searching library is very powerful and flexible, but it is possible to be tripped up by its configuration file.

Firstly, the program has to *find* the configuration file. The program should sort this out for itself at configuration time, but it is possible that you might have to give it some help. If you specify the `TEXMFCNF` environment variable, setting it to the directory which contains your TeX installation's `texmf.cnf` file, then this overrides the program's notion of where the configuration should be. You can find this file using the command `kpsewhich cnf texmf.cnf`.

Secondly, it's possible to break the configuration file. Certain TeX distributions (the ones that came with early RedHat 6.x distributions are ones I know about) are broken in an unfortunate way. See *Section 2.5.2* for a discussion.

2.5.2 Not finding fonts

It can sometimes happen that `dvi2bitmap` fails to find fonts, successfully calls `mktexpk` to build them, but then *still* fails to find them, even though `mktexpk` has put them where they should be. There are (at least) three possible reasons for this.

If you are using the `kpathsea` library, there might be some misconfiguration which is confusing it. You can trace `kpathsea`'s deliberations in massive detail by giving the option `-ggp` (item '`-g, --debug=[spec]`').

Perhaps you do not have the `kpathsea` library installed, or have disabled it, but you *have* requested that font-generation be enabled (see *Section 5.1*). What happens in this case is that `mktexpk` successfully builds the fonts, and installs them in the correct place, where 'correct place' means 'the place where `kpathsea` would find them'; you're not using `kpathsea`, so no fonts for you. What you have to do in this case is work out where the 'correct place' is (`kpsepath` and `kpsewhich` can help here), and specify that place using either the `-fp` option or the `DVI2BITMAP_PK_PATH` variable, as above (this is confusing, I know, but more-or-less unavoidable, since we are here trying to do `kpathsea`'s job, without `kpathsea`).

I think it is also possible to fall victim to a race condition, where the font is built successfully, but the program looks for it in the correct place before the font is fully flushed to disk, or (mumble) something like that. Simply running `dvi2bitmap` a second time seems to work OK. I'm not sure precisely what's going on here, and I'd welcome more precise observations, here.

Another, slightly nastier reason is as follows.

Some `texmf.cnf` files declare the location of the user-writable font directory though a setting like

```
VARTEXFONTS=$SELFAUTOPARENT/var/lib/texmf
```

whereas others have something like

```
VARTEXFONTS=$TEXMFLOCAL/fonts
```

Now, `$SELFAUTOPARENT` is a variable which is set by the kpathsea library to be the grandparent directory of the executable which uses the library. So, for `/usr/bin/{tex,latex,mktexpk,...}`, it's `/`, but if your `dvi2bitmap` binary doesn't live with the other dvi-ware then its `$SELFAUTOPARENT` will be different, so that `dvi2bitmap` will look for fonts in a *different* place from the place where `mktexpk` put them when it successfully generated them.

I would argue fairly strongly that having the `VARTEXFONTS` directory depend on the location of the dvi-ware *executables* is a very silly thing to do. This was the case in the teTeX distribution which came with RedHat 6.0, though this was fixed pretty rapidly. If you've fallen foul of this, then you can either

- change your `texmf.cnf` file to something more like the second example above; or
- install `dvi2bitmap` along with the other TeXware.

I'd much prefer the first alternative, myself.

A third option is to get `dvi2bitmap` to work around the problem, by telling it to claim to be some program which *is* installed along with the other dvi-ware. You do this with the `--enable-fake-progname` option to the configuration script (see item '`--enable-fake-progname`').

2.5.3 Generating fonts by hand

If you didn't enable automatic font-generation, or if you did and something went wrong, you might have to generate fonts by hand. You need to look at the documentation for your TeX system, specifically the `mktexpk` and `MakeTeXPK` scripts (one of which might be just an interface to the other).

See the discussion of the '`make test`' script in *Section 2.5.1*. Also, note that the option `-Qg`, given to `dvi2bitmap`, displays the font-generation commands which would be required to build the fonts missing from the specified DVI file. These are the commands which `dvi2bitmap` would employ to generate these fonts, when automatic-font-generation is enabled.

Since dvi2bitmap's default resolution is 72 dpi, as opposed to the usual printer resolution of 300 or 600 dpi, you are unlikely to have suitable fonts on your system, and will need to generate them. The program will generate these automatically, if it was configured with support for that (see *Section 5.1*); if it wasn't configured with that support, or if the automatic font generation fails, you might need to generate the fonts by hand.

How you generate fonts depends on your TeX distribution. As explained in *Section 2.4*, you can determine which fonts you need using the `-Qf` option. The teTeX and TeXLive TeX distributions include scripts to generate fonts for you; if you have a different distribution, there might be a similar script for you to use, or you might have to do it by hand. In the case of teTeX, the command you'd use in the example above would be:

```
% MakeTeXPK cmr10 165 110 1.5 ibmvga
```

This would generate fonts using the `ibmvga` Metafont mode, using a base resolution of 110 dpi (the default for that mode), at a magnification of 1.5 times, giving a resultant resolution of 165 dpi.

If you're using the TeXLive distribution, the equivalent command would be:

```
% mktexpk --mfmode ibmvga --mag 1.5 --bdpi 110 --dpi 165 cmr10
```

If you want to use the same mode as you use for printing documents, then the mode `localfont` should do the right thing. Otherwise, and probably better if these images are intended for the screen rather than paper, you could use a more specialised mode such as `ibmvga`, which has been tweaked to be readable at small resolutions. See the file `modes.mf` somewhere in your metafont distribution for the list of possibilities.

After you have created the fonts, try giving the command

```
% kpsewhich pk cmr10.165pk
```

to confirm that TeX and friends can find the new fonts, and that your dvi2bitmap environment variable is set correctly. This command is part of the `kpathsea` distribution, rather than the core TeX distribution, so may not be present on your system.

2.5.4 Font-string templates

The search-path and font-finder routes use font-string templates. Here, the components of a font file name, or a font-finding command, are specified using placeholders like `%f`. You may use

`%M`	mode (eg. ibmvga)
`%f`	font name (eg. cmr10)
`%d`	dpi (eg. 330)
`%b`	base dpi (eg. 110)
`%m`	magnification (eg. 3)
`%%`	%

Thus, using these values as an example, if one of the entries in `DVI2BITMAP_PK_PATH` were `</var/tmp/%M/%f.%dpk>`, this would expand into `</var/tmp/ibmvga/cmr10.330pk>`. Alternatively, if we had given the font-finder script as `/usr/local/teTeX/bin/kpsewhich pk %f.%dpk`, the `dvi2bitmap` would have executed the command `.../kpsewhich pk cmr10.330pk`, which would have returned with a suitable font path.

3 Usage notes

3.1 Good choices of fonts and scaling

There is a certain amount of subtlety in choosing fonts and resolutions for maximum readability.

The fonts that `dvi2bitmap` (currently) uses by default are from the `cmr` family, and generated using Metafont mode `ibmvga`, chosen because its design resolution, of 110 pixels to the inch, is approximately right for bitmaps viewed on the screen. This is not, however, necessarily the optimal choice in all circumstances.

