NAME

afm2tfm − convert Adobe font metrics to TeX font metrics

DESCRIPTION

PostScript fonts are (or should be) accompanied by font metric files such as **Times-Roman.afm,** which describes the characteristics of the font called Times-Roman. To use such fonts with TeX, we need TFM files that contain similar information. This program does that conversion. For more information, print out the dvips manual.

FILES

psfonts.map in the config directory used by dvips (update this).
../afm/Makefile in the dvips sources (look at this for examples).

SEE ALSO

dvips(1), tex(1), vptovf(1)

BUGS

An option to add letterspacing to the virtual font was rejected by Knuth because it would make the fi and fl ligatures break the normal rhythmic pattern.

AUTHORS

Tomas Rokicki <rokicki@cs.stanford.edu> and Donald E. Knuth

NAME

amslatex − structured text formatting and typesetting

SYNOPSIS

amslatex [first line]

DESCRIPTION

AMSLaTeX is an extension of LaTeX (see LaTeX − *A Document Preparation System*) that provides the powerful mathematical typesetting macros of AMSTeX (see *The Joy of TEX*) within the syntax of LaTeX. It should be contrasted with LAMS-TeX (**lamstex**(1)), which extends AMSTeX with LaTeX-like features.

AMSLaTeX documents can be processed by **latex**(1), so a separate executable program named **amslatex** is not usually provided.

AMSLaTeX is a TeX macro package, not a modification to the TeX source program, so all the capabilities described in **tex**(1) are present.

The AMSLaTeX macros encourage writers to think about the content of their documents, rather than the form. The ideal, not always realized, is to have no formatting commands (like ‘‘switch to italic’’ or ‘‘skip 2 picas’’) in the document at all; instead, everything is done by specific markup instructions: ‘‘emphasize’’, ‘‘start a section’’.

AMSTeX is the official typesetting system of the American Mathematical Society, and nearly all of its publications are typeset using AMSTeX. The online version of AMS Math Reviews uses AMSTeX input syntax for display of mathematical material. AMS authors can provide editors with computer-readable AMSTeX or AMSLaTeX files.

AMSLaTeX will become deprecated in favor of LaTeX 2e, when the latter is released.

FILES

/usr/local/lib/texmf/tex/ams/amslatex/doc/amslatex.tex
: Documentation for AMSLaTeX, typesettable by LaTeX. This supersedes *The Joy of TEX*, but AMSLaTeX users may still find that book a convenient supplement.

/usr/local/lib/texmf/tex/ams/amslatex/doc/testart.tex
: Sample document illustrating the *amsart* document style.

/usr/local/lib/texmf/tex/ams/amslatex/doc/testbook.tex
: Sample document illustrating the *amsbook* document style.

SEE ALSO

amstex(1), **lamstex**(1), **latex**(1), **tex**(1), **slitex**(1).
Leslie Lamport, LaTeX − *A Document Preparation System*, Addison-Wesley, 1985, ISBN 0-201-15790-X.
Michael Spivak, *The Joy of TEX*, 2nd edition, Addison-Wesley, 1990, ISBN 0-8218-2997-1.

NAME

amstex − structured text formatting and typesetting

SYNOPSIS

amstex [first line]

DESCRIPTION

The AMSTEX language is described in the book *The Joy of TEX*. AMSTEX is a TEX macro package, not a modification to the TEX source program, so all the capabilities described in **tex**(1) are present.

The AMSTEX macros encourage writers to think about the content of their documents, rather than the form. The ideal, not always realized, is to have no formatting commands (like ‘‘switch to italic’’ or ‘‘skip 2 picas’’) in the document at all; instead, everything is done by specific markup instructions: ‘‘emphasize’’, ‘‘start a section’’.

AMSTEX is the official typesetting system of the American Mathematical Society, and nearly all of its publications are typeset using AMSTEX. The online version of AMS Math Reviews uses AMSTEX input syntax for display of mathematical material. AMS authors can provide editors with computer-readable AMSTEX files.

For authors more familiar with LATEX, an AMS-supported variant called AMSLATEX is available.

SEE ALSO

amslatex(1), **latex**(1), **tex**(1), **slitex**(1).
Michael Spivak, *The Joy of TEX*, 2nd edition, Addison-Wesley, 1990, ISBN 0-8218-2997-1.

NAME

bibtex − make a bibliography for (La)TeX

SYNOPSIS

bibtex [**-min-crossrefs=***number*] [**-verbose**] [*auxname*]

DESCRIPTION

BIBTEX reads the top-level auxiliary (*.aux*) file that was output during the running of **latex**(1) or **tex**(1) and creates a bibliography (*.bbl*) file that will be incorporated into the document on subsequent runs of LATEX or TEX. The *auxname* on the command line must be given without the *.aux* extension. If you don't give the *auxname*, the program prompts you for it.

BIBTEX looks up, in bibliographic database (*.bib*) files specified by the \bibliography command, the entries specified by the \cite and \nocite commands in the LATEX or TEX source file. It formats the information from those entries according to instructions in a bibliography style (*.bst*) file (specified by the \bibliographystyle command, and it outputs the results to the *.bbl* file.

The LATEX manual explains what a LATEX source file must contain to work with BIBTEX. Appendix B of the manual describes the format of the *.bib* files. The 'BIBTEXing' document describes extensions and details of this format, and it gives other useful hints for using BIBTEX.

OPTIONS

The **-min-crossrefs** option defines the minimum number of **crossref** required for automatic inclusion of the crossref'd entry on the citation list; the default is two. Without the **-verbose** option, BibTeX operates silently. With it, a banner and progress reports are printed on *stdout*.

ENVIRONMENT

BIBTEX searches the directories in the path defined by the BSTINPUTS environment variable for *.bst* files. If BSTINPUTS is not set, it uses TEXINPUTS; and if TEXINPUTS is not set, it uses the system default:
.:/usr/local/lib/texmf/bibtex/bst
For *.bib* files, it uses the BIBINPUTS environment variable if that is set, otherwise the default:
.:/usr/local/lib/texmf/bibtex/bib
See **tex**(1) for the details of the searching.

If the environment variable TEXMFOUTPUT is set, BIBTEX attempts to put its output files in it, if they cannot be put in the current directory. Again, see **tex**(1). No special searching is done for the *.aux* file.

FILES

**.bst*	Bibliography style files.
btxdoc.tex	"BIBTEXing" − LATEXable documentation for general BIBTEX users
btxhak.tex	"Designing BIBTEX Styles" − LATEXable documentation for style designers
btxdoc.bib	database file for those two documents
xampl.bib	database file giving examples of all standard entry types
btxbst.doc	template file and documentation for the standard styles

All those files should be available somewhere on your system.

The host math.utah.edu has a vast collection of *.bib* files available for anonymous ftp, including references for all the standard TEX books and a complete bibliography for TUGboat.

SEE ALSO

latex(1), **tex**(1).
Leslie Lamport, LATEX − *A Document Preparation System*, Addison-Wesley, 1985, ISBN 0-201-15790-X.

AUTHOR

Oren Patashnik, Stanford University. This man page describes the web2c version of BIBTEX. Other ports of BIBTEX, such as Donald Knuth's version using the Sun Pascal compiler, do not have the same path searching implementation, or the command-line options.

NAME

dvi2tty − preview a TeX DVI−file on an ordinary ascii terminal

SYNOPSIS

dvi2tty [options] dvi−file

DESCRIPTION

dvi2tty converts a TeX DVI−file to a format that is appropriate for terminals and line printers. The program is intended to be used for preliminary proofreading of TeX-ed documents. By default the output is directed to the terminal, possibly through a pager (depending on how the program was installed), but it can be directed to a file or a pipe.

The output leaves much to be desired, but is still useful if you want to avoid walking to the laser printer (or whatever) for each iteration of your document.
Since *dvi2tty* produces output for terminals and line printers the representation of documents is naturally quite primitive. In principle Font Changes are totally ignored, but *dvi2tty* recognizes a few mathematical and special symbols that can be be displayed on an ordinary ascii terminal, such as the '+' and '-' symbol.

If the width of the output text requires more columns than fits in one line (c.f. the −w option) it is broken into several lines by *dvi2tty* although they will be printed as one line on regular TeX output devices (e.g. laser printers). To show that a broken line is really just one logical line an asterisk (“*”) in the last position means that the logical line is continued on the next physical line output by *dvi2tty.* Such a continuation line is started with a a space and an asterisk in the first two columns.

Options may be specified in the environment variable DVI2TTY. Any option on the command line, conflicting with one in the environment, will override the one from the environment.

Options:

−o file Write output to file “file”.

−p list Print the pages chosen by list. Numbers refer to TeX−page numbers (known as \count0). An example of format for list is “1,3:6,8” to choose pages 1, 3 through 6 and 8. Negative numbers can be used exactly as in TeX, e g −1 comes before −4 as in “−p−1:−4,17”.

−P list Like −p except that page numbers refer to the sequential ordering of the pages in the dvi−file. Negative numbers don't make a lot of sense here...

−w n Specify terminal width *n.* Legal range 16−132. Default is 80. If your terminal has the ability to display in 132 columns it might be a good idea to use −w132 and toggle the terminal into this mode as output will probably look somewhat better.

−v Specify height of lines. Default value 450000. Allows to adjust linespacing.

−q Don't pipe the output through a pager. This may be the default on some systems (depending on the whims of the person installing the program).

−e n This option can be used to influence the spacing between words. With a negative value the number of spaces between words becomes less, with a positive value it becomes more. −e−11 seems to worked well.

−f Pipe through a pager, $PAGER if defined, or whatever the installer of the program compiled in (often “more”). This may be the default, but it is still okay to redirect output with “>”, the pager will not be used if output is not going to a terminal.

−F Specify the pager program to be used. This overrides the $PAGER and the default pager.

−Fprog

Use “prog” as program to pipe output into. Can be used to choose an alternate pager (e g “-Fless”).

−t \tt fonts were used (instead of cm) to produce dvi file. (screen.sty is a powerfull mean to do that with LaTeX).

−a Dvi2tty normally tries to output accented characters. With the -a option, accented characters are output without the accent sign.

−l Mark page breaks with the two-character sequence ‘‘^L’’. The default is to mark them with a form-feed character.

−u Don’t make any attempts to find special Scandinavian characters. If such characters are in the text they will map to ‘‘a’’ and ‘‘o’’. This is probably the default outside of Scandinavia. (The installer makes this decision when the program is installed.)

−s Try to find the special Scandinavian characters that on most (?) terminals in Scandinavia are mapped to ‘‘{|}[\]’’. This can be the default, and output from files not containing these special characters will be identical regardless of this option.

FILES

/usr/ucb/more probably the default pager.

ENVIRONMENT

PAGER the pager to use.
DVI2TTY can be set to hold command-line options.

SEE ALSO

TeX, dvi2ps

AUTHOR

Svante Lindahl, Royal Institute of Technology, Stockholm
Improved C version: Marcel Mol
{seismo, mcvax}!enea!ttds!zap
marcel@duteca.et.tudelft.nl

BUGS

Blanks between words get lost quite easy. This is less likely if you are using a wider output than the default 80.

Only one file may be specified on the command line.

NAME

dviconcat − concatenate DVI files

SYNOPSIS

dviconcat [**−m** *magnification*] [**−o** *outfile*] *list of files*

DESCRIPTION

Dviconcat concatenates the named *file*s (or standard input, if no file names are given) onto the standard output or the given *outfile*, creating a new DVI file usable by any of the TeX conversion programs, or even by *dviconcat* itself.

The magnification of the output file is the same as that of the first input file unless the **−m** option is given. When *dviconcat* changes the magnification of any input file, it prints a warning to this effect.

AUTHOR

Chris Torek, University of Maryland

SEE ALSO

dviselect(1), latex(1), tex(1)
MC-TeX User's Guide

BUGS

The program should be called *dvicat*, but that name was taken (for the driver for the C/A/T typesetter).

NAME

dvidoc – display TEX DVI files on character devices

SYNOPSIS

dvidoc [**-e**] [**-sstartpage**] [**-mmaxpages**] dvifile

DESCRIPTION

Dvidoc attempts to display the contents of a DVI file on a character device, e.g. a tty. The output will not be an exact representation of the DVI file since TEX output is designed for high-resolution devices. Math mode and tables are not handled well. Also, line and page breaks will be different when output to a high-resolution device.

The *-e* flag produces an error report on *file.err* where file is the basename of the DVI file.

The *-s* flag allows a starting page to be specified in the form \count0[.\count1[.\count2]]. * is a wildcard. Remember to escape metacharacters.

The *-m* flag allows the number of pages to output to be specified.

The suffix *.dvi* is optional in the file spec.

To use *dvidoc* you must *input* the macro file *docmac.tex* at the beginning of your TEX document or include the style option *dvidoc* in the \documentstyle line of your LATEX document. This file redefines the width of the roman, italic, boldface and typewriter fonts as well as some key parameters. When you wish to output the final document to the laser printer or other high resolution device you should remove the includes and reprocess your document.

If you want to get eight-bits characters (Iso-latin-1 encoding), you have to replace the above files with *dvidoc8* or *docmac8.tex*

FILES

/usr/lib/tex/macros/docmac.tex Input macro package for TEX

/usr/lib/tex/macros/dvidoc.sty Style option for LATEX

SEE ALSO

tex(1), latex(1)

AUTHORS

The eight-bits management was added by Eric Picheral, CICB (picheral@cicb.fr)

NAME

dvidvi − converts a dvi file into another dvi file, with perhaps certain changes.

