

Formatting More

Fm displays a text file on your screen a page at a time — something like the *more* program. But *fm* also formats the text: fills out lines with hyphenation and justification, and paginates. If the text has formatting markup, *Nroff* -man macros or T_EX control words, that will help.

At the bottom of each screen page, *fm* waits for you to read, hit the space bar for the next page. You can also type `q` to quit or `b` to view the previous page (you can only go one page back).

Fm has some more advanced capabilities, which I will be describing below, but they shouldn't interfere with you just putting text on the screen to read, if that's all you want to do. If you do want to go further, aside from reading what follows, please take a look at some of the example documents I've provided in the distribution. Including this document itself, which was formatted with *fm* — its file name is `fm-man.tex`.

Besides displaying text on the screen, *fm* can produce output in various other forms. If you are reading this on your screen using *fm* itself (having typed `fm fm-man.tex`), you will see the name '*fm*' displayed in cyan. Or, you might be reading a paper copy prepared by *fm* for a PostScript printer (with `fm -p 10 fm-man.tex | lpr`), in which case you will see the name printed in Times-Roman at 10 point size. As another alternative, I might have used *fm* to make an all ascii + Latin-1 file suitable for emailing or posting in a newsgroup (with `fm -fu fm-man.tex >README.fm`), and then you will see the name *fm* preceded and followed by underline characters, to indicate that it is supposed to be "italic". If you look at the source file for this, you will see `{\it fm}` for the italic 'fm's.

Similarly, I used `{\tt...}` for some words and phrases above, which consequently will appear green on the screen, in Courier in print, but flanked by double quotes in plain ascii output.

Before proceeding, I want to apologize here for naming the file containing this document with the suffix `.tex` and for my references below to "T_EX-commands", which might imply that *fm* is compatible with T_EX, and that T_EX could be used to format and print this document. Neither of those things is true. However, *fm* control words are mostly also T_EX control words and have related functions. For anyone familiar with Plain T_EX, it makes them easier to remember.

I. Paragraphs.

Paragraphs are sometimes separated in a text by a blank line and sometimes by indenting only the first line of each paragraph. *Fm* understands both these conventions. However, if there are blanks before every line, you need to pass *fm* the **-i** flag, which tells it to ignore indentation, otherwise it will think every line is the beginning of a paragraph. With **-i**, only blank lines are considered to separate paragraphs. *Fm* cannot deal with files that have constant white space at the left margin and **additional** indentation at the beginning of each paragraph—it will lump all the text together into one huge paragraph (with the **-i** flag), or treat each line as an independent paragraph (without that flag).

II. Text Format Options.

The default length to which lines will be filled is 78, which, figuring 12 characters to the inch, is 6.5 inches in width — that is the plain T_EX default `\hsize`. The line length can be changed with the **-w**`<n>` command line option, or with a T_EX command `\hsize <dim>`, where the `<dim>` is specified as in T_EX as a decimal fraction of a horizontal unit: `in`, `pt`, `em`, `pc`, `cm` (i.e., inches, points, ems, picas, and centimeters). (However, these are the only horizontal units *fm* knows.) Thus, for example, the default length could be specified as `\hsize = 6.5in`, or `\hsize 39em` (figuring two columns to the em).

If your screen display is wide enough, the left margin will be made the same as T_EX's default — one inch (72 points, or 12 columns) — and at least the same amount allowed for at the right. That takes up 102 columns. If it's not wide enough, the left margin is decreased enough so as to fit lines on the screen and balance the left and right margins (if possible). With an 80 column display and the default line width of 78, that gives a one column margin at the left. The left margin can be set with the **-l**`<n>` option, where the `<n>` refers to display columns. It can also be changed with the T_EX command `\hoffset <dim>`, where the `<dim>` is specified as above (it may be negative).

III. Input File Options.

Paragraphs in the input text are assumed to be separated by blank lines unless *fm* is given the **-i** flag on the command line, in which case each line starting with blanks or tabs starts a new paragraph.

Text files are treated differently according to their file name suffixes. When the suffix is `.tex`, output paragraphs have indented first lines, but otherwise there is no indent and blank lines separate paragraphs. With the **-x** flag, a file is treated as a T_EX file, regardless of the suffix, and T_EX markup is recognized.

When the suffix is `.man`, *man* macros are recognized, and when the suffix is `.c`, there is no formatting other than colorizing C language keywords and comments. The flag **-m** forces interpretation as a file with *man* macros, and the flag **-c** forces interpretation as a C file.