You can produce some simple antialiasing by magnifying the output bitmaps then scaling them down, so that:

```
% dvi2bitmap --magnification=2 --scale=2 myfile.dvi
```

doubles the size of the bitmaps, then halfs it, effectively blurring it in the latter stage. This works quite well. You don't *necessarily* get better results with larger factors (though it does, of course, depend on the situation), because Metafont already does some work to make the characters easier to read, and I suspect that excessive antialiasing merely frustrates this.

If you choose a different Metafont mode, it can make a difference. In your TeX distribution, there should be a file called `modes.mf`, containing a large collection of Metafont font-generation modes (look for it using `kpsewhich modes.mf` if you have that command), and there are several modes in this set which have resolutions in the 70 to 200 range, which are therefore about the right size to be useful in this context. You're probably aiming for a resolution of around 100 pixels per inch, if you want the text in the output bitmap to be around the same size as the other text on your monitor. For example, try the `ncd` and `nec` modes:

```
% dvi2bitmap --magnification=2 --scale=2 --font-mode=ncd --resolution=95 try.dvi
% dvi2bitmap --scale=2 --font-mode=nec --resolution=180 try.dvi
% dvi2bitmap --magnification=2 --scale=4 --font-mode=nec --resolution=180 try.dvi
```

Note that the declared resolution must match the font mode -- the default resolution of 110 is designed to match the default mode of `ibmvga`. Also the `nec` mode, because its base resolution is large anyway, only needs to be scaled down to get adequate antialiasing.

It should be possible to create a Metafont mode specifically for `dvi2bitmap` applications. That might be a useful project for someone!

Another thing to look at is whether changing the font family can help. The Computer Modern family is, of course, designed for paper. The Concrete Math family[1], though also designed primarily for paper, has features which make it particularly suitable for this application. The FAQ article which discusses maths font choices[2] remarks that

> Since Concrete is considerably darker than Computer Modern, this variant may be of particular interest for use in low-resolution printing or in display-oriented applications such as posters, transparencies, or online documents.

[1] `http://www.tex.ac.uk/cgi-bin/texfaq2html?label=concrete`
[2] `http://www.tex.ac.uk/cgi-bin/texfaq2html?label=psfchoice`

As well as this, these fonts have rather simpler outlines than Computer Modern *and* they are rather more upright, both of which make them more robust to being rendered at rather small resolutions.

You can use the concrete maths fonts simply by adding the declaration

```
\usepackage{concmath}
```

to the preamble of your LaTeX document. In tests, the best configuration for clarity in a bitmap appeared to be a document using the concrete maths fonts, plus the `dvi2bitmap` invocation

```
% dvi2bitmap --magnification=2 --scale=4 --font-mode=nec --resolution=180 try-cc.dvi
```

Any observations on this topic would be warmly appreciated.

Many thanks to Doug du Boulay at `titech.ac.jp` for raising this issue, for thus prompting me to discuss the various options, and for then doing the critical testing work.

3.2 Using dvi2bitmap in a pipe

`dvi2bitmap` is perfectly happy reading DVI files from a pipe, so that

```
cat myfile.dvi | dvi2bitmap --pipe -
```

will work perfectly well. Since the program knows that the standard input -- indicated by the `-` argument to `dvi2bitmap` -- is not seekable, the option `--pipe` is actually redundant. This is not by itself particularly useful, since TeX is not written to send its DVI output into a pipe.

If, however, you make a 'named pipe' beforehand, using the Unix `mkfifo` command (a FIFO, or first-in-first-out, is the other name for such an object), then TeX can be persuaded to send its output there.

```
% mkfifo myfile.dvi
% ls -l myfile.dvi
prw-r--r--  1 norman  admin  0 Aug 12 00:18 myfile.dvi
% latex myfile >myfile.stdout &
% dvi2bitmap --pipe mkfile.dvi
```

Here, we create the FIFO using the `mkfifo` command; looking at it, we see that the first character on the `ls` line is a `p`, indicating the type of object it is. We start (La)TeX going in the background (achieved by the `&`), putting its chatty output into a file, and it merrily writes into the 'file' `myfile.dvi`. Immediately afterwards (we've shown this on two lines, but the two commands could be run together with only the `&` separating them) we start `dvi2bitmap`, telling it to read from the named pipe. The pipe effectively synchronises the two processes, so that if there is nothing to read, `dvi2bitmap` is briefly suspended, and if the pipe is full, LaTeX is suspended. After this performance, the DVI file, `myfile.dvi` ends up zero size again, and the process can be repeated.

You can go further than this, and use a FIFO for LaTeX's input, too:

```
% mkfifo myfile.tex myfile.dvi
% cat foo.tex >myfile.tex & \
    latex '\input myfile.tex' >myfile.stderr & \
    dvi2bitmap --pipe myfile.dvi
```

(the trailing backslashes indicate that this second set of commands is all on one line). The rather odd form of LaTeX invocation (note the quotes) stops TeX from peeking at the file, looking for the magic `%&` line which can tell it which format to use; since the input is a pipe, it's unseekable, so we must use this form, or else LaTeX fails.

3.3 Processing many bitmaps

If you have a large number of fragments of TeX to process, it is best not to invoke `dvi2bitmap` for each one individually. Also, the program does not (yet) allow you to specify more than one DVI file as input. In this situation, it is best to generate an input TeX or LaTeX file which contains *all* of the text you wish to process, with one fragment per page.

For example, the following can help:

```
\documentclass{article}
\pagestyle{empty}
\special{dvi2bitmap default imageformat png}
\newcommand{\neweq}[1]{\vfil\break\special{dvi2bitmap outputfile #1.png}}
\begin{document}

\neweq{index-html-alpha}
$\alpha$

\neweq{index-html-aprime}
$A'=(0,\alpha,0,0)$

[...]

\end{document}
```

That includes a special to make PNG the default output format, and defines a command sequence, `\neweq` which simultaneously forces a new page and inserts a special to name the output file. A script which generates a file like this and then looks for the resulting bitmaps, with known names of course, can run very efficiently. Combining that with the techniques in *Section 3.2* can work wonders.

3.4 Using marks to position bitmaps

When including small images in an HTML page, it can be difficult to get the images to line up with the rest of the text, because a text with descenders (characters which go below the line such as 'p' or 'g') cause the image to be offset from the text line. You can use the 'mark' facility to tell you how much you need to offset such an image so that it lines up properly with the surrounding text.

For example, consider the following usage:

```
% cat temp.tex
\nopagenumbers
\noindent\special{dvi2bitmap mark}%
Hello, this is dvi2bitmap
\bye
% tex temp.tex
This is TeX, Version 3.14159 (Web2C 7.3.11)
(./temp.tex [1] )
Output written on temp.dvi (1 page, 240 bytes).
Transcript written on temp.log.
```

So, a simple test file with a single line in it, with descenders (the 'p'). The 'mark' special goes at the beginning, following \noindent. This is important, otherwise the point that is marked is the very top-left of the image, not the bottom-left.