SYNOPSIS

dvidvi [*options*] *infile*[.dvi] *outfile*[.dvi]

DESCRIPTION

Dvidvi

The suffix *.dvi* is optional in the files spec.

OPTIONS

−f *n* page n is first page selected

−l *n* page n is last page selected

−n *n* select at most n pages. Notice that n is the number of pages selected, independently of the number of pages actually contained in a sheet

−i {**n1..n2**|**n1**}[,...]
include pages (ranges allowed). When this option is used, ONLY the specified pages are selected. However, we can exclude from these pages with the option −x

−x {*n1..n2*|*n1*}[,...]
exclude pages (ranges allowed)

−q work in quiet mode, that is do not print in the screen messages of how the work is being done.

−r reverse the order of the pages.

−m *n* see below, under *Page Layout and specifications*

PAGE NUMBERS

The page numbers for the above options −f −l −i and −x can be specified in different ways.

1) If a number n is given, it is interpreted as the n'th page from the begining of the .dvi file. Of course, this number is independent of the page number assigned by TEX.

2) TEX page numbers are those who are actually written in the page ; these page numbers can be modified, for example, by using the TEX commands \pagenumbering, \setcounter{page}{n}, and \addtocounter{page}{n}. A TEX page number can be specified by preceding the number n with the character @. Thus, if you specify **−f @25 −l @30** you select the pages between 25 and 30, these numbers being those assigned by TEX.

3) However, several pages can have the same TEX page number in a .dvi file. For example, the introductory pages in a book are numbered i, ii, and so on until the first chapter begins and then, the pages are numbered 1, 2, etc. In this case, the pages numbered i and 1 in the .dvi file have the same TEX page number. If you want to select for example the second occurrence of the page numbered 1, you can specify a page number as (@2)1. Thus @1 is equivalent to (@1)1. For example, if you specify
−f (@2)1 −l(@2)10 you select the pages between 1 and 10 of the first chapter, not the introductory pages between i and x.

PAGE LAYOUT AND SPECIFICATIONS

There is another parameter that tells *dvidvi* how you want to change page layout and specifications. This is the **−m** parameter.

The number preceding the colon is the modulo value. Everything will be done in chunks of pages this big. If there is no colon, than the default value is assumed to be one. The last chunk of pages is padded with as many blank pages as necessary.

The number preceding the colon is the modulo value. Everything will be done in chunks of pages this big. If there is no colon, than the default value is assumed to be one. The last chunk of pages is padded with as many blank pages as necessary.

Following the colon is a comma-separated list of page numbers. These page numbers are with respect to the current chunk of pages, and must lie in the range zero to the modulo value less one. If a negative sign precedes the number, then the page is taken from the mirror chunk; if there are m chunks, then the mirror chunk of chunk n is the chunk numbered m-n-1. Put simply, it is the chunk numbered the same, only from the end. This can be used to reverse pages. If no number is given, the page number defaults to 1.

Following each page number is an optional offset value in parenthesis, which consists of a pair of comma-separated dimensions. Each dimension is a decimal number with an optional unit of measure. The default unit of measure is inches, or the last unit of measure used. All units are in true dimensions. Allowable units of measure are the same that TEX allows: in, mm, cm, pt, pc, dd, and cc.

EXAMPLES

−m − Reverses the order of the pages. This time, both the modulo and the page number are defaulted.

−m 2:0 Selects the first, third, fifth, etc. pages from the file. Print this one after printing the next, taking the paper out of the feed tray and reinserting it into the paper feed.

−m 2:−1
Selects the second, fourth, etc. pages, and writes them in reverse order.

−m 4:−1,2(4.25in,0in)

−m 4:−3,0(4.25in,0in)
Useful for printing a little booklet, four pages to a sheet, double-sided, for stapling in the middle. Print the first one, put the stack back into the printer upside down, and print the second. The ‘in’ specifications are superfluous.

−m ,(1pt,1)
Scare your system administrator! Actually, things are so blurry with this option, you may want to send enemies letters printed like this. *Long* letters.

−m 4:0(5.5in,4.25),3(0,4.25)

−m 4:1(0in,4.25),2(5.5,4.25)
Print a four-page card on one sheet. Print the first, rotate the paper 180 degrees and feed it again. (PostScript people can do funny tricks with PostScript so this isn’t necessary.)

SEE ALSO

tex(1), latex(1)

AUTHORS

Tomas Rokicki <rokicki@cs.stanford.edu> and Esteban Zimanyi <ezimanyi@rc1.vub.ac.be>

NAME

dvips − convert a TeX DVI file to PostScript

SYNOPSIS

dvips [*options*] *file*[.dvi]

DESCRIPTION

The program **dvips** takes a DVI file *file*[.dvi] produced by TeX (or by some other processor such as GFtoDVI) and converts it to PostScript, normally sending the result directly to the laserprinter. The DVI file may be specified without the *.dvi* extension. Fonts used may either be resident in the printer or defined as bitmaps in PK files, or a 'virtual' combination of both. If the **MakeTeXPK** program is installed, **dvips** will automatically invoke METAFONT to generate fonts that don't already exist.

For more information, see the Texinfo manual *dvips.texi*, which should be installed somewhere on your system.

OPTIONS

-a Conserve memory by making three passes over the .dvi file instead of two and only loading those characters actually used. Generally only useful on machines with a very limited amount of memory, like some PCs.

-A Print only odd pages (TeX pages, not sequence pages).

-b num Generate *num* copies of each page, but duplicating the page body rather than using the *#numcopies* option. This can be useful in conjunction with a header file setting *\bop-hook* to do color separations or other neat tricks.

-B Print only even pages (TeX pages, not sequence pages).

-c num Generate *num* copies of every page. Default is 1. (For collated copies, see the *-C* option below.)

-C num

Create *num* copies, but collated (by replicating the data in the PostScript file). Slower than the *-c* option, but easier on the hands, and faster than resubmitting the same PostScript file multiple times.

-d num Set the debug flags. This is intended only for emergencies or for unusual fact-finding expeditions; it will work only if *dvips* has been compiled with the DEBUG option. For more information on possible values see section 15 of *dvips.tex.*

-D num

Set the resolution in dpi (dots per inch) to *num.* This affects the choice of bitmap fonts that are loaded and also the positioning of letters in resident PostScript fonts. Must be between 10 and 10000. This affects both the horizontal and vertical resolution. If a high resolution (something greater than 400 dpi, say) is selected, the *-Z* flag should probably also be used.

-e num Make sure that each character is placed at most this many pixels from its 'true' resolution-independent position on the page. The default value of this parameter is resolution dependent. Allowing individual characters to 'drift' from their correctly rounded positions by a few pixels, while regaining the true position at the beginning of each new word, improves the spacing of letters in words.

-E makes *dvips* attempt to generate an EPSF file with a tight bounding box. This only works on one-page files, and it only looks at marks made by characters and rules, not by any included graphics. In addition, it gets the glyph metrics from the *tfm* file, so characters that lie outside their enclosing *tfm* box may confuse it. In addition, the bounding box might be a bit too loose if the character glyph has significant left or right side bearings. Nonetheless, this option works well for creating small EPSF files for equations or tables or the like. (Note, of course, that *dvips* output is resolution dependent and thus does not make very good EPSF files, especially if the images are to be scaled; use these EPSF files with a great deal of care.)

-f Read the *.dvi* file from standard input and write the PostScript to standard output. The standard input must be seekable, so it cannot be a pipe. If you must use a pipe, write a shell script that copies the pipe output to a temporary file and then points *dvips* at this file. This option also disables the automatic reading of the *PRINTER* environment variable, and turns off the automatic sending of control D if it was turned on with the *-F* option or in the configuration file; use *-F* after this option if you want both.

-F Causes Control-D (ASCII code 4) to be appended as the very last character of the PostScript file. This is useful when *dvips* is driving the printer directly instead of working through a spooler, as is common on extremely small systems. NOTE! DO NOT USE THIS OPTION!

-h name
Prepend file *name* as an additional header file. (However, if the name is simply '-' suppress all header files from the output.) This header file gets added to the PostScript *userdict.*

-i Make each section be a separate file. Under certain circumstances, *dvips* will split the document up into 'sections' to be processed independently; this is most often done for memory reasons. Using this option tells *dvips* to place each section into a separate file; the new file names are created replacing the suffix of the supplied output file name by a three-digit sequence number. This option is most often used in conjunction with the *-S* option which sets the maximum section length in pages. For instance, some phototypesetters cannot print more than ten or so consecutive pages before running out of steam; these options can be used to automatically split a book into ten-page sections, each to its own file.

-k Print crop marks. This option increases the paper size (which should be specified, either with a paper size special or with the *-T* option) by a half inch in each dimension. It translates each page by a quarter inch and draws cross-style crop marks. It is mostly useful with typesetters that can set the page size automatically.

-K This option causes comments in included PostScript graphics, font files, and headers to be removed. This is sometimes necessary to get around bugs in spoolers or PostScript post-processing programs. Specifically, the %%Page comments, when left in, often cause difficulties. Use of this flag can cause some included graphics to fail, since the PostScript header macros from some software packages read portions of the input stream line by line, searching for a particular comment. This option has been turned off by default because PostScript previewers and spoolers have been getting better.

-l num The last page printed will be the first one numbered *num* Default is the last page in the document. If the *num* is prefixed by an equals sign, then it (and any argument to the *-p* option) is treated as a sequence number, rather than a value to compare with *\count0* values. Thus, using *-l =9* will end with the ninth page of the document, no matter what the pages are actually numbered.

-m Specify manual feed for printer.

-M Turns off the automatic font generation facility. If any fonts are missing, commands to generate the fonts are appended to the file *missfont.log* in the current directory; this file can then be executed and deleted to create the missing fonts.

-n num At most *num* pages will be printed. Default is 100000.

-N Turns off structured comments; this might be necessary on some systems that try to interpret PostScript comments in weird ways, or on some PostScript printers. Old versions of TranScript in particular cannot handle modern Encapsulated PostScript.

-o name
The output will be sent to file *name* If no file name is given, the default name is *file.ps* where the .dvi file was called *file.dvi;* if this option isn't given, any default in the configuration file is used. If the first character of the supplied output file name is an exclamation mark, then the remainder will be used as an argument to *popen;* thus, specifying *!lpr* as the output file will automatically queue the file for printing. This option also disables the automatic reading of the *PRINTER* environment variable, and turns off the automatic sending of control D if it was turned on with the *-F* option or in the configuration file; use *-F* after this option if you want both.

-O offset
Move the origin by a certain amount. The *offset* is a comma-separated pair of dimensions, such as *.1in,-.3cm* (in the same syntax used in the *papersize* special). The origin of the page is shifted from the default position (of one inch down, one inch to the right from the upper left corner of the paper) by this amount.

-p num The first page printed will be the first one numbered *num.* Default is the first page in the document. If the *num* is prefixed by an equals sign, then it (and any argument to the *-l* option) is treated as a sequence number, rather than a value to compare with *\count0* values. Thus, using *-p =3* will start with the third page of the document, no matter what the pages are actually numbered.

-pp pagelist
A comma-separated list of pages and ranges (a-b) may be given, which will be interpreted as *\count0* values. Pages not specified will not be printed. Multiple **-pp** options may be specified or all pages and page ranges can be specified with one **-pp** option.

-P printername
Sets up the output for the appropriate printer. This is implemented by reading in *config.printername* , which can then set the output pipe (as in, *!lpr -Pprintername* as well as the font paths and any other *config.ps* defaults for that printer only. Note that *config.ps* is read before *config.printername* In addition, another file called *~/.dvipsrc* is searched for immediately after *config.ps;* this file is intended for user defaults. If no *-P* command is given, the environment variable *PRINTER* is checked. If that variable exists, and a corresponding configuration file exists, that configuration file is read in.

-q Run in quiet mode. Don't chatter about pages converted, etc.; report nothing but errors to standard error.

-r Stack pages in reverse order. Normally, page 1 will be printed first.

-s Causes the entire global output to be enclosed in a save/restore pair. This causes the file to not be truly conformant, and is thus not recommended, but is useful if you are driving the printer directly and don't care too much about the portability of the output.

-S num Set the maximum number of pages in each 'section'. This option is most commonly used with the *-i* option; see that documentation above for more information.

-t papertype
This sets the paper type to *papertype.* The *papertype should be defined in one of the configuration files, along with* the appropriate code to select it. (Currently known types include *letter, legal, ledger, a4, a3,*) You can also specify *-t landscape,* which rotates a document by 90 degrees. To rotate a document whose size is not letter, you can use the *-t* option twice, once for the page size, and once for landscape. The upper left corner of each page in the *.dvi* file is placed one inch from the left and one inch from the top. Use of this option is highly dependent on the configuration file. Note that executing the *letter* or *a4* or other PostScript operators cause the document to be nonconforming and can cause it not to print on certain printers, so the paper size should not execute such an operator if at all possible.

-T offset
Set the paper size to the given pair of dimensions. This option takes its arguments in the same style as *-O*. It overrides any paper size special in the dvi file.

-U Disable a PostScript virtual memory saving optimization that stores the character metric information in the same string that is used to store the bitmap information. This is only necessary when driving the Xerox 4045 PostScript interpreter. It is caused by a bug in that interpreter that results in 'garbage' on the bottom of each character. Not recommended unless you must drive this printer.

-x num Set the magnification ratio to *num* /1000. Overrides the magnification specified in the .dvi file. Must be between 10 and 100000.