The simulation of *nroff* -*man* is pretty complete, so long as the input text sticks to the *man* macros and doesn't use *Nroff* commands directly. The simulation of T_EX is not like real T_EX at all, though *fm* is more or less upward compatible with T_EX, in the sense that if a file with the few T_EX commands that can be interpreted is passed through *fm* and looks ok on the screen, then T_EX should also accept it and give printout that looks sort of vaguely the same.

IV. Printing.

To generate PostScript code, just hand *fm* the **-p** flag with the pointsize you'd like and direct output to the printer or to a file. E.g., `fm -p10 fm-man.tex|lpr` for 10 point size Times. If you want Palatino instead of Times, add the **-b** flag: `fm -p12 -b fm-man.tex|lpr` would get 12 point Palatino printout. You can use fractional point sizes, but they are rounded off to the nearest 10th of a point.

Times and Palatino, in fact, are the only PS font families *fm* can deal with. The typefaces indicated in your document by `\rm`, `\bf`, `\it`, `\bi` are printed in Times or Palatino Roman, Bold, Italic, and BoldItalic, respectively. Courier and Courier-Oblique are used for `\tt` and `\sl`, and Helvetica for `\sa`. Justified lines are proportionally spaced, and superscripts and subscripts raised and lowered, respectively, by two points.

Characters in the math typeface use the PostScript Symbol font. I haven't provided symbolic names to access the symbols, so to use this facility, you just have to know that `a` prints as Greek alpha, for example. Although the Symbol font is variable width, *fm* forces the characters into the same fixed columns as Courier, so you may have to put space around symbols to keep them from running into surrounding characters.

The default height of a line is two points plus the point size, for 9 point size or larger, and one point plus the point size for smaller sizes, but you can change that with the `\baselineskip` command. The paper margin is one inch of white space all around unless you change the `\vsize` to add or remove lines at the bottom, `\hsize` to make the lines longer, `\hoffset` to displace the text area to the left or right, or `\voffset` to displace the text downward (for positive values) or upward on the page.

The units specified for the just mentioned commands which refer to absolute distances, like inches, points, picas, are scaled in terms of lines and columns according to the point size you've asked for, so that they really do refer to inches and so on. However, an em is always two columns and two exs is always one line.

Page numbers are put at the top center, two lines above the text area. You can have page numbers omitted altogether with the `\nopagenumbers` command, but there isn't any other way yet of changing the header line or any way to add a footer line.

The command line option **-d** with numeric argument giving a point size gets ordinary, non-PostScript output that might be suitable for a printer, in a pinch. The point size gives margins and line lengths that approximate what is required for Courier type on letter-size paper. You would ordinarily want to give the **-u** flag with the **-d** option, to get ascii interpretations of the typefaces.

V. Personalization.

To change the colors and modes used to display non-Roman typefaces, put a file similar to the text of this section in your home directory and name it `.fmconfig`. For each typeface whose display you'd like to change, give the name of the typeface – `roman`, `bold`, `italic`, `slanted`, `ttype`, `bitalic`, `sans`, `math`, `super`, and `sub` (also `prompt`) – followed immediately by a colon character. Then, after that, give the new color and/or mode you want to have displayed for this typeface. The color is given by one of the words `yellow`, `green`, `blue`, `cyan`, `red`, `magenta`, `white`. The mode is given by one of the words `bright`, `reverse`, `dim`, `blink`, `underscore`. The `prompt` keyword is not for a typeface, but for the `-MORE-` prompt.

In this example, defaults are given, so nothing would actually be changed:

```
bold: bright yellow
italic: cyan
slanted: bright blue
math: reverse green
ttype: green
roman: white
prompt: reverse white
```

Changing the color for `roman` may not look good or may leave the screen in an inappropriate mode.

To set flags here rather than on the command line, make a line with `flags` followed immediately with `:` (colon), then on the same line the flags you'd like to set. For instance, this line

```
flags: -jxi -w72
```

sets the line length to 72, says every file should be considered a TeX file, disables justification, and makes indentation in files count as a paragraph break.

You can also set some formatting default values here: `tenrm`, `hoffset`, `hsize`, `vsize`, `parindent`, `parskip`, `raggedright`, and `baselineskip`. Give the value just just as it would be given in a `.tex` file. For instance, this

```
\parskip = 2ex
```

will put a blank line before each paragraph. `tenrm` changes the way horizontal dimensions are interpreted (from 12 columns to the inch to 14.4 columns to the inch), so (if that's what you want) put that before `hsize` or `hoffset`.

VI. Hyphenation and Justification.

Fm uses T_EX's hyphenation method, so, though it is not applied in as sophisticated a way, it's pretty good. A command line flag **-h**<*n*> allows you to choose a level of hyphenation from 0, meaning no hyphenation at all, to 9 for the most hyphenation *fm* can do. The default level is 8.