The output with `--query=bitmaps` is:

```
% dvi2bitmap --verbose=quiet --query=bitmaps temp
Qbitmaps temp-page1.png 163 14 -1 11
```

...where `verbose=quiet` has been used to turn off the usual chatter. It tells us that the bitmap as a whole is 14 pixels deep, and that the mark, which is on the TeX baseline, remember, is at position (-1,11) relative to an origin at the top-left of the bitmap; or in other words there's 3 pixels of space between the TeX baseline and the bottom of the image. We therefore know that we need to offset the image by three pixels to make it line up properly.

There are several ways to do that, but one way is to use per-element CSS properties, like this:

```
% dvi2bitmap --verbose=quiet --query=bitmaps temp | awk
'{printf "<img src=\"%s\" style=\"margin-bottom: %dpx\"/>\n", $2,
$6-$4}'
<img src="temp-page1.png" style="margin-bottom: -3px"/>
%
```

So that's generated an <img> element which has the correct 'src' attribute (from column 2 of the Qbitmaps line) and the correct offset (column6 - column4). That uses awk, but if you're generating your HTML pages with something like Perl or Python (substantially more sensible than awk...!), say, you could do an analogous thing with the dvi2bitmap output, and put the generated <img> elements in the correct places in the output HTML file.

4 The `libdvi2bitmap` library

The main bulk of the code which implements `dvi2bitmap` is in a library, and the program itself is a rather thin wrapper on top of this. This library is intended to be usable for other applications.

The library is written in C++, and provides an abstraction of DVI files, PK font files, Bitmaps, and various other objects used to build these ones. By default, the build procedure creates both static and dynamic libraries.

The library API is extensively documented in the `doc_libdvi2bitmap/` directory of the distribution.

5 Building and installing dvi2bitmap

dvi2bitmap is packaged in two slightly different ways, one which respects the bundling conventions of the Starlink Project[3], and one which is more usual for network-distributed software.

When you unpack the distribution tarball, you should find at least the files `dvi2bitmap_source.tar`, `Makefile.starlink` and `mk`. If you're on a Starlink machine, you can, and should, build the software the Starlink way, using the `mk` script (see *Section 5.2*); but if you're not, you should unpack the `dvi2bitmap_source.tar` file, and build it the conventional way (see *Section 5.1*). Since the Starlink `mk` script is really just a driver for the Makefile within the tar file, you should get the same results both ways.

5.1 General installation and configuration

The package uses a automake/autoconf/libtool build system. Building should therefore be simple:

```
% ./configure
% make
% make install
```

but see the configuration options below.

By default, both a static and a dynamic library are built. If there is some reason why building the dynamic library fails on your platform, configure using `--disable-shared` and you'll build a static library only.

It's a good idea to run `(cd test;make)` as well. See *Section 2.5.1*.

To install, just copy the executable `dvi2bitmap` wherever you want it to live.

You can customise the program using flags to the `./configure` command:

`--with-kpathsea` **and** `--without-kpathsea`
: If you have the `kpathsea` library (see *Section 2.5.1*) but don't, for some reason, want to use it, then give the configure option `--without-kpathsea`. By default, the configuration enables use of the library if it is installed (that is, if the `kpathsea` include files and library are somewhere the compiler will find them. If `kpathsea` is disabled (by default or by request), then fonts will not be generated by default.

  If you have the `kpathsea` library, but it is not in the standard place, then you can provide an argument to the `--with-kpathsea` option giving the name of a directory below which are directories `include` and `lib`, containing the required `kpathsea` include files and library. If you don't have the `kpathsea` library available, see below (*Section 5.1.1*) for notes on obtaining it.

`--disable-texmfcnf`
: The `kpathsea` library finds its configuration files in two ways, either automatically if it is installed in the same directory as the rest of the TeXware, or using the `TEXMFCNF`

[3] `http://www.starlink.rl.ac.uk`

environment variable. The `dvi2bitmap` program sets the latter variable internally, unless it finds it already set. If this will be inconvenient, you can suppress this behaviour by providing the flag `--disable-texmfcnf`, or equivalently `--enable-texmfcnf=no`.

`--enable-fontgen`
The program can attempt to generate fonts, and will do so using the MetaFont mode `ibmvga`, which has a resolution of 110 dots-per-inch.

You can give an argument to this command, which specifies a command-line which will build and install a required font, and return its path on standard output. This uses the font-string template described in *Section 2.5.4*. You could duplicate the default (the `mktexpk` script if present) with the option

```
--enable-fontgen='<path>mktexpk --dpi %d --bdpi %b --mag %m --mfmode %M %f'
```

The default for this option is 'on' -- the program will attempt to generate fonts. Do note, however, that if the `kpathsea` library is not enabled, then the program will *not* be able to find the fonts it generates, unless you configure it correctly using either `-fp` or `DVI2BITMAP_PK_PATH` (see *Section 2.5.1*).

If you wish to disable this automatic font generation, give the option `--disable-fontgen`. Note that this does not completely disable font generation -- it merely sets the default for font generation to 'off', and it can be switched back on again using the option `-fG`.

If you wish to change the default mode, you can do so using the option item '`--with-fontgen-mode=mode,`

`--with-fontgen-mode=mode,res`
If you wish to change the default parameters for font-generation, you can set both the Metafont mode and resolution using this option. For example, the option `--fontgen-mode=pcprevw,118` will make `pcprevw`, which has a resolution of 118 dpi, the default MetaFont mode. Note that the resolution you specify *must* match the mode: see file `modes.mf` for a list of modes and resolutions (use `kpsewhich mf modes.mf` to find this). You can change the resolution and mode on the fly using the `-fm` and `-r` options to the compiled program (*Section 2.1*).

`--with-fontfinder`
This specifies a command to run to find fonts. It is preferable to use the `kpathsea` library if possible, but if this is difficult, then you can specify a script to run to find fonts. This uses the font-string template described in *Section 2.5.4*. To use the standard `kpsewhich` command, for example, you could give the option

```
--with-fontfinder='/usr/local/teTeX/bin/kpsewhich pk %f.%dpk'
```

`--enable-mktexpk` **and** `--enable-maketexpk`
In the default configuration, the program will generate missing fonts using one of the standard scripts present in most TeX distributions. The configuration process looks first for `mktexpk` then `MakeTeXPK`, and uses whichever it finds first. If you have both scripts but wish to use `MakeTeXPK` for some reason, you will have to give the option `--disable-mktexpk`; if you wish to disable both, you will have to give `--disable-maketexpk` as well. Both options take an optional argument giving the path to an alternative script with the same calling interface.

`--with-png` **(default: enabled)**
: If you give this option, and if the PNG library is installed (needs a version after 0.96), then the program will be compiled with support for PNG bitmaps as an output format. You can obtain the PNG library from the PNG home page[4]. You can disable the use of PNG with the option `--without-png`.