-X num
Set the horizontal resolution in dots per inch to *num.*

-Y num
Set the vertical resolution in dots per inch to *num.*

-Z Causes bitmapped fonts to be compressed before they are downloaded, thereby reducing the size of the PostScript font-downloading information. Especially useful at high resolutions or when very large fonts are used. Will slow down printing somewhat, especially on early 68000-based PostScript printers.

SEE ALSO

mf(1), afm2tfm(1), tex(1), latex(1), lpr(1), *dvips.tex.*

DIAGNOSTICS

NOTES

PostScript is a registered trademark of Adobe Systems Incorporated.

AUTHOR

Tomas Rokicki <rokicki@cs.stanford.edu>; extended to virtual fonts by Don Knuth. Path searching and configuration modifications by karl@cs.umb.edu.

NAME

dviselect − extract pages from DVI files

SYNOPSIS

dviselect [**−s**] [**−i** *infile*] [**−o** *outfile*] *list of pages* [*infile* [*outfile*]]

DESCRIPTION

Dviselect selects pages from a DVI file produced by TeX, creating a new DVI file usable by any of the TeX conversion programs, or even by *dviselect* itself.

A *range* is a string of the form *even*, *odd*, or *first:last* where both *first* and *last* are optional numeric strings, with negative numbers indicated by a leading underscore character ‘‘_’’. If both *first* and *last* are omitted, the colon may also be omitted, or may be replaced with an asterisk ‘‘*’’. A *page range* is a list of ranges separated by periods. A *list of pages* is described by a set of page ranges separated by commas and/or white space.

Dviselect actually looks at the ten *count* variables that TeX writes; the first of these (\count0) is the page number, with \count1 through \count9 having varied uses depending on which macro packages are in use. (Typically \count1 might be a chapter or section number.) A page is included in *dviselect*'s output if all its \count values match any one of the ranges listed on the command line. For example, the command ‘‘dviselect *.1,35:’’ might select everything in chapter 1, as well as pages 35 and up. ‘‘dviselect 10:30’’ would select pages 10 through 30 (inclusive). ‘‘:43’’ means everything up to and including page 43 (including negative-numbered pages). To get all even-numbered pages, use ‘‘even’’; to get all odd-numbered pages, use ‘‘odd’’. If a Table of Contents has negative page numbers, ‘‘:_1’’ will select it. Note that ‘‘*’’ must be quoted from the shell; the empty string is more convenient to use, if harder to read.

Instead of \count values, *dviselect* can also select by ‘‘absolute page number’’, where the first page is page 1, the second page 2, and so forth. Absolute page numbers are indicated by a leading equal sign ‘‘=’’. Ranges of absolute pages are also allowed: ‘‘dviselect =3:7’’ will extract the third through seventh pages. Dot separators are not legal in absolute ranges, and there are no negative absolute page numbers. Even/odd specifiers, however, are legal; ‘‘dviselect =even’’ selects every other page, starting with the second.

More precisely, an asterisk or an empty string implies no limit; an equal sign means absolute page number rather than \counts; a leading colon means everything up to and including the given page; a trailing colon means everything from the given page on; the word ‘‘even’’ means only even values shall be accepted; the word ‘‘odd’’ means only odd values shall be accepted; and a period indicates that the next \count should be examined. If fewer than 10 ranges are specified, the remaining \counts are left unrestricted (that is, ‘‘1:5’’ and ‘‘1:5.*’’ are equivalent). A single number *n* is treated as if it were the range *n:n*. An arbitrary number of page selectors may be given, separated by commas or whitespace; a page is selected if any of the selectors matches its \counts or absolute page number.

Dviselect normally prints the page numbers of the pages selected; the *−s* option suppresses this.

AUTHOR

Chris Torek, University of Maryland

SEE ALSO

dviconcat(1), latex(1), tex(1)
MC-TeX User's Guide
The TeXbook

BUGS

A leading ‘‘-’’ ought to be allowed for negative numbers, but it is currently used as a synonym for ‘‘:’’, for backwards compatibility.

Section or subsection selection will sometimes fail, for the DVI file lists only the \count values that were active when the page ended. Clever macro packages can alleviate this by making use of other ‘‘free’’ \count registers. Chapters normally begin on new pages, and do not suffer from this particular problem.

The heuristic that decides which arguments are page selectors and which are file names is often wrong. Using shell redirection or the −i and −o options is safest.

Dviselect does not adjust the parameters in the postamble; however, since these values are normally used only to size certain structures in the output conversion programs, and the parameters never need to be adjusted upward, this has not proven to be a problem.

NAME

dvitype − translate a dvi file for humans

SYNOPSIS

dvitype *dvi_file_name*

DESCRIPTION

The **dvitype** program translates a DVI (DeVice Independent) file output by (for example) **tex**(1) or **gftodvi**(1), to a file that humans can read. It also serves as a DVI file-validating program (i.e., if **dvitype** can read it, it's correct) and as an example of a DVI-reading program for future device drivers.

The output file can include all commands, just the important ones, or none at all (in which case only errors are reported). A subinterval of pages may be selected for transliteration; the magnification and resolution of the "output device" may be changed; and so on. All options are specified with an on-line dialog.

The *dvi_file_name* on the command line must be complete. The *.dvi* extension is not supplied if omitted. The output goes to *stdout*.

ENVIRONMENT

The environment variable TEXFONTS is used to search for the TFM files used in the DVI file. See **tex**(1) for the details of the searching. If TEXFONTS is not set, it uses the system default:
.:/usr/local/lib/texmf/fonts//tfm

SEE ALSO

gftype(1), **pktype**(1).
Donald E. Knuth, *TEXware*.

AUTHORS

Donald E. Knuth wrote the program. It was published as part of the *TEXware* technical report, available from the TEX Users Group. Howard Trickey and Pavel Curtis originally ported it to Unix.

NAME

gftodvi − make proof sheets from generic font files

SYNOPSIS

gftodvi [**−overflow-label-offset=***real*] [**−verbose**] [*gf_file_name*]

DESCRIPTION

The **gftodvi** program converts a generic font (*gf*) file output by, for example, **mf**(1), to a device independent (DVI) file (that can then be typeset using the same software that has already been written for TeX). The characters in the *gf* file will appear one per page, with labels, titles, and annotations as specified in Appendix H (Hardcopy Proofs) of *The METAFONTbook*.

gftodvi uses other fonts in addition to the main *gf* file. A 'gray' font is used to typeset the pixels that actually make up the character. (We wouldn't want all the pixels to be simply black, since then labels, key points, and other information would be lost.) A 'title' font is used for the information at the top of the page. A 'label' font is used for the labels on key points of the figure. A 'slant' font is used to typeset diagonal lines, which otherwise have to be simulated using horizontal and vertical rules. The default gray, title, and label fonts are *gray*, *cmr8*, and *cmtt10*, respectively; there is no default slant font.

To change the default fonts, you can give **special** commands in your METAFONT source file, or you can change the fonts online. An online dialog ensues if you end the *gf_file_name* with a '/'. For example,

gftodvi cmr10.300gf/
Special font substitution: grayfont black
OK; any more? grayfontarea /home/art/don/
OK; any more? slantfont /home/fonts/slantimagen6
OK; any more? <RET>

will use */home/art/don/black* as the 'gray' font and */home/fonts/slantimagen6* as the 'slant' font (this name indicates a font for lines with slope 1/6 at the resolution of an Imagen printer).

The *gf_file_name* on the command line must be complete. (The program prompts you for it if you don't give it.) Because the resolution is part of the extension, it would not make sense to append a default extension as is done with TeX or DVI-reading software. The output file name defaults to the same root as the *gf* file, with the *dvi* extension added. For example, the input file *cmr10.2602gf* would become *cmr10.dvi*.

OPTIONS

The argument to **−overflow-label-offset** specifies the distance from the right edge of the character bounding box at which the overflow equations (if any) are typeset. The value is given in TeX points. The default is a little over two inches.

Without the **−verbose** option, **gftodvi** operates silently. With it, a banner and progress report are printed on *stdout*.

ENVIRONMENT

gftodvi looks for *gf_file_name* using the environment variable GFFONTS. If that is not set, it uses the variable TEXFONTS. If that is not set, it uses the system default:
.:/usr/local/lib/texmf/fonts//gf

See **tex**(1) for the details of the searching.

FILES

/usr/local/lib/texmf/fonts/{gray.tfm,...}	The default fonts.
/usr/local/lib/texmf/mf/{gray.mf,...}	The METAFONT sources.

SEE ALSO

tex(1), **mf**(1).
Donald E. Knuth, *The METAFONTbook* (Volume C of *Computers and Typesetting*), Addison-Wesley, 1986, ISBN 0-201-13445-4.
Donald E. Knuth et al., *METAFONTware*.

AUTHORS

Donald E. Knuth wrote the program. It was published as part of the *METAFONTware* technical report, available from the TEX Users Group. Paul Richards ported it to Unix.

NAME

gftopk − convert generic font files to packed font files

SYNOPSIS

gftopk [**−v**] *gf_file_name* [*pk_file_name*]

DESCRIPTION

The **gftopk** program converts a generic font file output by, for example, **mf**(1), to a packed font file for use by DVI-reading programs. Packed font files (*pk* files) are much smaller than the corresponding *gf* files, so they are generally the font format of choice.

The *gf_file_name* on the command line must be complete. Because the resolution is part of the extension, it would not make sense to append a default extension as is done with TeX or DVI-reading software. The *pk_file_name* defaults to the same (stripped) name as *gf_file_name*, and it is placed in the current working directory with the *pk* suffix replacing *gf*. For example, the input file *cmr10.300gf* would become *cmr10.300pk*.

OPTIONS

Unless the **−v** switch is given, **gftopk** operates silently. With **−v**, the version number of the program and statistics about the packing are output to *stdout*.

ENVIRONMENT

gftopk looks for *gf_file_name* first in the current directory. If it is not present there, it uses the environment variable GFFONTS. If that is not set, it uses the variable TEXFONTS. If that is not set, it uses the system default:
.:/usr/local/lib/texmf/fonts//gf

See **tex**(1) for the details of the searching.

SEE ALSO

gftype(1), **pktogf**(1), **pktype**(1).
Donald E. Knuth et al., *METAFONTware*.

AUTHORS

Tomas Rokicki wrote the program. It was published as part of the *METAFONTware* technical report, available from the TeX Users Group. Paul Richards originally ported it to Unix.

NAME

gftype − translate a generic font file for humans to read

SYNTAX

gftype [**−i**] [**−m**] *gf_file_name* [*output_file_name*]

DESCRIPTION

The **gftype** program translates a *gf* (generic font) file output by, for example, **mf**(1), to a file that humans can read. It also serves as a *gf* file-validating program (i.e., if **gftype** can read it, it's correct) and as an example of a *gf*-reading program for other software that wants to read *gf* files.

The *gf_file_name* on the command line must be complete. Because the resolution is part of the extension, it would not make sense to append a default extension as is done with TEX or DVI-reading software. If no *output_file_name* is specified, the output goes to *stdout*.

OPTIONS

The output file includes bitmap images of the characters, if **−i** is given on the command line, and a symbolic listing of the *gf* commands, if **−m** is specified.

ENVIRONMENT

gftype looks for *gf_file_name* using the environment variable GFFONTS. If that is not set, it uses the variable TEXFONTS. If that is not set, it uses the system default:
.:/usr/local/lib/texmf/fonts//gf

SEE ALSO

dvitype(1), **pktype**(1).
Donald E. Knuth et al., *METAFONTware*.

AUTHORS

David Fuchs and Don Knuth wrote the program. It was published as part of the *METAFONTware* technical report, available from the TEX Users Group. Paul Richards originally ported it to Unix.

NAME

initex − initial TeX for making format (.fmt) files

DESCRIPTION

initex is a special version of the TEX program that has no preloaded macro packages, but is capable of converting a macro package into a special preformatted binary file, called a *format* (*.fmt*) file. That format file can subsequently be read at high speed by **virtex**.

Major macro packages may require TEX to process many thousands of lines of macros, and open and read scores of font files, all of which would contribute to a sizable startup overhead if the job had to be done every time TEX was run. **initex** allows TEX to do the job once, and then save the results in a binary format file that can be later loaded more rapidly by **virtex**.

initex is normally required only at the time TEX is installed, or whenever major macro packages are updated. Thus, it will be rare for anyone but system installers to invoke it. Nevertheless, it is just a normal program without special privileges, so ordinary users can use it to prepare a private format file.

Here is how you can ask **initex** to prepare format files for several major packages:

For plain **tex**:

initex 'plain \dump'
mv plain.fmt tex.fmt
The single quotes are necessary to protect the backslash from interpretation as a Unix shell escape character.

For **amstex**:

initex '&tex amstex \dump'
The single quotes around the first argument protect the ampersand from interpretation as a Unix shell background request.

For **etex**:

initex '&tex eplain \dump'
mv eplain.fmt etex.fmt

For **lamstex**:

initex '&tex amstexl \input lamstex \dump'
mv amstexl.fmt lamstex.fmt

For **latex**:

initex 'lplain \dump'
mv lplain.fmt latex.fmt

For **slitex**:

initex 'splain \dump'
mv splain.fmt slitex.fmt

For **texinfo**:

initex '&tex texinfo @dump'
The last argument is different from the previous examples because **texinfo** redefines the TEX escape character from backslash to at-sign. As for **amstex**, the file *tex.fmt* must already be available.

Note that several of these require that the format file for plain TEX be available, since the first argument word *&tex* asks for the loading of the file *tex.fmt*.

