NAME

mff − run METAFONT and GFtoPK to create members of TeX font families (formerly called mfjob)

SYNOPSIS

mff [**−nvRS** | **−MGTP** *template* | **−WCIJ** *string*=*mftext* | **−Z** *dpi*=*modename*
| **−eE** *mftext* | **−x** | **−k** *factor(s)* | **−p** *suffix* | **−a** *atsize(s)* | **−s** *factor(s)* | **−m** *magstep(s)* | **−wcij** *mftext* | **−z** *modename* | **−d** *dpi(s)* | **−f** *file* | **−D** | *font*]...

mff −V | **−h**

CONTENTS

DESCRIPTION

mff is a front-end to the METAFONT program (see *mf*(1)), and is intended to be useful for creating TeX fonts in batches. It will run METAFONT and GFtoPK (see *gftopk*(1)) and then transfer the files to the correct directories.

Family matters

mff supports conventions that can be used to maintain a family of fonts derived systematically from common program files. A family in this context is a group of related fonts designed together — for example, the Malvern family includes Malvern, Malvern bold extended and Malvern italic.

Instead of using one driver file for each variation within the family, a *generic driver file* named with the abbreviation of the family name (e.g., **ma.mf**) is read after the standard METAFONT parameter **designsize** and optionally others (such as **weight** or **slant**) have already been assigned values. Usually other 'ad hoc' parameters are calculated from these and then the character program files read in to create the font.

Fonts within the family will be given names formed from the family name abbreviation, a suffix indicating the variant type and size. The size goes at the end and can contain a '**p**' or '**m**' standing for a decimal point (there must be a digit before the 'decimal point' and also after it if it is '**p**'). An '**m**' indicates that the size is in millimetres rather than TeX points. For example, **ma35a10** is 'ten-point Malvern 35 light', **ma55a7p5** is '7.5-pt Malvern', and **ma74a4m2** is '4.2-mm Malvern 74 bold extended italic'. These do *not* have to have separate **.mf** driver files.

What mff does

For each *font* argument, *mff* removes the suffix giving a stub that should be the family name (and usually the generic driver name). For example, **ma74a12** would have 12 pt design size and style suffix **74a**; **mabxi4m2** has 4.2 mm design size and suffix **bxi**. Then it invokes METAFONT with a first line

> **\mode=***modename***; font_size** *size***;** *assignments to parameters***;** *extra mf text***; input** *driver***; bye.**

The *modename* tells METAFONT what sort of device the font is being created for. The **font_size** command sets the standard METAFONT variable **designsize** (see Appendix F of the *METAFONTbook*). The 'assignments to parameters' are generated from the suffix and a lookup table (see under *Parsing the suffix* below). The 'extra mf text' (set with the **−E** option) might include more **input** commands (other **input** commands can't be used before the first **input** because it would give the font the wrong name).

After METAFONT has been run, the GFtoPK program is invoked to create the bitmap file. The PK and TFM files are then renamed into the user's 'font area' directories and the GF file is removed.

The tables used to translate suffixes into METAFONT code and so on are all definable using command-line options. See under the **−f** option and **FILES** below for information on defining such tables sensibly.

Log files will be named after the generic driver file rather than the individual fonts, so that if there are many fonts from the same family in the command line only one log file is written — for the last font generated. Usually this is a good thing because it slows the accumulation of useless log files. If the font is made successfully then the log file usually is not wanted and if the font fails, *mff* terminates and leaves the log file with the error message in it.

Using mff with conventional driver files

To use *mff* to create a font using non-generic driver files, use the **−S** option to supress the parsing step.

To use *mff* with the **dc**/**ec** fonts you might try using **−e gensize=designsize**; the **generate** command they use is equivalent to **input**. With the Sauter fonts there is the problem that the generic driver has ‘**b−**’ or ‘**build−**’ prepended to it; as well as using **−e design_size=designsize** you will have to use the **−g** option to get around this (or rename the files). (I would claim that using the **font_size** macro and builtin variable **designsize** is more ‘obvious’ and sensible than making up a new name and using that. Also, why do neither of those systems appear to take any notice of the # convention for measurements in ‘sharp’ units?)

OPTIONS

The large number of options is a result of the admittedly eccentric decision to make the command language used to describe how to make fonts the same as the language of the command-line options.