`--enable-gif` **(default: disabled)**
: The program generates only XBM bitmaps by default. If you want it to be able to generate GIFs, then give the configure option `--enable-gif`. The GIF format is the copyright of CompuServe. As far as I understand it, one does not need a licence from CompuServe if one is distributing non-commercial, not-for-profit software, such as this. You probably shouldn't enable GIF support when you build this program unless you're in that category as well. But don't listen to me: there's a much fuller account of the whole sorry business in the Graphics File Formats FAQ[5] (HTML[6]).

`--enable-fake-progname`
: This option enables a workaround which allows `dvi2bitmap` to have the expected behaviour when (a) you do not install `dvi2bitmap` along with the other dvi-ware, *and* (b) your `texmf.cnf` file has `VARTEXFONTS` (or a similar variable) depending on one of the `SELFAUTO...` variables (such a `texmf.cnf` file is probably broken, but that may not be your problem, or within your power to fix). This option makes `dvi2bitmap` claim to be a different DVI-reading program which *is* installed in the standard place. See *Section 2.5.2* for discussion. The configuration script uses the location of the `xdvi` program by default, but you can override this by giving the full path to an alternative as an argument to this option (that is, `--enable-fake-progname=/path/to/dviprog`).

  Since this uses undocumented behaviour of the library ('use the source, Luke!'), you almost certainly shouldn't enable it unless you have to.

`--with-path-seps`
: The default configuration is for Unix, and uses the Unix defaults for filesystem path and search path separators. If you are building it on some other architecture, you can alter the defaults by giving a two-character argument to this option, giving the two separators in order. For example, the arguments appropriate for DOS would be

```
--with-path-seps=';'
```

The `./configure` command without any options is equivalent to `./configure --with-kpathsea --with-png --enable-mktexpk` (meaning that kpathsea and PNG output will be enabled if library support for them is found).

The program builds successfully on (at least):

[4] `http://www.libpng.org/pub/png/`
[5] `ftp://rtfm.mit.edu/pub/usenet/news.answers/graphics/fileformats-faq`
[6] `http://www.oreilly.com/centers/gff/gff-faq/gff-faq1.htm`

Platform	Version	Compiler
powerpc-apple-darwin6.6 (MacOS X, 10.2.6)	0.11b1	g++ 3.1 20020420 (prerelease)
sun-sparc-solaris2.9	0.11b1	CC: Sun WorkShop 6 update 2 C++
alphaev67-dec-osf5.1	0.11b1	Compaq C++ V6.5-014
i686-pc-linux-gnu (RedHat 7.3)	0.11b1	g++ 2.96
i686-pc-linux-gnu (RedHat 7.3)	0.10	gcc 2.96
i686-pc-linux-gnu (RedHat 6.2)	0.10b1	egcs-2.91.66
powerpc-apple-darwin6.4 (MacOSX 10.2)	0.10	g++ 3.1 20020420 (prerelease)
sparc-sun-solaris2.8	0.10b1	egcs-2.91.66
alphaev56-dec-osf5.0	0.10b1	egcs-2.91.66
i686-pc-linux-gnu (RedHat 6.2)	0.9-7p1	egcs-2.91.66
powerpc-unknown-linux-gnu (Mac mklinux DR-0.3?)	0.9	egcs-2.90.25 980302 (egcs-1.0.2 prerele
sparc-sun-solaris2.7	0.9	egcs-2.91.66
sparc-sun-solaris2.7	0.9	gcc 2.8.1
sparc-sun-solaris2.7	0.9-6	WorkShop Compilers 5.0 98/12/15 C-
alpha-dec-osf4.0f	0.9-6	Compaq C++ V6.2-024 for Digital U
i386-pc-solaris2.6	0.9-7p1	gcc v2.8.1/libstdc++ v2.8.1.1

The 'version' column is the last version which was actually tested on that platform/compiler combination. Reports of compilations on other platform/compiler combinations gratefully received.

It should be written in standards-conforming C++, so if it doesn't build then (1) it's not as conformant as I think it is (in which case please tell me), (2) your compiler is not as conformant as you think it is (in which case please don't tell me), or (3) you need to invoke some magic to get the compiler to be conformant (in which case tell me, if there's something I can do in the autoconfigure script).

You can override the C++ compiler the configure script will choose by setting the environment variable `CXX`, either via

```
% CXX=cxx ./configure
```

or

```
% env CXX=cxx ./configure
```

depending on your shell.

Run regression tests with `(cd test;make)` in the build directory. This includes a separate whole-program test which additionally gives advice about setting environment variables. You can run this script separately with the command `(cd test;make pathtest)`

Regression test 6 currently fails to link when using Compaq `cxx`, for some arcane C++ reason I have yet to diagnose. The whole-script test mentioned above works, though.

5.1.1 Obtaining the `kpathsea` library

Not all TeX distributions install the `kpathsea` library, even though they install applications built with it, and the `texmf.cnf` configuration file which controls it.

If the library does not appear to be in your distribution, then you can obtain and build it yourself. The library is distributed as part of the `web2c` (Unix TeX source) distribution, which you can find at `<ftp://ftp.tug.org/tex/web2c.tar.gz>`, or mirrored on CTAN sites (for example at `<http://www.tex.ac.uk>` in directory `<systems/web2c>`).

Take a copy of the library (this is a *big* distribution), and unpack it. Go into the `kpathsea` directory and do the usual '`./configure; make; make install`' routine. With (some?) older distributions of the library this appeared not to work, and you had to go to the parent of the `kpathsea` directory, delete the web2c directory (which is the bulk of the distribution), then configure and build it as usual, ignoring the warnings about the missing main `texmf` tree.

5.2 Starlink nodes

If you are on a Starlink node, then you should be able to use the usual `mk` script. Define the environment variables `INSTALL` and `SYSTEM` as usual. `SYSTEM` can be any one of the Starlink-supported platforms `ix86_Linux`, `alpha_OSF1` or `sun4_Solaris`. Then give the commands

```
% ./mk build
% ./mk install
```

This build configures support for GIFs, plus support for the `kpathsea` library if that library is present on the system (it is not distributed with this package and not present by default on all project machines).

There are some issues involved in creating, and hence in installing, an `export_run` distribution. Part of the configuration of `dvi2bitmap` involves establishing the location, on the build system, of font-building scripts and TeX configuration files. However, the point of the `export_run` distributions is that it should run on systems *other* than the build system. We deal with this by (a) configuring `dvi2bitmap` to use a custom font-building script, which is distributed and installed with the package, and which is simply an interface to whichever of the real font-builders is available on the target system, and (b) configuring it with `--disable-texmfcnf`. The former should be OK as long as one of `mktexpk` and `MakeTeXPK` is in the path after installation. The latter, however will cause a problem if `kpathsea` is enabled (which may or may not be the case for the `export_run` system), since this will probably *fail* unless the environment variable `TEXMFCNF` is defined (see `kpsewhich cnf texmf.cnf`). You can see what features are enabled with the command `dvi2bitmap -V`.