It does not matter whether the command-line arguments are passed as a single argument, or as separate arguments, since TEX reconstructs the command line anyway.

SEE ALSO

amslatex(1), **amstex**(1), **lamstex**(1), **latex**(1), **slitex**(1), **tex**(1), **virtex**(1).

NAME

latex − structured text formatting and typesetting

SYNOPSIS

latex [first line]

DESCRIPTION

The LaTeX language is described in the book LaTeX − *A Document Preparation System*. LaTeX is a TeX macro package, not a modification to the TeX source program, so all the capabilities described in **tex**(1) are present.

The LaTeX macros encourage writers to think about the content of their documents, rather than the form. The ideal, very difficult to realize, is to have no formatting commands (like ‘‘switch to italic’’ or ‘‘skip 2 picas’’) in the document at all; instead, everything is done by specific markup instructions: ‘‘emphasize’’, ‘‘start a section’’.

The primary source of documentation for LaTeX is the LaTeX manual referenced below, and the local guide in the file *local-guide.tex* or *local.tex* or some such somewhere in the directory /usr/local/lib/texmf/tex.

SEE ALSO

amslatex(1), **amstex**(1), **slitex**(1), **tex**(1),
Leslie Lamport, LaTeX − *A Document Preparation System*, Addison-Wesley, 1985, ISBN 0-201-15790-X.

NAME

makeindex − a general purpose, formatter-independent index processor

SYNOPSIS

makeindex [−c] [−g] [−i] [−l] [−o *ind*] [−p *num*] [−q] [−r] [−s *sfile*] [−t *log*] [*idx0 idx1 idx2*...]

DESCRIPTION

The program *makeindex* is a general purpose hierarchical index generator; it accepts one or more input files (often produced by a text formatter such as TEX (*tex*(1L)) or *troff*(1), sorts the entries, and produces an output file which can be formatted. The index can have up to three levels (0, 1, and 2) of subitem nesting. The way in which words are flagged for indexing within the main document is specific to the formatter used; *makeindex* does *not* automate the process of selecting these words. As the output index is hierarchical, *makeindex* can be considered complimentary to the *awk*(1)-based *make.index*(1L) system of Bentley and Kernighan, which is specific to *troff*(1), generates non-hierarchical indices, and employs a much simpler syntax for indicating index entries. For illustration of use with *troff* and *TEX*, see the section EXAMPLES below.

The formats of the input and output files are specified in a style file; by default, input is assumed to be a *.idx* file, as generated by LATEX.

Unless specified explicitly, the base name of the first input file (*idx0*) is used to determine the names of other files. For each input file name specified, a file of that name is sought. If this file is not found and the file name has no extension, the extension *.idx* is appended. If no file with this name is found, *makeindex* aborts.

If exactly one input file was given and no explicit style file was specified using **−s**, *makeindex* uses a file with the extension *.mst* as default style file (when present).

For important notes on how to select index keywords, see the document by Lamport cited below. As an issue separate from selecting index keywords, a systematic mechanism for placing index terms in a document is suggested in *Index Preparation and Processing*, a paper cited below.

OPTIONS

−c Compress intermediate blanks (ignoring leading and trailing blanks and tabs). By default, blanks in the index key are retained.

−g Employ German word ordering in the index, in accord with rules set forth in DIN 5007. By default, *makeindex* employs a word ordering in which precedence is: symbols, numbers, uppercase letters, lowercase letters. The sequence in German word ordering is: symbols, lowercase letters, uppercase letters, numbers. Additionally, this option enables *makeindex* to recognize the German TEX-commands {"a, "o, "u and "s} as {ae, oe, ue and ss} during the sorting of the entries. The quote character must be redefined in a style file (for example, redefine quote as '+'). If the quote character is not redefined, *makeindex* will produce an error message and abort.

−i Take input from *stdin*. When this option is specified and **−o** is not, output is written to *stdout*.

−l Letter ordering; by default, word ordering is used (see the ORDERING section).

−o *ind* Employ *ind* as the output index file. By default, the file name is created by appending the extension *.ind* to the base name of the first input file (*idx0*).

−p *num* Set the starting page number of the output index file to be *num* (useful when the index file is to be formatted separately). The argument *num* may be numerical or one of the following:

any The starting page is the last source page number plus 1.

odd The starting page is the first odd page following the last source page number.

even The starting page is the first even page following the last source page number.

The last source page is obtained by searching backward in the log file for the first instance of a number included within paired square brackets ([...]). If a page number is missing or the log file is not found, no attempt will be made to set the starting page number. The source log file name is determined by appending the extension *.log* to the base name of the first input file (*idx0*).

−q Quiet mode; send no messages to *stderr*. By default, progress and error messages are sent to *stderr* as well as to the transcript file.

−r Disable implicit page range formation; page ranges must be created by using explicit range operators; see SPECIAL EFFECTS below. By default, three or more successive pages are automatically abbreviated as a range (e.g. 1—5).

−s sty Employ *sty* as the style file (no default). The environment variable INDEXSTYLE defines the path where the style file should be found.

−t log Employ *log* as the transcript file. By default, the file name is created by appending the extension *.ilg* to the base name of the first input file (*idx0*).

STYLE FILE

The style file informs *makeindex* about the format of the *.idx* input files and the intended format of the final output file; examples appear below. This file can reside anywhere in the path defined by the environment variable INDEXSTYLE. The style file contains a list of <*specifier*, *attribute*> pairs. There are two types of specifiers: input and output. Pairs do not have to appear in any particular order. A line begun by '%' is a comment. In the following list of specifiers and arguments, <string> is an arbitrary string delimited by double quotes ("..."), <char> is a single letter embraced by single quotes ('...'), and <number> is a nonnegative integer. The maximum length of a <string> is 2048. A literal backslash or quote must be escaped (by a backslash). Anything not specified in the style file will be assigned a default value, which is shown at the head of the rightmost column.

INPUT STYLE SPECIFIERS

actual <char> ´@´
Symbol indicating that the next entry is to appear in the output file.

arg_close <char> ´}´
Closing delimiter for the index entry argument.

arg_open <char> ´{´
Opening delimiter for the index entry argument.

encap <char> ´|´
Symbol indicating that the rest of the argument list is to be used as the encapsulating command for the page number.

escape <char> ´\\´
Symbol which escapes the following letter, unless its preceding letter is **escape**. Note: **quote** is used to escape the letter which immediately follows it, but if it is preceded by **escape**, it is treated as a ordinary character. These two symbols *must* be distinct.

keyword <string> "\\indexentry"
Command which tells *makeindex* that its argument is an index entry.

level <char> ´!´
Delimiter denoting a new level of subitem.

quote <char> ´"´
Note: **quote** is used to escape the letter which immediately follows it, but if it is preceded by **escape**, it is treated as a ordinary character. These two symbols *must* be distinct.

range_close <char> ´)´
Closing delimiter indicating the end of an explicit page range.

range_open <char> ´(´
Opening delimiter indicating the beginning of an explicit page range.

OUTPUT STYLE SPECIFIERS

preamble <string>
"\\begin{theindex}\n"
Preamble of output file.

postamble <string>
"\n\n\\end{theindex}\n"
Postamble of output file.

setpage_prefix <string>
"\n \\setcounter{page}{"
Prefix of command which sets the starting page number.

setpage_suffix <string>
"}\n"
Suffix of command which sets the starting page number.

group_skip <string>
"\n\n \\indexspace\n"
Vertical space to be inserted before a new group begins.

headings_flag <string>
0
Flag indicating treatment of new group headers, which are inserted when before a new group (symbols, numbers, and the 26 letters): positive values cause an uppercase letter to be inserted between prefix and suffix, and negative values cause a lowercase letter to be inserted (default is 0, which produces no header).

heading_prefix <string>
""
Header prefix to be inserted before a new letter begins.

symhead_positive <string>
"Symbols"
Heading for symbols to be inserted if **headings_flag** is positive.

symhead_negative <string>
"symbols"
Heading for symbols to be inserted if **headings_flag** is negative.

numhead_positive <string>
"Numbers"
Heading for numbers to be inserted if **headings_flag** is positive.

numhead_negative <string>
"numbers"
Heading for numbers to be inserted if **headings_flag** is negative.

item_0 <string>
"\n \\item "
Command to be inserted between two primary (level 0) items.

item_1 <string>
"\n   \\subitem "
Command to be inserted between two secondary (level 1) items.

item_2 <string>
"\n   \\subsubitem "
Command to be inserted between two level 2 items.

item_01 <string>
"\n   \\subitem "
Command to be inserted between a level 0 item and a level 1 item.

item_x1 <string>
"\n   \\subitem "
Command to be inserted between a level 0 item and a level 1 item, where the level 0 item does not have associated page numbers.

item_12 <string>
"\n   \\subsubitem "
Command to be inserted between a level 1 item and a level 2 item.

item_x2 <string>
"\n   \\subsubitem "
Command to be inserted between a level 1 item and a level 2 item, where the level 1 item does not have associated page numbers.

delim_0 <string>
", "
Delimiter to be inserted between a level 0 key and its first page number (default: comma followed by a blank).

delim_1 <string> ", "
Delimiter to be inserted between a level 1 key and its first page number (default: comma followed by a blank).

delim_2 <string> ", "
Delimiter to be inserted between a level 2 key and its first page number (default: comma followed by a blank).

delim_n <string> ", "
Delimiter to be inserted between two page numbers for the same key in any level (default: comma followed by a blank).

delim_r <string> "--"
Delimiter to be inserted between the starting and ending page numbers of a range.

delim_t <string> ""
Delimiter to be inserted at the end of a page list. This delimiter has no effect on entries which have no associated page list.

encap_prefix <string> "\\"
First part of prefix for the command which encapsulates the page number.

encap_infix <string> "{"
Second part of prefix for the command which encapsulates the page number.

encap_suffix <string> "}".
Suffix for the command which encapsulates the page number.

line_max <number> 72
Maximum length of a line in the output, beyond which a line wraps.

indent_space <string> "\t\t"
Space to be inserted in front of a wrapped line (default: two tabs).

indent_length <number> 16
Length of **indent_space** (default: 16, equivalent to 2 tabs).

suffix_2p <string> ""
Delimiter to replace the range delimiter and the second page number of a two page list. When present, it overrides **delim_r**. Example: "f.".

suffix_3p <string> ""
Delimiter to replace the range delimiter and the second page number of a three page list. When present, it overrides **delim_r** and **suffix_mp**. Example: "ff.".

suffix_mp <string> ""
Delimiter to replace the range delimiter and the second page number of a multiple page list (three or more pages). When present, it overrides **delim_r**. Example: "f.".

EXAMPLES

TEX EXAMPLE

The following example shows a style file called *book.ist*, which defines an index for a book which can be formatted independently of the main source:

```
preamble
"\\documentstyle[12pt]{book}
\\begin{document}
\\begin{theindex}
{\\small\n"
postamble
```

```
"\n\n}
\\end{theindex}
\\end{document}\n"
```

Assuming that a particular book style requires the index (as well as any chapters) to start from an odd page number, and that the input file is named *foo.idx*, the following command line produces output in file *footmp.ind*:

```
makeindex  -s book.ist  -o footmp.ind  -p odd  foo
```

Here a non-default output file name is used to avoid clobbering the output for the book itself (presumably *foo.dvi*, which would have been the default name for the index output file!).

TROFF EXAMPLE

A sample control file for creating an index, which we will assume resides in the file *sample.ist*:

```
keyword "IX:"
preamble
".\\\" start of index output
\".\\\" enter two column mode
.2C
.SH
.ce
INDEX
.XS
INDEX
.XE
.R
.ps 9p
.vs 11p
.sp
.de I1
.ti 0.25i
..
.de I2
.ti 0.5i
.."
postamble "\n.\\\" end of index output"
setpage_prefix "\n.nr % "
setpage_suffix ""
group_skip "\n.sp 1.0"
headings_flag 1
heading_prefix "\n.IS\n"
heading_suffix "\n.IE"
item_0 "\n.br\n"
item_1 "\n.I1\n"
item_2 "\n.I2\n"
item_01 "\n.I1\n"
item_x1 "\n.I1\n"
item_12 "\n.I2\n"
item_x2 "\n.I2\n"
delim_0 ", "
delim_1 ", "
delim_2 ", "
delim_r "-"
delim_t "."
encap_prefix "\\fB"
```

```
encap_infix ""
encap_suffix "\\fP"
indent_space ""
indent_length 0
```

The local macro package may require modification, as in this example of an extension to the **–ms** macros (note that at some sites, this macro should *replace* a pre-existing macro of the same name):

```
.
.de IX
.ie '\\n(.z'' .tm IX: \\$1 \\$2 \\$3 \\$4 \\$5 \\$6 \\$7 \\$8 \\$9 {\\n(PN}
.el \\!.IX \\$1 \\$2 \\$3 \\$4 \\$5 \\$6 \\$7 \\$8 \\$9 {\\n(PN}
..
```

(note that the string `{\\n(PN}` is separated from the rest of the line by a tab. If your local macro package does not contain this extension, just include those lines at the beginning of your file. Here is a simple *troff*(1) input file, which we will assume is named *sample.txt*:

```
This is a sample file to test the \fImakeindex\fP(1L)
program, and see
.IX {indexing!programs!C language}
.IX {makeindex@\fImakeindex\fP(1L)}
.bp
.rs
.IX {Knuth}
.IX {typesetting!computer-aided}
how well it functions in the \fItroff\fP(1) environment.
```

Note that index entries are indicated by the **.IX** macro, which causes the following text to be written to *stdout* along with the current page number.