Options and *font* arguments can be intermixed; they are read in left-to-right order. A plus replacing a hyphen is used to set something back to its default value. Flags that take an argument do not take an argument when used with a **+**. (See *argloop*(3) for more details.)

Choosing which font to create

An argument named *foo(s)* below can be a list of values (the list must still form one shell word, so should not include unquoted spaces). For example a valid *dpi(s)* would be ‘**118,300**’. Each font creation will be performed once for each value.

−a *atsize(s)*

This is used to calculate the scaling of following fonts to match a TEX ‘**at** *size*’ specification. An *atsize* is a size which is interpreted as points or millimetres depending on whether the following font name has an ‘**m**’ in the size. When the next font name is parsed, *mff* calculates the nearest magstep (see under **−m**) to the ratio *at-size* / *design-size* and sets the scaling to that value. For example, if it appears as ‘**−a 17 ma55a10**’, the scaling is set to magstep 3, which is 1.728, so it is equivalent to ‘**−m 3 ma55a10**’. **+a** sets the scaling to 1 (same as **+s** or **+m**).

−d *dpi(s)*

Sets the device for which fonts are being created. A *dpi* is the number of pixels per inch of the device, used to identify it in an table of ‘modenames’ (see under **−Z**). **+d** selects 200 DPI (the *lowres* device, which is defined in **plain.mf**).

−m *magstep(s)*

Sets the magnification at which METAFONT will be run to be magstep *magstep*, that is, 1.2 raised to the *magstep*th power. A *magstep* is either an integer, or **h** (meaning magstephalf = sqrt(1.2) = 1.095), or **−h** (meaning 0.913). **+m** sets the magnification to be 1.0 (same as **+a** or **+s**).

−s *scale(s)*

Sets the METAFONT magnification to *scale*. A *scale* can either be a number (possibly including a decimal point) less than 100.0, in which case it is taken as is, or over 100, in which case it is assumed to be a TEX **scaled** number and is divided by 1000. **+s** sets the magnification to 1.0 (same as **+a** or **+m**).

−Z *dpi=modename*

Adds an entry to the ‘modename’ table. This is used to map *dpi* values onto the METAFONT names for those devices. A modename is what goes after ‘**\mode=**’ in the METAFONT command line. If a modename starts with a double-quote character, as in

```
mff −Z ′500="zambo"′ . . .
```

then ‘**smode=**’ is used instead (this causes METAFONT to read a file **zambo.mf** rather than attempt to use a predefined mode — see Section 4 of Appendix B of the *METAFONTbook*). **−Z −** discards the ‘modename’ table. **+Z** restores the default ‘modename’ table (which contains only Knuth’s *lowres* device — not very useful).

−z *modename*

Overrides the modename derived from the ‘modename’ table and the *dpi* value. **+z** cancels **−z.**

Parsing the suffix

Names are parsed from right-to-left. After the size has been stripped off, for each of four tables of *suffix=mftext* pairs, if there is a match between a *suffix* and the end of the stub, then the *mftext* is inserted into the METAFONT command line and the *suffix* is stripped off. The *mftext* is expected to be an assignment or equation, such as **weight=1.25**. (Semicolons are automatically inserted between them.)

The resulting stub is taken to be the family name — and therefore the name of the generic driver file. If it ends in a hyphen, the hyphen is removed too — this is for when the basename would otherwise seem to end with a suffix. For example, if **fab** was a family name, you might use ‘**fab−12**’ to avoid having it split as **fa b 12 .** The font created will still be called **fab12.tfm**. (TEX font names should not contain hyphens.)

The tables are intended to provied three dimensions of ‘variant style’ − the *weight* (W) table gives an axis from ‘light’ to ‘bold’; the *condensedness* (C) table gives an axis from ‘condensed’ to ‘expanded’; and the *italic* (I) table chooses roman, italic or obliqued options. (Omitted suffixes standing for ‘regular’.)

The fourth table (J) has no fixed meaning, but might be used to select a subset of the possible characters in that style — for example, a full font might have programs for uppercase, lowercase, small-caps, math italic, accents, accented letters, punctuation and symbols: more than can be fitted into 256 characters. The J table might have entries for **msy** (mathematical symbols) and **csc** (caps-and-small-caps).