6 Bugs, extras, and further developments

To report bugs, please send to `norman@astro.gla.ac.uk` a brief description of the problem; a minimal TeX file which reproduces it; some indication of the machine you're running on (`uname -a` is good); and the output of `dvi2bitmap -V`, which shows the options you have enabled.

See the `TODO` list in the distribution, for the list of things I at least would like to see added to the code.

Bright ideas, fixes and (especially) implementations, cheerfully received.

In the `<.../extras>` directory of the distribution is a script `img-eqlist.pl`, which transforms a file of LaTeX fragments into a LaTeX file, keeping track of filenames, and avoiding generating duplicate bitmaps for duplicated maths.

7 References and acknowledgements

CTAN, the Comprehensive TeX Archive Network, is *the* repository of TeX and LaTeX documentation. The archive is mirrored in numerous places, but the UK node is at `<http://www.tex.ac.uk>`.

DVItype and PKtoPX are two programs Donald Knuth intended as model DVI and PK file readers, and as containers for the canonical documentation of the DVI and PK file formats. They might be available as part of your TeX distribution, but are also available on CTAN, in `</tex-archive/systems/knuth/texware/dvitype.web>` and `</tex-archive/systems/knuth/pxl/pktopx.web>`.

The *DVI Driver Standard, Level 0* is available on CTAN, in directory `</tex-archive/dviware/driv-standard>`. This incorporates sections of the DVItype documentation. This program claims to conform to this standard: if it doesn't, please let me know.

Thanks for bug reports and other suggestions to Eitan Gurari[7] (heroic tester), Oliver Schurr and Oleg Bartunov (`oleg@sai.msu.su`).

Yamabe Kazuharu (`tako_da@qc4.so-net.ne.jp`) supplied the writer for XPM bitmaps.

[7] `http://www.cis.ohio-state.edu/~gurari/`

A Maths and SGML/HTML

- W3C's maths WG[8]. This covers MathML[9], which is now a W3C Recommendation. The working group also supports the www-math[10] mailing list.
- Maths special topic at the SGML web page[11]
- Also, maths proposal from O'Reilly[12].
- MathML at Concordia[13]

A.1 LaTeX maths within HTML

The *real* issue here (for me at least) is rendering equations within an HTML document. There are several tools available which can do that with different trade-offs. The most popular method is to write the equations in a LaTeX document, process it, and then hoik the equations out of the resulting DVI file somehow (typically using `dvips` and a postscript to gif converter), and display them on the web as gifs. The big disadvantage with this is that you get an awful lot of gifs, and the conversion is rather inefficient.

All this hassle *should* become irrelevant once we get browsers which can render MathML directly.

There are reviews of the problems, and some of the tools, in articles *Maths Typesetting for the Internet*[14], and *Comparative Review of World-Wide-Web Mathematics Renderers*[15].

LaTeX2HTML[16] is the granddaddy of these translators -- it parses the LaTeX using Perl, and spits out HTML, turning maths into gifs. It's very robust by now.

John Walker's textogif[17] is a Perl program which orchestrates the various tools to do the conversion via postscript, once you've generated the DVI file. It works, but it's *terribly* slow, which was the motivation for this program.

TeX4ht[18] (TeX for Hypertext) uses TeX's own parser, but still produces equations as gifs. TeX4ht can also emit MathML from LaTeX. The TeX4ht documentation has a useful collection of resources. There's an alternative location for TeX4ht at TUG[19].

tth: TeX to HTML translator[20] (manual[21]). `tth` translates LaTeX maths directly to HTML, with remarkable success and astonishing speed, and with good failure strategies. It works very sweetly, but (a) requires you to tweak your browser to have it map the symbol font appropriately,

[8] `http://www.w3.org/Math/`
[9] `http://www.w3.org/TR/REC-MathML/`
[10] `http://lists.w3.org/Archives/Public/www-math/`
[11] `http://www.oasis-open.org/cover/topics.html#sgml-math`
[12] `http://www.oreilly.com/people/staff/crism/math/`
[13] `http://indy.cs.concordia.ca/mathml/`
[14] `http://forum.swarthmore.edu/typesetting/index.html`
[15] `http://hutchinson.belmont.ma.us/tth/mmlreview/`
[16] `http://www.tex.ac.uk/tex-archive/support/latex2html/`
[17] `http://www.fourmilab.ch/webtools/textogif/textogif.html`
[18] `http://www.cis.ohio-state.edu/~gurari/TeX4ht/mn.html`
[19] `http://www.tug.org/applications/tex4ht/mn.html`
[20] `http://hutchinson.belmont.ma.us/tth/`
[21] `http://hutchinson.belmont.ma.us/tth/manual.cgi`

and (b) the resulting HTML can't be printed legibly. From the same source is TtHMML[22], which translates (La)TeX to HTML plus MathML.

nDVI[23] is a DVI viewer plugin for Unix Netscape. This addresses the problem at the client end.

A.2 Other approaches to maths

A quite different approach is to use a different markup for maths, possibly requiring specialised client software. These other notations typically use semantic markup -- expressing the structure of the maths. At first sight, this seems preferable to LaTeX's presentational markup, but its weaknesses for authoring are exposed (I feel) when you realise that maths is not as closed and unambiguous a language as computer scientists feel it ought to be. Semantic markup's strength is in interfaces with computer algebra systems, and databases -- Abramowitz and Stegun would be ideal in this form! The major dislocation between the two approaches is what makes conversion from presentational to semantic markup so easy. In passing, I'll note that MathML has both a presentational and a semantic variant.

MINSE[24] uses a server to render maths into gifs on the fly. It seems to work rather nicely, but works with its own semantic maths notation.

There is (was?) a project called Euromath[25], which includes a structured SGML editor. This project included a converter which could transform LaTeX to Euromath SGML[26].

OpenMath[27] might be a successor to Euromath. It's an EC Esprit project which 'proposes to develop standards for the semantically-rich representation of mathematics'.

GELLMU[28] is a LaTeX-like markup language, intended to be easy to convert to SGML. Specifically, it is intended to support maths (and hence conversion to MathML) well.

The following are specifically concerned with maths in SGML, using either MathML or other maths DTD fragments.

WebEQ[29] is a suite of Java programs which implement MathML. It's commercial.

TeXML[30] is a gadget from IBM which converts XML to TeX via a DTD fragment. You transform your XML to an equivalent document marked up in TeXML, which you then separately transform to TeX.

EzMath[31] is a Dave Raggett proposal for producing maths on the web. It uses yet another notation, and converts it to online form using a plugin (no printing, and Windows only, as of April 1999).