CREATING THE INDEX FILE IN THE BOURNE SHELL

To create an input file for *makeindex*, **in the Bourne shell** environment, do the equivalent at your site of the command:

```
psroff -ms -Tpsc -t sample.txt > /dev/null 2> sample.tmp
```

Some sites will require *ditroff* instead of *psroff*. To filter out any genuine error messages, invoke *grep*(1):

```
grep '^IX: ' sample.tmp > sample.idx
```

CREATING THE INDEX FILE USING UCSF ENHANCED TROFF/TRANSCRIPT

With UCSF Enhanced troff/TRANSCRIPT, the **–I** option of *psroff*(1L) can produce both formatter output and an index file:

```
psroff -ms -I sample.inp -Tpsc sample.txt
```

If it is wished to suppress the formatter output:

```
psroff -ms -I sample.inp -Tpsc -t sample.txt > /dev/null
```

COMPLETING THE INDEX

Any of the above procedures leaves the input for *makeindex* in *sample.inp*. The next step is to invoke *makeindex*:

```
makeindex -s sample.ist sample.idx
```

This leaves *troff*(1)-ready output in the file *sample.ind*.

ORDERING

By default, *makeindex* assumes *word ordering*; if the **–l** option is in effect, *letter ordering* is used. In word ordering, a blank precedes any letter in the alphabet, whereas in letter ordering, it does not count at all. This is illustrated by the following example:

word order *letter order*

sea lion	seal
seal	sea lion

Numbers are always sorted in numeric order. For instance,

9 (nine), 123
10 (ten), see Derek, Bo

Letters are first sorted without regard to case; when words are identical, the uppercase version precedes its lowercase counterpart.

A special symbol is defined here to be any character not appearing in the union of digits and the English alphabetic characters. Patterns starting with special symbols precede numbers, which precede patterns starting with letters. As a special case, a string starting with a digit but mixed with non-digits is considered to be a pattern starting with a special character.

SPECIAL EFFECTS

Entries such as

```
\indexentry{alpha}{1}
\indexentry{alpha!beta}{3}
\indexentry{alpha!beta!gamma}{10}
```

in the input file will be converted to

```
\item alpha, 1
   \subitem beta, 3
      \subsubitem gamma, 10
```

in the output index file. Notice that the **level** symbol ('!') is used above to delimit hierarchical levels.

It is possible to make an item appear in a designated form by using the **actual** ('@') operator. For instance,

```
\indexentry{alpha@{\it alpha\/}}{1}
```

will become

```
\item {\it alpha\/},  1
```

after processing. The pattern preceding '@' is used as sort key, whereas the one following it is written to the output file. Note that two appearances of the same key, one with and one without the **actual** operator, are regarded as *distinct* entries.

The item, subitem, and subsubitem fields may have individual sort keys:

```
\indexentry{aa@{\it aa\/}!bb@{\it bb\/}!cc@{\it cc\/}}{1}
```

This will be converted to

```
\item {\it aa}, 1
   \subitem {\it bb}, 3
      \subsubitem {\it cc}, 10
```

It is possible to encapsulate a page number with a designated command using the **encap** ('|') operator:

```
\indexentry{alpha|bold}{1}
```

will be converted to

```
\item alpha, \bold{1}
```

where, with a suitable definition for TeX, `\bold{n}` will expand to `{\bf n}`. In this example, the three output attributes associated with page encapsulation **encap_prefix**, **encap_infix**, and **encap_suffix**, correspond to backslash, left brace, and right brace, respectively. This mechanism allows page numbers to be set in different fonts. For example, the page where the definition of a keyword appears can be in one font, the location of a primary example can be in another font, and other appearances in yet a third font.

The **encap** operator can also be used to create cross references in the index:

```
\indexentry{alpha|see{beta}}{1}
```

will become

```
\item alpha, \see{beta}{1}
```

in the output file, where

```
\see{beta}{1}
```

will expand to

```
{\it see\/} beta
```

Note that in a cross reference like this the page number disappears.

A pair of **encap** concatenated with **range_open** ('|(') and **range_close** ('|)') creates an explicit page range:

```
\indexentry{alpha|(}{1}
\indexentry{alpha|)}{5}
```

will become

```
\item alpha, 1—5
```

Intermediate pages indexed by the same key will be merged into the range implicitly. This is especially useful when an entire section about a particular subject is to be indexed, in which case only the range opening and closing operators need to be inserted at the beginning and end of the section. Explicit page range formation can also include an extra command to set the page range in a designated font:

```
\indexentry{alpha|(bold}{1}
\indexentry{alpha|)}{5}
```

will become

```
\item alpha, \bold{1--5}
```

Several potential problems are worth mentioning. First, entries like

```
\indexentry{alpha|(}{1}
\indexentry{alpha|bold}{3}
\indexentry{alpha|)}{5}
```

will be interpreted as

```
\item alpha, \bold{3}, 1--5
```

but with a warning message in the transcript about encountering an inconsistent page encapsulator. An explicit range beginning in a Roman page number and ending in Arabic is also considered an error. In this instance, (if possible) the range is broken into two subranges, one in Roman and the other in Arabic. For instance,

```
\indexentry{alpha|(}{i}
\indexentry{alpha}{iv}
\indexentry{alpha}{3}
\indexentry{alpha|)}{7}
```

will be turned into

```
\item alpha, i--iv, 3--7
```

with a warning message in the transcript file complaining about an illegal range formation.

Finally, every special symbol mentioned in this section may be escaped by the **quote** operator ('"'). Thus

```
\indexentry{alpha"@beta}{1}
```

will actually become

```
\item alpha@beta,  1
```

as a result of executing *makeindex*. The quoting power of **quote** is eliminated if it is immediately preceded by **escape** ('\'). For example,

```
\indexentry{f\"ur}{1}
```

becomes

```
\item f\"ur, 1
```

which represents an umlaut-accented 'u' to the TeX family of processors.

From version 2.11 of *makeindex*, the **quote** operator may quote *any* character in the range 1 ... 255. Character 0 is excluded because it is used internally in the *makeindex* source code as a string terminator. With this change, sort keys can be created for all eight-bit characters except 0. The sorting order is

punctuation characters (in ASCII order),
digits,
control characters (1 ... 31),
space (32),
letters (ignoring case),
characters 127 ... 255.

Here is an example showing the indexing of all printable ASCII characters other than letters and digits, assuming the default TeX format. For convenience, the page number references are the corresponding ASCII ordinal values.

```
\indexentry{" @"  (space)}{32}
\indexentry{"!@"! (exclamation point)}{33}
\indexentry{""@"" (quotation mark)}{34}
\indexentry{"#@"\# (sharp sign)}{35}
\indexentry{"$@"\$ (dollar sign)}{36}
\indexentry{"%@"\% (percent sign)}{37}
\indexentry{"&@"\& (ampersand)}{38}
\indexentry{"<@"$<$ (left angle bracket)}{60}
\indexentry{"=@"= (equals)}{61}
\indexentry{">@"$>$ (right angle bracket)}{62}
\indexentry{"?@"? (query)}{63}
\indexentry{"@@"@ (at sign)}{64}
\indexentry{"[@"[ (left square bracket)}{91}
\indexentry{"\@"\verb=\= (backslash)}{92}
\indexentry{"]@"] (right square bracket)}{93}
\indexentry{"^@"\verb=^= (caret)}{94}
\indexentry{"_@"\verb=_= (underscore)}{95}
\indexentry{"`@"\verb=~= (grave accent)}{96}
\indexentry{"{@"\"{ (left brace)}{123}
\indexentry{"|@"\verb="|= (vertical bar)}{124}
\indexentry{"}@"\"} (right brace)}{125}
\indexentry{"~@"\verb=~= (tilde)}{126}
```

Characters in the actual fields following the '@' character which have special significance to TeX must be represented as control sequences, or as math mode characters. Note particularly how the entries for the at sign, left and right braces, and the vertical bar, are coded. The index file output by *makeindex* for this example looks like this:

```
\begin{theindex}

  \item ! (exclamation point), 33
  \item " (quotation mark), 34
```

```
    \item \# (sharp sign), 35
    \item \$ (dollar sign), 36
    \item \% (percent sign), 37
    \item \& (ampersand), 38
    \item $<$ (left angle bracket), 60
    \item = (equals), 61
    \item $>$ (right angle bracket), 62
    \item ? (query), 63
    \item @ (at sign), 64
    \item [ (left square bracket), 91
    \item \verb=\= (backslash), 92
    \item ] (right square bracket), 93
    \item \verb=ˆ= (caret), 94
    \item \verb=_= (underscore), 95
    \item \verb=˜= (grave accent), 96
    \item \{ (left brace), 123
    \item \verb=|= (vertical bar), 124
    \item \} (right brace), 125
    \item \verb=˜= (tilde), 126

    \indexspace

    \item   (space), 32

  \end{theindex}
```

FILES

/usr/local/bin/makeindex	executable file
/usr/local/lib/tex/macros/idxmac-amstex.tex	TEX macro file used by *makeindex*
/usr/local/lib/tex/macros/idxmac.tex	TEX macro file used by *makeindex*
/usr/local/lib/tex/macros/makeidx.doc	TEX macro file used by *makeindex*
/usr/local/lib/tex/macros/makeidx.sty	TEX macro file used by *makeindex*

SEE ALSO

ditroff(1L), latex(1L), make.index (1L), qsort(3), tex(1L), troff(1L)

UCSF Enhanced troff/TRANSCRIPT — An Overview, R. P. C. Rodgers and Conrad Huang, LSMB Technical Report 90-2, UCSF School of Pharmacy, San Francisco, 1990.

Index Preparation and Processing, Pehong Chen and Michael A. Harrison, *Software: Practice and Experience*, **19**(9), 897–915, September 1988.

Automating Index Preparation, Pehong Chen and Michael A. Harrison. Technical Report 87/347, Computer Science Division, University of California, Berkeley, 1987 (a LATEX document supplied with *makeindex*).

MakeIndex: An Index Processor for LATEX, Leslie Lamport, February 1987 (a LATEX document supplied with *makeindex*).

Tools for Printing Indices, Jon L. Bentley and Brian W. Kernighan, *Electronic Publishing — Origination, Dissemination, and Design*, 1(1), 3–18, June 1988 (also available as: Computing Science Technical Report No. 128, AT&T Bell Laboratories, Murray Hill, NJ 07974, 1986).

AUTHOR

Pehong Chen, Chen & Harrison International Systems, Inc. Palo Alto, California, USA <chen@renoir.berkeley.edu>.

Manual page extensively revised and corrected, and *troff* (1) examples created by Rick P. C. Rodgers, UCSF

School of Pharmacy <rodgers@cca.ucsf.edu>.

ACKNOWLEDGMENTS

Leslie Lamport contributed significantly to the design. Michael Harrison provided valuable comments and suggestions. Nelson Beebe improved on the portable version, and maintains the source distribution for the TEX Users Group. Andreas Brosig contributed to the German word ordering. The modification to the **−ms** macros was derived from a method proposed by Ravi Sethi of AT&T Bell Laboratories. The *LOG* and *CONTRIB* files in the *makeindex* source distribution record other contributions.

NAME

mf, inimf, virmf – Metafont, a language for font and logo design

SYNOPSIS

mf [first line]

inimf [first line]

virmf [first line]

DESCRIPTION

METAFONT reads the program in the specified files and outputs font rasters (in *gf* format) and font metrics (in *tfm* format). The METAFONT language is described in *The METAFONTbook*.

Like TEX, METAFONT is normally used with a large body of precompiled macros, and font generation in particular requires the support of several macro files. The basic program as compiled is called **inimf**; it can be used to precompile macros into a *.base* file. The **virmf** variant is used to reload the *.base* files quickly. Typically, **virmf** has links to it by the names of the various base files. For example, if you link **virmf** to **foo** and then execute the program **foo**, you get the base file *foo.base*. You can load a different base by saying, e.g., *virmf &mybase*.

As described in *The METAFONTbook*, the command line (or first input line) should otherwise begin with a file name or a \controlsequence. The normal usage is to say

> **mf** '\mode=<printengine>; [mag=magstep(*n*);]' input *font*

to start processing *font.mf*. The single quotes are the best way of keeping the Unix shell from misinterpreting the semicolons and from removing the \ character, which is needed here to keep METAFONT from thinking that you want to produce a font called *mode*. (Or you can just say *mf* and give the other stuff on the next line, without quotes.) Other control sequences, such as *batchmode* (for silent operation) can also appear. The name *font* will be the "jobname", and is used in forming output file names. If METAFONT doesn't get a file name in the first line, the jobname is *mfput*. The default extension, *.mf*, can be overridden by specifying an extension explicitly.

A log of error messages goes into the file *jobname*.*log*. The output files are *jobname*.*tfm* and *jobname*.<number>*gf*, where <number> depends on the resolution and magnification of the font. The *mode* in this example is shown generically as <printengine>, a symbolic term for which the name of an actual device or, most commonly, the name *localfont* (see below) must be substituted. If the mode is not specified or is not valid for your site, METAFONT will default to *proof* mode which produces large character images for use in font design and refinement. Proof mode can be recognized by the suffix *.2602gf* after the jobname. Examples of proof mode output can be found in *Computer Modern Typefaces* (Volume E of *Computers and Typesetting*). The system of *magsteps* is identical to the system used by TEX, with values generally in the range 0.5, 1.0, 2.0, 3.0, 4.0 and 5.0. A listing of *gf* numbers for 118-dpi, 240-dpi and 300-dpi fonts is shown below.