Give the **-j** flag for right justification.

VII. Typeface Options.

Fm understands six text attributes, “typefaces”, corresponding to T_EX's Roman, *italic*, **bold**, *ttype*, *slanted*, and μ_ατ_η fonts, which you get with the T_EX commands `\rm`, `\it`, `\bf`, `\tt`, `\sl`, and `$`, respectively. They are displayed in white, cyan, bright yellow, green, bright blue, and reverse green, by default. There are two other attributes: **sans** and **bold italic** available with control words `\sa` and `\bi`. Superscripts, which you get with `^{\dots}`, and subscripts, which you get with `_{\dots}`, are also shown in different colors. (In print, they are raised and lowered slightly.) Here are superscripts and here are subscripts.

You can change the colors by putting a file `.fmconfig` in your home directory specifying the colors and modes you'd like to have. See the section on Personalization and the example file `sample.fmconfig` for details.

With the **-n** flag, only screen modes with no colors are used — `standout` for bold, `dim` for italic and slanted, `reverse` for `ttype` and `math`.

Yet another option, **-o**, is overstriking for bold and overprinting underline characters for italic and slanted, with other faces unmarked.

You can instead get strictly `ascii` output with the **-u** command line flag. Then, the typefaces are shown with flanking underlines for italic, asterisks for bold, double quotes for `ttype`, slashes for slanted, and single quotes for `math`. This flag also turns off the facility for showing special graphic characters, like the box drawing characters (e.g., `|`, `|`, `|`) that *fm* ordinarily uses for `|`, `\vdash`, `\dashv`, etc. If you prefer not to have the differences among typefaces shown at all, you can add the **-a** flag—that is, use `fm -ua file`.

VIII. Latin-1 and Graphic Characters.

The accented characters in Latin-1, ISO 8859-1, are displayed as is (in hopes that your terminal can show them) and printed appropriately. They can also be gotten using T_EX commands, as the following example illustrates:

Also, wé háve sóme acúte áccents, wè hàve sòme cùte gràve àccents, and wê hâve sòme cîrcûmfilêx âccents. Plus wë hàve sòme ümläut àccents. A dashed-word gets a dash longer than hyphen—and there is a long dash. More special symbols: ¶ for paragraph, §2a is section 2a. How's this → for a right arrow?

Here is the input text that produced the above passage:

Also, w\`e h\`ave s\`ome ac\`ute \`accents,
w\`e h\`ave s\`ome c\`ute gr\`ave \`accents,
and w\^e h\^ave s\^ome c\^irc\^umfil\^ex \^accents.
Plus w\`e h\`ave s\`ome \`uml\`aut \`accents.
A dashed--word gets a dash longer than hyphen---and there is a long
dash.
More special symbols: \P\ for paragraph, \S 2a is section 2a. How's
this \rightarrow\ for a right arrow?

For the screen, characters from the Alternate Character Set of vt100 compatible terminals are used for dashes, em dashes, horizontal and vertical lines, drawing boxes, and for arrow symbols.

IX. Automatic Numbering.

The `\exno` command returns a number that increases each time it's used. This is not compatible with plain T_EX, though it would be easy enough to define in T_EX, of course. For instance, the following: (1) peaches, (2) pears, and (3) pies, was produced from: `(\exno) peaches , (\exno) pears, and (\exno) pies.`

The sequence of numbers can be started at any number with `\exn<num>`. E.g., `\exn200, \exno, \exno` gives: 200, 201, 202. Then you can refer to a small range of numbers relative to the current number. After the preceding, `\exnl` ('l' for last) is 202, `\exnp` ('p' for previous) is 201, `\exne` ('e' for earlier) is 200, and looking forward to the next example, `\exnn` ('n' for next) is 203, `\exns` ('s' for subsequent) is 204, and `\exnf` ('f' for following) is 205. Numbers before `\exne` can be gotten with `\exnb-d` and after `\exnf` with `\exng-h`.

If you want roman numerals, start the sequence with 'i', 'I' (for caps), or with a negative number: `\exni, \exno, \exno` gives: i, ii, iii, `\exnI, \exno, ...` gives: I, II, III, IV, V, and `\exn-44, \exno, ...` gives: XLIV, XLV, XLVI, XLVII. (Those last numerals are still in caps because they were started in caps previously.)

Series can also start with 'a', using `\exna`: a, b, c, d, or with 'A', starting the series with `\exnA`: A, B, C, D, E, ...