[22] `http://hutchinson.belmont.ma.us/tth/mml/`
[23] `http://www.nikhef.nl/~t16/public/ndvi/ndvi_doc.html`
[24] `http://www.lfw.org/math/top.html`
[25] `http://www.dcs.fmph.uniba.sk/~emt/`
[26] `http://www.dcs.fmph.uniba.sk/~emt/EmSystem.html#editor`
[27] `http://www.nag.co.uk/projects/OpenMath/`
[28] `http://www.albany.edu/~hammond/gellmu/`
[29] `http://www.webeq.com/`
[30] `http://www.alphaworks.ibm.com/formula/texml`
[31] `http://www.w3.org/People/Raggett/EzMath`

B TeX dimensions

When producing this program, I became terribly confused about the variety of dimensions which appear in DVI and PK files. Table 1 is a summary of the sizes which appear, for the benefit of anyone else attempting a project like this. The reference [Dn] refers to section '*n*' of the webbed DVItype document and [Pn] to section '*n*' of the PktoPX document (see *Section 7*).

If you feel I have misunderstood something here, or got one of the conversion factors wrong (I hate these!), please correct me.

Context	Description
DVI preamble	`num`, `den`: multiply a 'DVI unit' by num/den to obtain a length in units of 10^{-7}m
	`mag`: DVI units are actually multiplied by (num × mag)/(1000 × den)
DVI font definition	`d`: a design size, in DVI units. The nominal size of the font.
	`s`: a 'fixed point' scale factor, range $2^{27} > s > 0$, scaling `d` (see note).
PK preamble	`ds`: the font design size in units of 2^{-20} points.
	`hppp`, `vppp`: number of pixels per point, times 2^{16} (see note).
Character	`tfmwidth`: the width of the character (see note).
	`w`, `h`: the width and height, in pixels, of the character pixel map.
	`hoff` and `voff`: offset of the pixel map from the reference point.
	`dm`, `dx`, `dy`: the pixel escapements. `dm` in pixels, `dx` and `dy` in pixels times 2^{16} (see note

Table 1: Sizes in TeX

1. `s` scales the design size, so that a font is actually used at (s × mag)/(1000 × d) times its normal size.

2. `hppp` and `vppp` aren't used as sizes, but can be used to check you have the right fonts by comparing resolution, etc..

3. `tfmwidth` is the 'physical' size of a character, and is the only size that TeX uses in its calculations, and which the DVI reader uses when working out how far to move the reference point when it sets a character. This is defined in [P9] to be in units of `FIXes`, where one `FIX` is 2^{-20} times the design size in points. [D37] describes how to multiply these widths by scaling factors without overflowing.

4. The difference between the pixel escapements and `tfmwidth` is that the latter is a resolution-independent shift of the DVI reference point, and the former is the PK file's recommendation of the number of pixels the DVI processor should actually move. The DVI processor keeps track of the two reference points, and readjusts the pixel-based one when rounding errors move it too far from the resolution-independent one. See [D89] and [D91]; also [D40].

A few useful conversions are:

- The design size of a font is a physical length, of ds/2^{20} points. [P12]
- A `FIX` is a physical size, of length (designsize)/2^{20}
- A TFM width is a physical size. The `tfmwidth` parameter is in units of `FIXes`, so that the TFM width is a length of `tfmwidth` fixes, which is equal to tfmwidth/2^{20} × ds/2^{20} points.

- Writing ‘dviu’ for the unit ‘DVI units’, ‘sp’ for the scaled point of 2^{-16} points, ‘px’ for pixels, and $d\mu$ for Knuth’s deci-micron, or 10^{-7}m,

$$\begin{aligned} 1\text{sp} = \frac{1}{2^{20}}\text{pt} &= \frac{25400000}{7227 \times 2^{16}} d\mu \\ 7227\text{pt} &= 254\text{cm} = 25400000 d\mu \\ 1\text{dviu} &= \frac{\text{num} \times \text{resolution}}{\text{den} \times 254000}\text{px} \\ 1\text{dviu} &= \frac{\text{num} \times 7227}{\text{den} \times 25400000}\text{pt} \end{aligned}$$

C Release notes

C.1 Release 0.13b3

No functionality changes

Small bug-fixes, from beta-testers

A couple of configuration fixes, for portability. The configure option `--enable-gif` is now the default.

C.2 Release 0.13b2

There are no significant functionality changes from 0.12-2. However the build system changed significantly when the repository was relocated to sourceforge, and this warrants a new minor version, since the required changes, though they should be leaving the functionality unchanged, are more than just bugfixes.

C.3 Release 0.12-2

Bugfix release.

- The mmap configure code was wrong in that, while it wouldn't attempt to call mmap() on systems without a working mmap ('working' according to the AC_FUNC_MMAP test), it would try to compile it, which is incorrect.
- The problem was reported by a user who was building it under cygwin. The `./configure` output which he forwarded suggested that the system did have `mman.h`, but didn't have a 'working' mmap -- so *it* thinks it's got mmap, but autoconf disagrees (I've no idea what autoconf takes as 'working'). The actual report was an 'invalid cast' error which I've fixed.

C.4 Release 0.12-1

Bugfix release.

- Add configure code to cope with systems which declare mmap but fail to declare `MAP_FAILED` (required by Single Unix and POSIX).

C.5 Release 0.12

New and changed functionality:

- None -- no functionality changes from 0.12b2

C.6 Release 0.12b2

New and changed functionality:

- Add 'unit' special, and fix bugs in the 'mark' and 'strut' specials. This causes a few single-pixel changes in the reported locations of marks and the bounding-boxes coerced by struts.
- Conversions between units ('pt', 'cm', and so on) are generalised, and extra library functionality means you'll get this right more often.
- Refactoring of internal logic handling magnification and unit conversions. This is tremendously error-prone, and difficult to write a regression test for, but it produces apparently correct results as far as I've been able to test it. Be on the lookout for subtle errors.
- Various uninitialised-variable bugs caught.

C.7 Release 0.12b1

New and changed functionality:

- Input files are now mapped (using mmap(2)) if that functionality is available, and if they're determined to be seekable (ie, not a pipe). This should increase the speed of access to large DVI files (but this probably wasn't a huge bottleneck before).
- Modified DviFile::currH and currV so that they now report positions in multiple more useful units. Enhanced library documentation.
- Added dvireport.cc program to report the contents of DVI files (in a different way from the dv2dt suite), which also acts as an example of the use of the library.
- Modify `--output` so that if there is no '%' character in the 'pattern', `dvi2bitmap` takes this to be the name of the one output file; omitting the '%' was previously an error.
- Added 'mark' special; the format of the `--query=bitmaps` output has been consequently slightly enhanced.
- Modified interface to `DviFile::getEvent` so that the returned event should be released by a method call, rather than `delete`.
- Fixed bug which meant DviFile could search for PK files in demented sizes (non-initialised variable...)

C.8 Release 0.11

Bugfixes: correct semantics of DviFile::getEndOfPage, so that skipping the last page now works; and Bitmap::scaleDown, so that creating multiple bitmaps now works again.

Reorder checks in configure, for robustness

C.9 Release 0.11b1

Major developments:

- The program is a wrapper for libdvi2bitmap, a library which abstracts DVI and PK files, Bitmaps, and various other objects supporting these. This is built as both a static and a shareable library.
- Can read DVI files from a non-seekable stream such as a pipe.
- Internal bitmaps are now expandable (no more guessing the output bitmap size)
- Build system redone using automake and libtool
- Released under the GPL.
- A few minor option changes (hopefully the last ones) since the previous version

C.10 Release 0.10b1

`dvi2bitmap` is now GPLed!