MAGSTEP	118 dpi	240 dpi	300 dpi
mag=magstep(0)	118	240	300
mag=magstep(0.5)	129	263	329
mag=magstep(1)	142	288	360
mag=magstep(2)	170	346	432
mag=magstep(3)	204	415	518
mag=magstep(4)	245	498	622
mag=magstep(5)	294	597	746

Magnification can also be specified not as a magstep but as an arbitrary value, such as 1.315, to create special character sizes.

Before font production can begin, it is necessary to set up the appropriate base files. The minimum set of components for font production for a given print-engine is the *plain.mf* macro file and the local *mode_def* file. The macros in *plain.mf* can be studied in an appendix to the *METAFONTbook*; they were developed by Donald E. Knuth, and this file should never be altered except when it is officially upgraded. Each *mode_def* specification helps adapt fonts to a particular print-engine. There is a regular discussion of *mode_def*s in

TUGboat, the journal of the TEX Users Group. The local ones in use on this computer should be in */usr/local/lib/texmf/mf/modes.mf*. With only *plain.mf* and the *modes* file loaded it is possible to create fonts of simple characters, such as those used for the METAFONT logo, and those used for the LATEX line and circle fonts, but the production of Computer Modern fonts would be speeded by making a *cmmf.base* file (which includes the macros in *cmbase.mf* as well as those in *plain.mf*).

Several environment variables can be used to set up directory paths to search when METAFONT opens a file for input. For example, the **csh** command

setenv MFINPUTS .:~me/mylib:/usr/local/lib/texmf/mf

or the **sh** command sequence

MFINPUTS=.:~me/mylib:/usr/local/lib/texmf/mf
export MFINPUTS

would cause all invocations of METAFONT and its derivatives to look for \input files first in the current directory, then in a hypothetical user's *mylib*, and finally in the system library. Normally, the user will place the command sequence which sets up the MFINPUTS environment variable in his or her *.login* or *.profile* file.

The *e* response to METAFONT's error-recovery mode invokes the system default editor at the erroneous line of the source file. There is an environment variable, MFEDIT, that overrides the default editor. It should contain a string with "%s" indicating where the filename goes and "%d" indicating where the decimal linenumber (if any) goes. For example, an MFEDIT string for the **vi** editor can be set with the **csh** command

setenv MFEDIT "/usr/ucb/vi +%d %s"

The **ENVIRONMENT** section below lists the relevant environment variables, and their defaults.

A convenient file in the library is *null.mf*, containing nothing. When **mf** can't find the file it thinks you want to input, it keeps asking you for another file name; responding 'null' gets you out of the loop if you don't want to input anything.

ONLINE GRAPHICS OUTPUT

METAFONT can use most modern displays, so you can see its output without printing. Chapter 23 of *The METAFONTbook* describes what you can do. This implementation of METAFONT uses environment variables to determine which display device you want to use. First it looks for a variable MFTERM, and then for TERM. If it can't find either, you get no online output. Otherwise, the value of the variable determines the device to use: **hp2627**, **sun** (for SunView), **tek**, **uniterm** (for an Atari ST Tek 4014 emulator), **xterm** (for either X10 or X11). Some of these devices may not be supported in all METAFONT executables; the choice is made at compilation time.

ENVIRONMENT

The default values for all environment variables are set at the time of compilation in the file *kpathsea/paths.h*. See the Kpathsearch library documentation (the 'Path specifications' node) for the details of the searching.

If the environment variable TEXMFOUTPUT is set, METAFONT attempts to put its output files in it, if they cannot be put in the current directory. Again, see **tex**(1).

MFINPUTS	Search path for *input* and *openin* files. Default: *.:/usr/local/lib/texmf/mf//:/usr/local/lib/texmf/fonts//src//*
MFBASES	Search path for base files. Default: *.:/usr/local/lib/texmf/ini*
MFPOOL	Search path for METAFONT internal strings. (Used by **inimf** only.) Default: */usr/local/lib/texmf/ini*
MFEDIT	Command template for switching to editor. Default: *emacs +%d %s*
MFTERM	Determines the online graphics display. If MFTERM is not set, and DISPLAY is set, the Metafont window support for X is used. (DISPLAY must be set to a valid X server specification, as usual.) If neither MFTERM nor DISPLAY is set, TERM is used to guess the window support to use.

MAKETEXMF	Analogous to TEX's MAKETEXTEX, q.v.
USE_MAKETEXMF	Analogous.

FONT UTILITIES

A number of utility programs are available. The following is a partial list of available utilities and their purpose. Consult your local METAFONT guru for details.

gftopk	Takes a *gf* file and produces a more tightly packed *pk* font file.
gftodvi	Produces proof sheets for fonts.
gftype	Displays the contents of a *gf* file in mnemonics and/or images.
pktype	Mnemonically displays the contents of a *pk* file.
mft	Formats a source file as shown in *Computer Modern Typefaces*.

FILES

/usr/local/lib/texmf/ini/mf.pool	Encoded text of METAFONT's messages.
/usr/local/lib/texmf/ini/.base*	Predigested METAFONT base files.
/usr/local/lib/texmf/mf//plain.mf	The standard base.
/usr/local/lib/texmf/mf//modes.mf	The file of *mode_def* s for your site's various printers

SUGGESTED READING

Donald E. Knuth, *The METAFONTbook* (Volume C of *Computers and Typesetting*), Addison-Wesley, 1986, ISBN 0-201-13445-4.
Donald E. Knuth, *METAFONT: The Program* (Volume D of *Computers and Typesetting*), Addison-Wesley, 1986, ISBN 0-201-13438-1.
Donald E. Knuth, *Computer Modern Typefaces* (Volume E of *Computers and Typesetting*), Addison-Wesley, 1986, ISBN 0-201-13446-2.
TUGboat (the journal of the TEX Users Group).

COMMENTS

Warning: "Type design can be hazardous to your other interests. Once you get hooked, you will develop intense feelings about letterforms; the medium will intrude on the messages that you read. And you will perpetually be thinking of improvements to the fonts that you see everywhere, especially those of your own design."

SEE ALSO

gftopk(1), **gftodvi**(1), **gftype**(1), **mft**(1), **pltotf**(1), **tftopl**(1).

BUGS

On January 4, 1986 the "final" bug in METAFONT was discovered and removed. If an error still lurks in the code, Donald E. Knuth promises to pay a finder's fee which doubles every year to the first person who finds it. Happy hunting.

AUTHORS

METAFONT was designed by Donald E. Knuth, who implemented it using his WEB system for Pascal programs. It was originally ported to Unix by Paul Richards at the University of Illinois at Urbana-Champaign. This page was mostly written by Pierre MacKay.

NAME

pktogf − convert packed font files to generic font files

SYNOPSIS

pktogf [**−v**] *pk_file_name* [*gf_file_name*]

DESCRIPTION

The **pktogf** program converts a packed font file (*pk*) to a generic font file (*gf*). Packed font files are much smaller than the corresponding *gf* files, but some DVI readers only understand *gf* files.

The *pk_file_name* on the command line must be complete. Because the resolution is part of the extension, it would not make sense to append a default extension as is done with TEX or DVI-reading software. The *gf_file_name* defaults to the same (stripped) name as *pk_file_name*, and it is placed in the current working directory with the *gf* suffix replacing *pk*. For example, the input file *io.300pk* would become *io.300gf*.

OPTIONS

Unless the **−v** switch is given, **pktogf** operates silently. With **−v**, the version number of the program and statistics about the unpacking are output to *stdout*.

ENVIRONMENT

pktogf looks for *pk_file_name* using the environment variable PKFONTS. If that is not set, it uses the variable TEXPKS. If that is not set, it uses TEXFONTS. If that is not set, it uses the system default:
.:/usr/local/lib/texmf/fonts//pk//

See **tex**(1) for the details of the searching.

SEE ALSO

gftopk(1), **gftype**(1), **pktype**(1).
Donald E. Knuth et al., *METAFONTware*.

AUTHORS

Tomas Rokicki wrote the program. Pierre MacKay adapted it for compilation with **web2c**.

NAME

pktype − translate a packed font file for humans to read

SYNTAX

pktype *pk_file_name* [*output_file_name*]

DESCRIPTION

The **pktype** program translates a packed font file (*pk*) (output by, for example, **gftopk**(1) to a file that humans can read. It also serves as a *pk* file-validating program (i.e., if **pktype** can read it, it's correct) and as an example of a *pk*-reading program for other software that wants to read *pk* files.

The *pk_file_name* on the command line must be complete. Because the resolution is part of the extension, it would not make sense to append a default extension as is done with TEX or DVI-reading software. If no *output_file_name* is specified, the output goes to the *stdout*.

The output file gives a compact encoding of the packed encoding, using conventions described in the source code. Run lengths of black pixels alternate with parenthesized run lengths of white pixels, and brackets are used to indicate when a row should be repeated.

ENVIRONMENT

pktype looks for *pk_file_name* using the environment variable PKFONTS. If that is not set, it uses the variable TEXPKS. If that is not set, it uses TEXFONTS. If that is not set, it uses the system default:
.:/usr/local/lib/texmf/fonts//pk//

See **tex**(1) for the details of the searching.

SEE ALSO

dvitype(1), **gftopk**(1), **gftype**(1).
Donald E. Knuth et al., *METAFONTware*.

AUTHORS

Tomas Rokicki wrote the program. Donald E. Knuth originally ported it to Unix. Pierre MacKay adapted it for compilation with **web2c**.

NAME

pltotf − convert property list files to TeX font metric (tfm) format

SYNOPSIS

pltotf [**−verbose**] *pl_file_name tfm_file_name*

DESCRIPTION

The **pltotf** program translates a (human-oriented) WEB property list file to a (program-oriented) TeX font metric file. Thus, after editing the property list file, a TFM file can be generated for use with, for example, **tex**(1).

Both the *pl_file_name* and the *tfm_file_name* must be complete; no adding of default extensions or path searching is done.

OPTIONS

Without the **−verbose** option, **pltotf** operates silently. With it, a banner and progress report are printed on *stdout*.

SEE ALSO

tftopl(1).
Donald E. Knuth, *TEXware*.
Donald E. Knuth, *The METAFONTbook*, (Volume C of *Computers and Typesetting*), Addison-Wesley, 1986, ISBN 0-201-13445-4.

AUTHORS

Donald E. Knuth wrote the program. It was published as part of the *TEXware* technical report, available from the TeX Users Group. Howard Trickey, Pavel Curtis, and Richard Furuta originally ported it to Unix.

NAME

Psfig/TeX (1.10) − PostScript figures in TeX

SYNTAX

\input psfig

DESCRIPTION

Psfig/TeX is a macro package for TeX, as well as LaTeX, that facilitates the inclusion of PostScript figures into (La)TeX documents. With the help of a compatible postprocessor, PostScript figures are automatically scaled and positioned on the page, and the proper amount of space is reserved. Figures can be presented as traditional broken-out displays, and Custom characters may be created and used freely throughout a document.

The **dvips** program developed by Tomas Rokicki has full **psfig** support. The **OzTeX** package has support for all **psfig functions except compressed figures.**

USAGE

To include a figure with **psfig**, include the **psfig** style at the top of your document. In LaTeX: ‘‘**\documentstyle**[psfig,...]{article}’’; in TeX: ‘‘**\input** psfig.sty’’. Then invoke the macro ‘‘**\psfig**{figure=*filename*}’’, where *filename* is the name of a PostScript file. **Psfig** will automatically position the figure at the current point on the page, and reserve the proper amount of space.

OPTIONS

Options may be specified in the form ‘‘**\psfig**{figure=*filename*,*option*=*value*,*option*=*value*,...}’’; recognized options include **height**, **width**, **rheight**, **rwidth**, **clip**, and **angle**.

height — The ‘‘**height**=*dimen*’’ option specifies how tall the figure should be on the page. When no **width** is specified, the figure is scaled equally in both dimensions.

width — The ‘‘**width**=*dimen*’’ option specifies how wide the figure should be on the page. When no **heigth** is specified, the figure is scaled equally in both dimensions.

If no **height** or **width** option is given, the figure’s ‘natural’ size will be used. By listing both a **height** and **width**, figures can be scaled disproportionately.

rheight — There are two sizes associated with each **psfig** figure: the size at which it is to be printed on the page and the size it reserves in TeX. This latter size is appropriately termed the reserved size, and is expressed as an option of the form ‘‘**rheight**=*dimen*’’, to reserve a vertical size of *dimen*. If omitted, the reserved size defaults to the real size. Some special effects need to be transparent to TeX and thus have a zero reserved size, such as a grey box enclosing a paragraph.

rwidth — To reserve a width in TeX, the ‘‘**rwidth**=*dimen*’’ option is used.

clip — The ‘‘**clip**=t’’ option sets the PostScript clipping path so that no marks will show up outside the declared bounding box. This option allows the user to isolate part of a larger figure, by specifying a new bounding box: ‘‘**\psfig**{figure=figs/piechart.ps,height=2in,bbllx=306bp,% bblly=396bp,bburx=486bp,bbury=546bp,clip=}’’

angle — Figures can be rotated by **psfig** using the ‘‘**angle**=*degrees*’’ option. Example: ‘‘**\psfig**{figure=figs/rosette.ps,height=1.0in,angle=90}’’.