The tables are used in the order J, I, C, W; therefore they appear in the name in the order W, C, I, J. Thus **madcicsc14** might be ‘fourteen-point Malvern demibold condensed italic caps-and-small-caps’.

The meanings of the four tables are not cast in stone, of course.

−W *suffix=mftext*

Adds an association to table W. **+W** discards table W.

−w *mftext*

Overrides the text derived from the W table. The parsing is still performed. **+w** cancels **−w.**

The other three tables have corresponding option letters **−C**, **−c**, **−I**, **−J**, **−J** and **−j**.

−g *file* This overrides the deduced ‘generic driver file’ name. For example, to use a file **malvern.mf** for the generic driver rather than **ma.mf**, *mff* could be invoked with **mff −g malvern**. **+g** cancels **−g**.

−S Indicates that this is a fixed-*S*ize font — the *font_size* assignment is omitted from the METAFONT command line and the parsing of the W, C, I and J suffixes is not attempted. This is appropriate for standard fonts like **cmr12** which do not have the ability to be built at different sizes.

Templates

The following four options tell *mff* how to go about installing a font.

−G *template*

Tells *mff* how to invoke GFtoPK. If this is just a hyphen, the GF-to-pixel step is omitted. **+G** sets the GFtoPK template to ‘**gfto%p**’. The command is broken into words at whitespace. For example, **mff −g ′gfto%p −v′ . . .** will use GFtoPK in its verbose mode.

−M *template*

Tells how to invoke the METAFONT program. **+M** sets the METAFONT template to be ‘**mf**’.

−P *template*

Tells where pixel files ought to be deposited once created. A value of just a hyphen means do not rename the pixel files. If there is no **%f** in this template, then it is assumed to be the name of a directory and **%f.%n%p** (and possibly a slash) is implicitly appended to it. **+P** resets the pixel directory template to ‘**%f.%n%p**’. (On systems with unreasonably short filenames, the default suffix is ‘**%f/%n.%p**’.)

−T *template*
Tells where TFM files ought to be deposited once created. A value of just a hyphen means to not move the TFM files. If there is no **%f** then **%f.tfm** is implicitly appended. **+T** resets the TFM directory template to ‘**%f.tfm**’.

The following %-substitutions are made in templates supplied by the above commands:

%% A percent sign.

%b The basename of the current font.

%d The current *dpi* value.

%D The current *dpi* value multipled by the current magnification.

%f The current *font* name.

%g the current driver file name (usually the same as **%b**).

%k The current *factor* times 1000 (set with **−k**).

%n The current *dpi* value multipled by the current *factor* value times the current magnification — e.g., when making a 300-dpi PXL font, this will be 1500.

%p The pixel file suffix — usually ‘**pk**’.

%s The current magnification, times 1000.

%u Either **pt** or **mm** depending on which units the design size is assumed to be in.

%z The current modename — e.g., ‘**CanonCX**’ for a LaserWriter.

Miscellaneous

−D Prints (‘dumps’) the current values of the table and other variables used to determine the METAFONT command line.

−e *mftext*
Adds extra METAFONT text, which goes before the first **input** command. Multiple **−e** options accumulate (semicolons are implied between them). **+e** discards the accumulated extra text.

−E *mftext*
As **−e** except it goes after the first **input** command.

−f *file* The *file* is searched for in the directories listed in the MFFPATH environment variable and if a readable file is found, it is parsed for more options. An optional **.mff** suffix will be supplied if omitted. Files can be nested by using **−f** inside a file. If the *file* is just a hyphen then the standard input stream is used. **+f** rereads the initialization files (see under **FILES** below). See *argloop*(3) for details about the format of the files.

−h Prints a summary of the options.

−k *factor(s)*
Uses *factor(s)* to construct pixel file names in the style of old-fashioned PXL files. The number in the suffix of the pixel file is expected to be multiplied by the *factor*; presumably either GFtoPXL or something like *mftobdf* is being used which shrinks or magnifies pixel files in addition to the magnification METAFONT may be applying. This list of values is ignored unless **−x** is used. **+k** sets the factor to 5.0 (which is correct for GFtoPXL).

−n Does a dryrun — instead of running METAFONT and so on, the operations that would be performed are printed. Renaming of files is represented as **mv** *oldname newname*, and deletion of the GF file as **rm** *gffile*. (That is, the output resembles shell commands.) **+n** cancels **−n**.