Option processing has completely changed, with long options added, and the short options reorganised. The following table highlights important changes, where there is a straightforward mapping.

old	new	Note
`-bh`	`-h, --height`	
`-bw`	`-w, --width`	
`-bp`	`-t, --paper-size`	
`-C, -c`	`-c, --crop`	
`-fp`	`-F, --font-search`	[1]
`-fm`	`-M, --font-mode`	
`-fg, -fG`	`-G, --font-gen`	
`-g`	`-g, --debug`	
`-m`	`-m, --magnification`	
`-n`	`-n, --preamble-only`	[2]
`-l`	`-l, --end-page`	
`-p`	`-p, --start-page`	
`-pp`	`-P, --page-range`	[3]
`-o`	`-o, --output`	
`-P`	`-X, --process`	[4]
`-q`	`-v, --verbose`	
`-Q`	`-Q, --query`	[5]
`-r`	`-r, --resolution`	
`-R`	`-R, --colours`	[4]
`-s`	`-s, --scalefactor`	
`-t`	`-T, --output-type`	
`-V`	`-V, --version`	

Notes:

- 1: this is now more flexible. Old `-fp mypath` is now `--font-search=path=mypath`

- 2: `-n` is (conventional) shorthand for `--process=preamble-only`
- 3: page-selection syntax has changed.
- 4: extra syntax added.
- 5: For consistency, *none* of the query options `--query` now automatially exit.

Font searching can now be done by a script specified either at compile time, or as the argument to the `--font-search=command=...` option. This frees you from dependence on the `kpathsea` library, without being therefore obliged to specify an explicit font path.

Changes in the configure script:

- Configure option `--enable-png` has turned into `--with-png` for consistency with `--with-kpathsea`, since part of this option's purpose is to indicate the possibly non-standard location of the `libpng` library.
- Addition of `--with-fontfinder` option.

C.11 Release 0.9-7p1

Jeff Holt (`jeff.holt@hotsos.com`) noted build problems. Either the RedHat includes or gcc have recently become fussier, which exposed a clash of prototypes within the kpathsea code. Extra configuration code added to resolve this.

C.12 Release 0.9-7

Yamabe Kazuharu (`tako_da@qc4.so-net.ne.jp`) supplied a writer for XPM bitmaps. This is built-in, not using the XPM library (see the XPM home page[32]), so may not support all the XPM functionality.

C.13 Release 0.9-6

Fixed a character-handling bug. Oleg Bartunov (`oleg@sai.msu.su`) identified an error in the handling of some of the opcodes in the DVI file, which meant that only 7-bit fonts were being set correctly (how parochial of me!). He also sent me the fix -- many thanks to him.

Also made a change to the options of the configuration script, turning `--enable-kpathsea` back to `--with-kpathsea`, which is more sensible, in retrospect (I think the processing of this was rather garbled in the previous version).

C.14 Release 0.9-5

Added some functionality, but more importantly corrected two nasty font bugs.

[32] `http://www-sop.inria.fr/koala/lehors/xpm.html`

- Added options `-bp`, to set initial bitmap size based on paper sizes and `-Qp` to list the sizes available. Added option `-PC` to turn off cropping, which is useful if you want output of a certain size.
- Fixed a bug which caused fonts loaded at magnified sizes to be requested at the wrong size, even though, when they were found to be missing, the correct font-generation command was being issued.
- Fixed the longish-standing bug to do with spacing at smaller font sizes: cmr7 and cmr12 now look as their maker intended.
- Corrected a bug which caused page-selection to be done subtly wrongly (font selections were being ignored, and byte values of 140 in DVI data would have caused merry hell).
- configure script: The configuration for PNG now doesn't rely on a libpng version better than 0.96. Font-generation is now enabled by default, despite the potential for confusion, which should be covered in the documentation.
- Added better diagnostics to InputByteStream.cc, so that if it goes wrong, you're given at least a clue why.

C.15 Release 0.9-4

No functionality change. The only difference from release 0.9-4 is to the packaging for Starlink nodes (added a working `export_run` target).

C.16 Release 0.9-3

No significant added functionality to the main program, but:

- Added `-QG` and `-Qg` options (see item '`-Q, --query=[keyword-list]`').
- Added `test` directory, and test script, which aims to give useful advice about setting `DVI2BITMAP_PK_PATH` if that turns out to be necessary.
- More discussing in the documentation about generating fonts, specifically referring folk to the '`make test`' target in the Makefile.
- Now option `-q` turns off warnings, but not errors (was rather inconsistent before), and `-qq` turns off errors, too.

C.17 Release 0.9-2

Bugfix release. The program was crashing on Suns, when built with gcc 2.8.1. I'm not sure I found any underlying problem, but I fixed something, which stopped it crashing for me (Mmmm...).

This release adds a script `img-eqlist.pl`, to transform a file of LaTeX maths fragments into a LaTeX file, keeping track of filenames, and avoiding generating duplicate bitmaps for duplicated maths. See *Section 6*.

C.18 Release 0.9

No big changes, but a sufficiently significant change in the functionality to require a new minor version, rather than just a new release number.

- The `-Q` options (see item '`-Q, --query=[keyword-list]`') now write their output lines (to stdout) prefixed by the option letters, so that the output of `-Qf` now consists of lines starting '`Qf` '. This makes it easier to disentangle the output if several of these options are present -- it was added when option `-Qb` was added -- but means that any scripts parsing the previous version's output would break.
- Options `-Rf` and `-Rb` added (see item '`-R, --colours=[keyword-value-list], --colors=[keyword-va`
to set the foreground and background colours on the command line, rather than just using the foreground special. This is still a bit rudimentary, as it can be done only once, for the whole document.
- Some improvements to the PNG output, so that when transparent PNGs are viewed against their 'natural' background, they look the same as the same image without transparency (ie, I fixed the relationship of the output palette and the alpha channel in this case).

C.19 Release 0.8

Added support for palettised PNG output. This was intended to allow me to support full transparency on PNGs, but the particular case which is appropriate for this seems to be very poorly supported in PNG viewers, so, although I believe the output to be correct, you can't see the benefit in most cases.

That made it easy to support changing the foreground and background colours for output text, so I did that as an encore.

Added the 'strut' special.

C.20 Release 0.7

Version number fiddling: this is still beta software, and as such shouldn't be sent out with a major version number of 1. Also, I'd been updating the 'release number' (after the hyphen) with new releases -- this is incorrect, as this should indicate only bug-fixing or documentation changes, rather than changes in functionality. I've therefore rationalised the version numbering, and changed history to the extent of modifying the notes in this ReleaseNotes section. This shouldn't cause significant confusion, as this package has up to now only been circulated internally.

Added support for PNG graphics, using the libpng library, if it's installed.