AUTOSCALING

With autoscaling, some rotated figures come out smaller because the diagonal of their bounding box is longer than their **height** or **width** alone. This behavior can be disabled with **\psscalefirst**, and re-enabled with **\psrotatefirst**. With **\psscalefirst** a new **height** and **width** are computed after the bounding box. While the rotated figures will all come out at the same size, their reserved sizes will be different, thus they may not be aligned correctly.

SOURCES OF FIGURES

Any PostScript program can be used as a **psfig** figure, as long as it adheres to the Encapsulated PostScript Format (EPSF), which dictates a set of forbidden operators and a comment structuring convention. **Psfig** is actually less restrictive than the EPSF requirements; it can handle some of the "non-compliant" postscript operators, and the only required comment is the Bounding Box comment (see below).

BOUNDING BOXES

To properly translate and scale a figure **psfig** must know its ‘natural’ position on the page; this information is present in the ‘bounding box comment’ of a PostScript program. This is a line of the form ‘‘%%BoundingBox: *bbllx bblly bburx bbury*’’ in the header of the file.

The bounding box specifies four coordinates of a rectangle that defines the outermost edges of the PostScript figure: the lower-left *x* coordinate (*bbllx*), the lower-left *y* coordinate (*bblly*), the upper-right *x* coordinate (*bburx*), and the upper-right *y* coordinate (*bbury*). This rectangle must be present in any file to be used as a **psfig** figure. Most modern graphics packages produce (encapsulated) PostScript files which contain a ‘‘%%BoundingBox:’’ comment line in the header. All *bb*... values are in points, relative to the default transformation matrix.

MISSING BOUNDING BOX

When no bounding box comment line is present in your PostScript file, you can edit one in. The only mandatory PostScript convention is that the first line of the file should begin with the characters ‘%!’. A good place for the bounding box comment is the second line of the file.

The bounding box values may also be specified directly in the **\psfig** argument, using clauses of the form ‘‘bbllx=*bbllx*,bblly=*bblly*,...’’, in which case the figure file is not searched for the bounding box.

To control the amount of white space around a figure in a (La)TeX document, it may be necessary to experiment with the bounding box parameters *bb*..., also in cases where the bounding box is provided by the plot or graphics package.

DRAFT FIGURES

Some PostScript figures take quite a long time to transmit and print; for these figures a draft mode is available to speed printing of draft versions of the document. A figure printed in draft mode will appear as an empty box. The macro **\psdraft** will switch into draft mode, and all subsequent **psfig** macros will produce draft figures until reaching the macro **\psfull**, which switches out of draft mode.

No **\special** commands are used in draft mode, so a draft document can be previewed using *any* DVI viewer. The printing of boxes in draft mode can be disabled/enabled with **\psnodraftbox** and **\psdraftbox**.

SILENT MODE

Normally, **psfig** will print advisory messages to remind you that it is including figures as (La)TeX processes a document. This behavior can be disabled with **\pssilent**, and re-enabled with **\psnoisy**.

COMPRESSED POSTSCRIPT FILES

As off version 1.9, **psfig** allows the inclusion of compressed PostScript files (when using **dvips**).

To compress PostScript files, the shell script **pscompress** is used. **Pscompress** produces two files: *filename*.bb and *filename*.Z (or *filename*.gz, depending if **gzip** is available or not). The first file contains the bounding box comment only, while the second file contains the actual compressed PostScript file. Usage: ‘‘% **pscompress** *filename*’’.

When (La)TeX is run on *your_document*.tex, the file *filename*.bb is scanned for the bounding box comment, to allow (La)TeX to properly reserve the right size for the figure. Only when the file *your_document*.dvi is printed by means of **dvips**, the actual PostScript file *filename*.Z (or *filename*.gz) is decompressed (**zcat**-ed or **gunzip**-ed) and included in *your_document*.ps.

MACINTOSH FIGURES

Since the Macintosh drawing applications produce PostScript, they can be used to create figures. However, the PostScript produced by most Macintosh applications is often not well suited to be included directly as a **psfig** figure, unless it is saved as an "EPSF compliant" file. If the file is not "EPSF compliant" then the postscript may have to be edited before being used as an included figure. See the **psfig** tool **cleanfig** and the sample document **macdemo.tex** for guidance.

Non-EPSF Macintosh PostScript files often require a ‘‘Laserprep’’ prolog in order to be properly printed. A laserprep prolog can be included with the **dvips** special header command at the top of your (La)TeX file: ‘‘**\special**{header=lprep70.procs}’’. **Psfig** provides ‘lprep68.procs’, as well as ‘lprep70.procs’, to suit various versions of the Macintosh applications.

FIGURE SEARCH PATH

Psfig first searches in the current directory for a figure (or in the specified directory if given an absolute path). If it fails to find the figure in the current directory, it optionally searches a search path of figure directories to see if the figure is present. To specify the figure search path, use "\psfigurepath{dir1:dir2:...:dirn}", where dir1...dirn are the directories figures are to be found in.

PROBLEMS

Problem: The page with graphics is not printed.

Solution: Remove lines containing ‘**showpage**’ and/or ‘**stop**’ from the PostScript graphics file. Lines of this kind are usually found near the end of the file.

Problem: The figure is positioned in weird places.

Generally, this is an indication that the PostScript ‘‘%%BoundingBox:’’ comment is missing. It may also be that a translation takes places *after* the bounding box was determined.

Solution: Edit a bounding box comment into the PostScript file and/or experiment with the *bb...* values (see: MISSING BOUNDING BOX above).

BUGS

The **\psfig** macro is (unfortunately) sensitive to whitespace, and will be confused by any extra spaces or newlines in its argument.

The LaTeX **letter** style must be loaded after the **\psfig** macros due to naming conflicts. This can be forced by including psfig with an explicit **\input** command before the **\documentstyle** command.

ACKNOWLEDGEMENTS

This work was done while the author was with the Department of Computer and Information Science, University of Pennsylvania. Ned Batchelder co-developed the original troff version of this program with the author, and was responsible for much of the overall design. For more detailed information on the original version of see *Psfig − A Ditroff Preprocessor for PostScript Figures* in the USENIX 87 proceedings, or *Bringing troff up to speed* in the July 1987 issue of Unix Review.

Greg Hager provided the initial pure-TeX implementation of **psfig**. J. Daniel Smith of Schlumberger CAD/CAM implemented the rotation feature and improved the file scanning routines, using certain code fragments from Tom Rokicki’s **dvips** program.

Thanks to Sake Hogeveen (hogeveen.fys.ruu.nl) for editing this man page.

AUTHOR

Trevor Darrell

SEE ALSO

tex(1), latex(1), dvips(1), psfig-troff(1)

A printed version of the psfig-tex manual may be obtained by typing
‘‘% **latex** psfig-doc’’, then ‘‘% **dvips** psfig-doc’’,
and finally printing the resulting psfig-doc.ps file on your nearest PostScript printer. The manual provides many examples with real-life PostScript graphics, see the text of psfig-doc.tex for more usage examples.

NAME

tangle − translate WEB to Pascal

SYNOPSIS

tangle *webfile[.web]* [*changefile[.ch]*]

DESCRIPTION

The **tangle** program converts a WEB source document into a Pascal program that may be compiled in the usual way with the on-line Pascal compiler (e.g., **pc**(1)). The output file is all in lower case and packed into lines of 72 characters or less, with the only concession to readability being the termination of lines at semicolons when this can be done conveniently.

The WEB language allows you to prepare a single document containing all the information that is needed both to produce a compilable Pascal program and to produce a well-formatted document describing the program in as much detail as the writer may desire. The user of WEB must be familiar with both TEX and Pascal. WEB also provides a relatively simple, although adequate, macro facility that permits a Pascal program to be written in small easily-understood modules.

The command line should have either one or two names on it. The first is taken as the WEB file (and *.web* is added if there is no extension). If there is another name, it is a change file (and *.ch* is added if there is no extension). The change file overrides parts of the WEB file, as described in the WEB system documentation.

The output files are a Pascal file and a string pool file, whose names are formed by adding *.p* and *.pool* respectively to the root of the WEB file name.

SEE ALSO

pc(1), **pxp**(1) (for formatting **tangle** output when debugging), **tex**(1).

Donald E. Knuth, *The WEB System of Structured Documentation*.

Donald E. Knuth, *Literate Programming*, Computer Journal **27**, 97−111, 1984.

Wayne Sewell, *Weaving a Program*, Van Nostrand Reinhold, 1989, ISBN 0-442-31946-0.

Donald E. Knuth, *TEX: The Program* (Volume B of *Computers and Typesetting*), Addison-Wesley, 1986, ISBN 0-201-13437-3.

Donald E. Knuth, *METAFONT: The Program* (Volume D of *Computers and Typesetting*), Addison-Wesley, 1986, ISBN 0-201-13438-1.

These last two are by far the largest extant examples of WEB programs.

There is an active Internet electronic mail discussion list on the subject of literate programming; send a subscription request to *litprog-request@shsu.edu* to join.

AUTHORS

WEB was designed by Donald E. Knuth, based on an earlier system called DOC (implemented by Ignacio Zabala). The **tangle** and **weave** programs are themselves written in WEB. The system was originally ported to Unix at Stanford by Howard Trickey, and at Cornell by Pavel Curtis.

NAME

tex, virtex − text formatting and typesetting

SYNOPSIS

tex [first line]

virtex [first line]

DESCRIPTION

TEX formats the interspersed text and commands contained in the named files and outputs a typesetter independent file (called *DVI*, which is short for *DeVice Independent*). TEX capabilities and language are described in *The TEXbook*.

TEX is normally used with a large body of precompiled macros, and there are several specific formatting systems, such as LATEX, which require the support of several macro files. The basic programs as compiled are called **initex** and **virtex**, and are distinguished by the fact that **initex** can be used to precompile macros into a *.fmt* file, which is used by *virtex*. On the other hand, **virtex** starts more quickly and can read a precompiled *.fmt* file, but it cannot create one. It is the version of TEX which is usually invoked in production, as opposed to installation.

Any arguments given on the command line to the TEX programs are passed to them as the first input line. (But it is often easier to type extended arguments as the first input line, since Unix shells tend to gobble up or misinterpret TEX's favorite symbols, like backslashes, unless you quote them.) As described in *The TEXbook*, that first line should begin with a filename or a \controlsequence. The normal usage is to say

tex paper

to start processing *paper.tex*. The name *paper* will be the ‘‘jobname’’, and is used in forming output filenames. If TEX doesn't get a filename in the first line, the jobname is *texput*. The default extension, *.tex*, can be overridden by specifying an extension explicitly.

If there is no *paper.tex* in the current directory, TEX will look through a search path of directories to try to find it. If *paper* is the ‘‘jobname’’, a log of error messages, with rather more detail than normally appears on the screen, will appear in *paper.log,* and the output file will be in *paper.dvi.* The system library directory */usr/local/lib/texmf/tex* contains the basic macro package *plain.tex,* described in *The TEXbook*, as well as several others. Except when *.fmt* files are being prepared it is unnecessary to *\input plain*, since almost all instances of TEX begin by loading *plain.fmt*. This means that all of the control sequences discussed in *The TEXbook* are known when you invoke **tex**. For a discussion of *.fmt* files, see below.

The *e* response to TEX's error prompt causes the system default editor to start up at the current line of the current file. The environment variable TEXEDIT can be used to change the editor used. It can contain a string with "%s" indicating where the filename goes and "%d" indicating where the decimal line number (if any) goes. For example, a TEXEDIT string for **vi** can be set with the **csh** command

setenv TEXEDIT "/usr/ucb/vi +%d %s"

A convenient file in the library is *null.tex*, containing nothing. When TEX can't find a file it thinks you want to input, it keeps asking you for another filename; responding ‘null’ gets you out of the loop if you don't want to input anything. You can also type your EOF character (usually control-D).

The **initex** and **virtex** programs can be used to create fast-loading versions of TEX based on macro source files. The **initex** program is used to create a *format (.fmt)* file that permits fast loading of fonts and macro packages. After processing the fonts and definitions desired, a *\dump* command will create the format file. The format file is used by **virtex**. It needs to be given a format filename as the first thing it reads. A format filename is preceded by an &, which needs to be escaped with \ , or quoted, to prevent misinterpretation by the Unix shell if given on the command line.

Fortunately, it is no longer necessary to make explicit references to the format file. The present version of TEX, when compiled from this distribution, looks at its own command line to determine what name it was called under. It then uses that name, with the *.fmt* suffix appended, to search for the appropriate format file. During installation, one format file with the name *tex.fmt,* with only the *plain.tex* macros defined, should have been created. This will be your format file when you invoke **virtex** with the name **tex**. You can also create a file *mytex.fmt* using **initex**, so that this will be loaded when you invoke **virtex** with the name **mytex**. To make the whole thing work, it is necessary to link **virtex** to all the names of format files that you have prepared. Hard links will do for system-wide equivalences and Unix systems which do not use symbolic links. Symbolic links can be used for access to formats for individual projects. For example: **virtex** can be hard linked to **tex** in the general system directory for executable programs, but an individual version of TEX will more likely be linked to a private version by a symbolic link:

ln −s /usr/local/bin/virtex $HOME/bin/mytex

Another approach is to set up an alias using, for example, **csh**(1):

alias mytex virtex \&myfmt

Besides being more cumbersome, however, this approach is not available to systems which do not accept aliases. Finally, there is a program known as **undump**(1) which takes the headers from an *a.out* file (e.g., **virtex**) and applies them to a core image which has been dumped by the Unix quit signal. This is very system-dependent, and produces extremely large files when used with a large-memory version of TEX. This can produce executables which load faster, but the executables also consume more disk space.