−p *suffix*
Sets the suffix that will be used for pixel files. **+p** sets the pixel suffix to ‘**pxl**’ if **−x** is in effect, ‘**pk**’ otherwise.

−R Retain (that is, do not delete) the GF file after the GFtoPK step is performed. This is useful if the METAFONT commands was something like *echo*(1) which does not actually create a GF file in the first place!

−v Set ‘verbose’ mode. The parsing of options and reading of files will be described in detail. (Each line will start with a ‘**#**’ so that if **−n** is being used the result is still a reasonable shell script.) More **−v** flags cause more verbosity (thus **−vv** or **−vvv** etc.). Often used but rarely useful. **+v** cancels any number of **−v** options.

−V Print version number information.

−x Assume we are producing PXL files or something equally unpleasant, and so multiply pixel file suffixes by *factor(s)* set by **−k** above. **+x** turns off **−x**.

ENVIRONMENT

MFFINIT

This is parsed for options before the command-line is read. (Double-quotes can be used as in files.)

MFFPATH

This is a colon-separated list of directories in which files for the **−f** option can be found. If this is not set, just the current directory is searched.

PATH Used when searching for METAFONT and GFtoPK executables.

FILES

/mclab/pdc/lib/mff.rc
$HOME/.mffrc
./mff.rc

These files are read for options when *mff* starts up (or when requested with the **+f** option) before the MFFINIT variable and the command-line arguments. The system **mff.rc** can contain system-wide setups, like the local printing devices and standard suffixes. The user ‘rc’ files allow per-user and per-project tailoring.

∗**.mff** Other files containing *mff* options.

/mclab/pdc/src/mff/example.mffrc

An example *$HOME/.mffrc* file. (/mclab/pdc/src/mff is the directory *mff* source is in).

/mclab/pdc/src/mff/{cm,Ditko,mflogo}/*

Examples of METAFONT fonts designed to be used with *mff*.

∗**.mf** source files.

*driver***.log**

Log file created by METAFONT.

∗**.tfm** TEX font metric file, created by METAFONT and used by TEX.

∗**.**∗**gf** Generic font file, created by METAFONT and turned into a PK file by GFtoPK.

∗**.**∗**pk** Bitmap file, created from the GF file and used by DVI drivers such as *dvips*(1). (On systems with short filenames, a file *driver***.pk** is used to store the output of GFtoPK before it is renamed as per the **−P** option.)

AUTHOR

Damian Cugley <pdc@prg.ox.ac.uk>

Please send bug reports, fixes and comments to <malvern@prg.ox.ac.uk> — this is a mailing list that you can join by sending a request to me at the address <malvern-request@prg.ox.ac.uk>. The Malvern list is also for discussion of Malvern, a METAFONT font designed to take advantage of *mff*’s special features; it ought to be available from the same archive that *mff* was obtained from.

If you find *mff* amusing or useful, please send a pretty postcard to Damian Cugley, Computing Laboratory, 11 Keble Road, Oxford, UK OX1 3QD.

SEE ALSO

mf(1), gftopk(1), argloop(3), findfile(3)

The *mff* source files.

Donald E. Knuth, *The METAFONTbook.* Addison-Wesley, 1986. ISBN 0−201−13444−6.

Donald E. Knuth, *The TEXbook.* Addison-Wesley, 1984. ISBN 0−201−13448−9.

LIMITATIONS

The command-line passed to METAFONT is limited to 512 characters. The commands supplied with **−M** and **−G** are limited to 1024 characters when the %-substitutions are expanded.

NOTES

This program used to be called *mfjob* but has been renamed (as of version 2.8) to prevent confusion with the MFjob program that is part of Eberhard Mattes' emTEX distribution for MS-DOS and OS/2 computers. An immediate consequence is that the default suffix for files of options, and the name of 'rc' files, have changed. There are also some other minor incompatible changes incorportated at the same time.

For information on differences for *gcc*-based Atari ST systems, see *argloop*(3) and *findfile*(3).

BUGS

It seemed sensible at the time to indicate which mode to use with the dots-per-inch value because this number is generally much shorter than the corresponding mode name. Systems with printers with the same pixel density but otherwise different characteristics require the use of **−z**. This system also fails to make sense if a printer has a bizarre dots-per-inch value, such as 4000/3.