The Metafont mode used for building fonts is now configurable at runtime; the default is configurable at configuration time; and default default (!) mode and resolution now properly match, so that you no longer have to give the `-r` option every single time.

The manpage source was brought up to date.

The `-f` option was split into the `-fp`, `-fm`, `-fg` and `-fG` options.

The `--enable-fontgen` flag was added to the configuration script.

Various clarifications to the documentation.

Assorted tidy-ups to get it to satisfy `-Wall` on a wider range of platforms.

C.21 Release 0.6

Added support to allow the program to lie about its name. For a discussion of the need for this, see *Section 2.5.1.*

C.22 Release 0.5

Missing fonts are now generated by the program, and the `TEXMFCNF` variable is now set automatically (unless that behaviour is suppressed at configuration time).

Support for more sophisticated cropping.

C.23 Release 0.4

The `DVI2BITMAP_PK_PATH` environment variable now accepts a colon-separated list of directories to search. The font searching algorithm which uses that variable now does the font-size rounding calculation properly, and will additionally search for fonts within 0.2% of the target size, as required by the DVI Driver Standard.

The kpathsea font searching mechanism is still the preferred way of finding fonts, but since many folk don't have, or don't want to obtain, access to the library, I've made the simple font-searching algorithm marginally more sophisticated, as described above.

C.24 Release 0.3

No significant changes to functionality, but a couple of documentation and packaging improvements.

Change history

Distribution 0.13

Norman Gray , 19 August 2005

Minor portability changes. Sources relocated to sourceforge.

Distribution 0.12

Norman Gray , 20 December 2003

Significant refactoring of library, and addition of functionality.

Distribution 0.10

Norman Gray , 18 May 2003

Enhancement of internal documentation. Added --help option.

Distribution 0.10b1

Norman Gray , 13 February 2003

Major changes to options. Supports external programs to find fonts.

2.1 Options Completely new option specs

Change 2 April 2001

Norman Gray , 2 April 2001

Minor documentation updates.

2.5.1 Finding fonts Included a reference to the RedHat config problem, pointing towards the next section.

2.5.2 Not finding fonts Slightly expanded the reference to the RedHat config problem, including a cross-reference.

5.1 General installation and configuration Corrected the URL for the Graphics File Format FAQ.

Distribution 0.9-7

Norman Gray , 5 February 2001

The program now generates XPM bitmaps, too, and I noted this where appropriate.

Distribution 0.9-6

Norman Gray , 12 January 2001

Took opportunity of a bug-fix release of the software to make miscellaneous minor edits to the documentation.

5.1 General installation and configuration Added ‘version’ column to table, noting which version was actually tested.

A.1 LaTeX maths within HTML Updated nDVI URL

Change 8 November 2000

Norman Gray , 8 November 2000

Minor changes to the configuration options, plus a couple of documentation tidyups.

2.1 Options Mentioned that the -gp option also debugs kpathsea.

5.1 General installation and configuration Changed the disable-kpathsea configuration option to without-kpathsea (see also release notes for this version).

Distribution 0.9-5

Norman Gray , 27 June 2000

Added -bp, -PC, -Qp options, and corrected bugs which caused fonts to be generated at the wrong sizes, and to be spaced incorrectly.

2.1 Options Added -bp, -Qp, -PC

Distribution 0.9-4

Norman Gray , 23 June 2000

Adjustments to distribution -- no functionality change.

5.2 Starlink nodes Added a note discussing the `export_run` distribution bundle, and possible problems with it.

Distribution 0.9-3

Norman Gray , 21 June 2000

Mostly clarifying and amplifying explanations.

2.5.1 Finding fonts Added discussion of the 'make test' script, and clarified (I hope) description of how to set DVI2BITMAP_PK_PATH.

Change 16 June 2000

Norman Gray , 16 June 2000

Explained how to set DVI2BITMAP_PK_PATH automatically.

2.5.1 Finding fonts Added description of setting DVI2BITMAP_PK_PATH automatically.

Distribution 0.9-2

Norman Gray , 12 June 2000

Bugfix release.

Distribution 0.9

Norman Gray , 11 June 2000

Minor functionality changes, but significant interface changes, so new version.

Distribution 0.8

Norman Gray , 8 June 2000

Documented changes to list of specials (foreground and background colours, and strut).

2.2 DVI specials Added foreground and background specials, to set foreground and background colours.

2.2 DVI specials Added strut special.

Distribution 0.7-1

Norman Gray , 12 May 2000

Described problems finding fonts.

2.5.1 Finding fonts Described potentially confusing interaction between enable-kpathsea and enable-fontgen configuration options.

5.1 General installation and configuration Moved description of --enable-fontgen, and new default setting.

6 Bugs, extras, and further developments Added bugreport address

Distribution 0.7

Norman Gray , 5 May 2000

New release. Documentation tidyups. Mentioned new support for PNG graphics.

2.1 Options Clarified the -p, -l and -pp options. Changed -Q into -qq, and -L into -Qf.

2.2 DVI specials Added documentation of imageformat special.

5.1 General installation and configuration Described --enable-fontgen. Rename of kpathsea option to --disable-kpathsea. Addition of --enable-png.

5.1.1 Obtaining the `kpathsea` library Improved instructions on getting the kpathsea library source.

Distribution 0.6

Norman Gray , 18 September 1999

Add --enable-fake-progname.

2.5.1 Finding fonts Mentioned --enable-fake-progname

5.1 General installation and configuration Mentioned --enable-fake-progname

Change 13 September 1999

Norman Gray , 13 September 1999

Added page-selection features, and documented them.

2.1 Options Added documentation of -p, -l, -pp options. Note that the old -l and -L options have changed into the new -L and -LL options.

2.5.1 Finding fonts Added a discussion of the foibles of the texmf.cnf configuration file when it comes to finding fonts.

Distribution 0.5

Norman Gray , 6 September 1999

Assorted changes, to describe significant new functionality.

2.1 Options Added documentation of -c and -C options

2.2 DVI specials Rewritten to cover new special commands, particularly support for cropping.

5.1 General installation and configuration Largely rewrote the configuration instructions, in an attempt to make them clearer.

Distribution 0.4

Norman Gray , 5 July 1999

Altered documentation of font-searching, to match new functionality.

2 Usage Added a description of my usage model, to help make things a little clearer.

2.5 Finding and generating fonts Described new functionality for environment variable.

Change 24 June 1999

Norman Gray , 24 June 1999

Slight restructuring to make it easier to find the instructions on generating fonts. Fixed a couple of typos.

2.5 Finding and generating fonts Renamed this section from vague 'environment', and moved description of font generation from examples.

Change 19 June 1999

Norman Gray , 19 June 1999

Added kpathsea support, and appendix on TeX dimensions.

5.1 General installation and configuration Added instructions on configuring in support for the kpathsea library

Distribution initial

Norman Gray , 14 June 1999

First public release

Version 0

Norman Gray , 14 June 1999

Changes in version 0

Norman Gray , 14 June 1999

Initial version