When looking for a font *f*, TEX (and its companion programs) first look for a file starting with *f* in the various font directories (see the next section). If no such file is found, it then looks for a file *texfonts.map* in each of the font directories in turn. Each non-blank non-comment line of *texfonts.map* specifies mappings from one name to another. (Comments start with % and continue to the end of the line.) The target name is the first word (words are separated by spaces or tabs) and the source name is the second. (Subsequent words are ignored, so that information intended for other programs can be given there.) Thus, going back to *f* for a moment, if TEX reads a *texfonts.map* entry that looks like *g f* it will then search for a font file starting with *g*.

ENVIRONMENT

See the Kpathsearch library documentation (the ‘Path specifications’ node) for precise details of how the environment variables are used.

One caveat: In most TEX formats, you cannot use ˜ in a filename you give directly to TEX, because ˜ is an active character, and hence is expanded, not taken as part of the filename. Other programs, such as METAFONT, do not have this problem.

All the programs in the **web2c** distribution (as well as some others) use this same search method.

Normally, TEX puts its output files in the current directory. If any output file cannot be opened there, it tries to open it in the directory specified in the environment variable TEXMFOUTPUT. There is no default value for that variable. For example, if you say *tex paper* and the current directory is not writable, if TEXMFOUTPUT has the value */tmp*, TEX attempts to create */tmp/paper.log* (and */tmp/paper.dvi*, if any output is produced.)

TEXINPUTS	Search path for *input* and *openin* files. This should probably start with “.”, so that user files are found before system files. Default: *.:/usr/local/lib/texmf/tex//*
TEXFONTS	Search path for font metric (*.tfm*) files. Default: *.:/usr/local/lib/texmf/fonts//tfm*
TEXFORMATS	Search path for format files. Default: *.:/usr/local/lib/texmf/ini*
TEXPOOL	search path for **initex** internal strings. Default: */usr/local/lib/texmf/ini*
TEXEDIT	Command template for switching to editor. Default: *emacs +%d %s*
MAKETEXTEX	Arguments to pass to the *MakeTeXTeX* script before the filename to create. None by default. (If set, also implies invoking *MakeTeXTeX*.)
USE_MAKETEXTEX	If set, a program *MakeTeXTeX* is invoked when TEX cannot find an input file (before it complains about “can’t find file”). If neither MAKETEXTEX nor USE_MAKETEXTEX are set, whether *MakeTeXTeX* is invoked is the choice of installer.
MAKETEXTFM	Analogous.
USE_MAKETEXTFM	Analogous.

FILES

/usr/local/lib/texmf/ini/tex.pool	Encoded text of TEX's messages.
/usr/local/lib/texmf/fonts/texfonts.map	Filename mapping definitions.
/usr/local/lib/texmf/fonts//.tfm*	Metric files for TEX's fonts.
/usr/local/lib/texmf/fonts//.nnn{gf,pk}*	Character bitmaps for various devices. These files are not used by TEX.
/usr/local/lib/texmf/ini/.fmt*	Predigest TEX format (.fmt) files.
/usr/local/lib/texmf/tex//plain.tex	The basic macro package described in the *TEX*book.

SEE ALSO

mf(1), **undump**(1),
Donald E. Knuth, *The TEXbook*, Addison-Wesley, 1986, ISBN 0-201-13447-0.
Leslie Lamport, LATEX – *A Document Preparation System*, Addison-Wesley, 1985, ISBN 0-201-15790-X.
Michael Spivak, *The Joy of TEX*, 2nd edition, Addison-Wesley, 1990, ISBN 0-8218-2997-1.
TUGboat (the journal of the TEX Users Group).

TRIVIA

TEX, pronounced properly, rhymes with "blecchhh." The proper spelling in typewriter-like fonts is "TeX" and not "TEX" or "tex."

AUTHORS

TEX was designed by Donald E. Knuth, who implemented it using his WEB system for Pascal programs. It was ported to Unix at Stanford by Howard Trickey, and at Cornell by Pavel Curtis. The version now offered with the Unix TEX distribution is that generated by the WEB to C system (**web2c**), originally written by Tomas Rokicki and Tim Morgan.

NAME

tftopl − convert TeX font metric (tfm) files to property lists

SYNOPSIS

tftopl [**−charcode-format**=*format*] [**−verbose**] *tfm_file_name* [*pl_file_name*]

DESCRIPTION

The **tftopl** program translates a (program-oriented) font metric file to a (human-oriented) property list file. Thus, you can edit the contents of the TFM files, if the font designer has not done his or her job properly, or if you're encountering strange difficulties, or if you're just curious. It also serves as a TFM-file validating program, i.e., if no error messages are given, the input file is correct.

The *pl_file_name* must be complete; no adding of default extensions or path searching is done. If the *pl_file_name* is not given, standard output is used. The *tfm_file_name* is extended with *.tfm* if it lacks a suffix. Also, path searching is done for the *.tfm* file using the environment variable TEXFONTS. If that is not set, **tftopl** uses the system default:
.:/usr/local/lib/texmf/fonts//tfm

See **tex**(1) for the details of the searching.

OPTIONS

The argument *format* to **−charcode-format** specifies how character codes are output in the PL file. By default, only letters and digits are output using the **C** integer code (i.e., in ASCII); the others are output in octal. (Unless the font's coding scheme starts with **TeX math sy** or **TeX math ex**, in which case all character codes are output in octal.) If *format* is **ascii**, all character codes that correspond to graphic characters, besides the left and right parentheses, are output in ASCII. Finally, if *format* is **octal**, all character codes are output in octal.

Without the **−verbose** option, **tftopl** operates silently. With it, a banner and progress report are printed on *stdout*.

SEE ALSO

pltotf(1),
Donald E. Knuth, *TEXware*.
Donald E. Knuth, *The METAFONTbook* (Volume C of *Computers and Typesetting*), Addison-Wesley, 1986, ISBN 0-201-13445-4.

AUTHORS

Donald E. Knuth wrote the program. It was published as part of the *TEXware* technical report, available from the TEX Users Group. Howard Trickey and Pavel Curtis originally ported it to Unix.

NAME

tgrind − grind nice program listings using TeX

SYNOPSIS

tgrind [**−f**] [**−**] [**−n**] [**−h** header] [**−d** <description file>] [**−l<language>**] [**−r**] [**−s** <size>] name ...

DESCRIPTION

Tgrind formats program sources in a nice style using *TeX(1)*. Comments are placed in italics, keywords in bold face and strings in typewriter font. Source file line numbers appear in the right margin (every 10 lines). The start of a function is indicated by the function name in large type in the right margin.

In regular mode *tgrind* processes its input file(s) and passes them to *TeX(1)* for formating and output.

In format mode (i.e., when the **-f** flag is used), *tgrind* processes its input file(s) and writes the result to standard output. This output can be saved for later editting, inclusion in a larger document, etc.

The options are:

−f forces format mode

− take input from standard input

−n don't boldface keywords

−h specifies text to go on the left side of every output page (default is none)

−d specifies the language definitions file (default is /usr/local/lib/vgrindefs)

−l specifies the language to use. Currently known are PASCAL (−lp), RATFOR (−lr), Modula-2 (−lm2), MODEL (−lm), C (−lc or the default), ISP (−li), Yacc (−ly), Prolog (−lprolog), Icon (−lI), TeX (−ltex), CSH (−lcsh), and Bourne Shell (−lsh).

−r forces landscape format for output (the default is portrait).

−s specifies the fontsize of the main text in points (default is 10). The available sizes are 8, 9 and 10.

FILES

tgrindmac.tex	macro package
tgrindmacl.tex	macro package (for landscape output)
/usr/local/lib/tfontedpr	preprocessor
/usr/local/lib/vgrindefs	language descriptions

AUTHOR

Van Jacobson, Lawrence Berkeley Laboratory (based on "vgrind" by Dave Presotto & William Joy of UC Berkeley).
Eric Picheral, CICB, added the −r and −s options.

SEE ALSO

tex(1), vgrind(1), vgrindefs(5)

BUGS

The marginal−function−name mechanism depends on the quality of the language description in *vgrindefs*. The distributed *vgrindefs* fails to recognize many legal C function declarations.

Arbitrary formatting styles for programs mostly look bad. The use of spaces to align source code often fails miserably (because of the variable width output font). If you plan to *tgrind* your program, try to use tabs.

The indexing option of *vgrind* is missing. However, nice indices can be generated using the **-v** option of *ctags* together with appropriate TeX macros.

The **-f** flag means different things to *tgrind* and *vgrind*.

NAME

vftovp − convert virtual font (vf) files to virtual property lists

SYNOPSIS

vftovp [**−charcode-format**=*format*] [**−verbose**] *vf_file_name tfm_file_name* [*vpl_file_name*]

DESCRIPTION

The **vftovp** program translates a (program-oriented) virtual font file and its accompanying TEX font metric file to a (human-oriented) property list file. Thus, the virtual font can be edited, and its exact contents can be displayed mnemonically. It also serves as a VF-file validating program; if no error messages are given, the input files are correct.

The *vpl_file_name* must be complete; no adding of default extensions is done.

OPTIONS

The argument *format* to **−charcode-format** specifies how character codes are output in the PL file. By default, only letters and digits are output using the **C** integer code (i.e., in ASCII); the others are output in octal. (Unless the font's coding scheme starts with **TeX math sy** or **TeX math ex**, that is, when all are output in octal.) If *format* is **ascii**, all character codes that correspond to graphic characters, besides the left and right parentheses, are output in ASCII. On the other hand, if *format* is **octal**, all character codes are output in octal.

Without the **−verbose** option, **vftovp** operates silently. With it, a banner and progress report are printed on *stdout*.

ENVIRONMENT

vftovp looks for *vf_file_name* using the environment variable VFFONTS. If that is not set, it uses the variable TEXFONTS. If that is not set, it uses the system default:
.:/usr/local/lib/texmf/fonts//vf

See **tex**(1) for the details of the searching.

SEE ALSO

pltotf(1), **tftopl**(1), **vptovf**(1).

AUTHORS

Donald E. Knuth wrote the program, based in part on an idea of David Fuchs, starting with the code for **tftopl**(1). Karl Berry adapted it for compilation with **web2c**.

NAME

virtex − virgin TeX for production typesetting

DESCRIPTION

virtex is a version of the TEX program that has no preloaded macro packages, but is capable of loading a special preformatted binary file, called a *format* (*.fmt*) file produced by the **initex**(1) program.

Preprocessing of large macro files and font definitions by **initex** makes it possible to rapidly load complex packages, like plain TEX, AMSTEX, AMSLATEX, or LATEX, without the overhead of macro parsing and file opening that would otherwise be experienced.

virtex can be invoked directly, providing it with a name of a format file specially prefixed by an ampersand:

virtex '&plain' filename

This causes it to load the binary file named *plain.fmt* from a directory in the standard TEXINPUTS search path, and then load the text file *filename* and process it. The quotes are needed to protect the ampersand from Unix shell interpretation.

Because this need occurs frequently, Unix **virtex** provides a convenient shorthand. If it is invoked under a name other than **virtex**, such as **footex**, it will behave as if it had been invoked as *virtex '&footex'* and first load a file named *footex.fmt*. Thus, **amslatex**, **amstex**, **lamstex**, **latex**, **slitex**, **tex**, **texinfo**, and others can all be file system links to **virtex**, without requiring additional disk space.

SEE ALSO

initex(1), **tex**(1).

NAME

vptovf − convert virtual property lists to virtual font metrics

SYNOPSIS

vptovf [**−verbose**] *vpl_file_name vfm_file_name tfm_file_name*

DESCRIPTION

The **vptovf** program translates a (human-oriented) property list file to a pair of (program-oriented) files in the VF (virtual font) and TFM (TEX font metric) formats. Thus, a virtual font file can be edited and its exact contents can be displayed mnemonically. New virtual fonts, which map characters as seen by TEX into an arbitrary sequence of low-level typesetting operations, can also be created in this way.

All three file names, *vpl_file_name, vf_file_name,* and *tfm_file_name,* must be complete; no adding of default extensions or path searching is done.

OPTIONS

Without the **−verbose** option, **vptovf** operates silently. With it, a banner and progress report are printed on *stdout*.

SEE ALSO

pltotf(1), **tftopl**(1), **vftovp**(1).

AUTHORS

Donald E. Knuth wrote the program, based in part on an idea of David Fuchs, starting with the code for **pltotf**(1). Karl Berry adapted it for compilation with **web2c**.

NAME

weave − translate WEB to TeX

SYNOPSIS

weave [**−x**] *webfile[.web]* [*changefile[.ch]*]

DESCRIPTION

The **weave** program is used to create a TEX file for viewing a WEB program. It takes appropriate care of typographic details like page layout and the use of indentation, italics, boldface, etc., and it supplies extensive cross-index information that it gathers automatically. The command line arguments are the same as for **tangle** except for the option: **−x** says to omit the index, module name list, and table of contents pages. (A *CONTENTS.tex* file will still be written when the TEX file is processed, however, unless some macros in *webmac.tex* are redefined.)

The output TEX file name is formed by adding *.tex* to the root of the WEB file name.

There are several macros that probably should be redefined by the programmer at the beginning of the WEB file. It is a good idea to set \title to the name of the program. And, to cause output of only changed modules, one can say \let\maybe=\iffalse (usually as the first change in the change file).

FILES

/usr/local/lib/texmf/tex/webmac.tex TEX macros used by **weave** output.

SEE ALSO

tangle(1) for references, authors, and other information.