There are four other independent numbering sequences that work similarly: `\sxno`, `\cxno`, `\fxno`, `\gxno`. The sections of this document are numbered with `\sxno`.

Unlike other T_EX control words, blank space after `\exno`, etc., is not required or ignored.

X. Snippets.

I've used the control word `\vbb` several places in this document to quote illustrative text without having the control words interpreted. After `\vbb`, and up to the next blank line, text is displayed or printed as is. You can also quote C code, with keywords and comments automatically recognized and shown in different typefaces, by enclosing the snippet of code with `\beginC` at the beginning and `\endC` at the end.

Here is some example C code:

```
/* count newlines in string; update linecount; issue warning if
 * blank line
 */
static void notenewlines(char *s)
{   int warn = 0;
    while (*s) {
        if (*s == '\n') {
            if (warn && !quiet_opt)
                ERROR("blank line within tree");
            else warn++;
            linenumber++;
        }
        else if (*s != ' ' && *s != '\t') warn = 0;
        s++;
    }
}
```

And now back to T_EX mode.

XI. Tables.

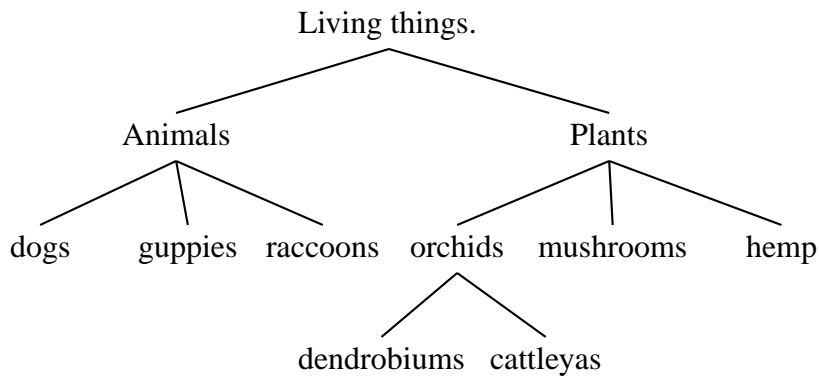
Columnar displays can be set up using the `\settabs` control word, which lets you set tabs either by specifying some number of equal width columns or by giving a sample line with the tab stops given with `&` characters. It's very much like T_EX's `\settabs` command.

Here's an example table, which looks better on the screen than it does printed, because I don't yet have a means of printing the Alternate Character Set box-drawing characters it uses. (Also, tabbing for PostScript printout still has some bugs, as you may notice.)

attribute	with -u	with -n	with -o
bold	*...*	mode md	overstruck
<i>italic</i>	..._	mode mh	ovstk ul
<i>slanted</i>	/.../	mode mh	ovstk ul
typewriter"..."		mode mr	overstruck
<i>bitalic</i>	[...]	mode md	overstruck
sans	{...}	none	normal
μσπη	'...'	mode mr	overstruck

XII. Trees.

Branching tree diagrams are constructed with the `\tree` command by giving a list of node names between `\tree` and `\endtree`. The indentation of the node names gives the structure: tree sisters have the same indentation and daughters have greater indentation than their mothers. Here is an example:



That example had this in the source file:

```

\tree
Living things.
  Animals
    dogs
    guppies
    raccoons
  Plants
    orchids
      dendrobiums
      cattleyas
    mushrooms
    hemp
\endtree

```

The absolute amount of indentation doesn't matter—only how the indentation of each line compares with that of others.

You can use an alternative notation with parentheses around each node name, if you want. The following gives the same result as the above:

```

\tree
(Living things.
  (Animals
    (dogs)
    (guppies)
    (raccoons)
  )
  (Plants
    (orchids
      (dendrobiums)
      (cattleyas)
    )
    (mushrooms)
    (hemp)
  )
)

```

```
\endtree
```

Using this method, only the parentheses matter, so the following more compact form works, too:

```
\tree
(Living things.
 (Animals (dogs) (guppies) (raccoons))
 (Plants (orchids (dendrobiums) (cattleyas)) (mushrooms) (hemp)))
\endtree
```

Using parentheses, you don't necessarily have to supply all the closing right parentheses at the very end of the tree, because *fn* will do that for you.

There are loads of options for doing trees—the tree interpreter has its own set of backslash commands for inverting trees or parts of them (which is fun), evening out the leaves, and so on. There is a separate manual on trees.

However, aside from special tree commands and T_EX commands for changing typefaces and getting special characters and symbols, most T_EX commands don't work inside a tree. `\hskip` and `\kern` do work, but only for positive skips or kerns.